Uncertainty quantification for vision regression tasks

Xuanlong Yu

université PARIS-SACLAY

Uncertainty quantification for vision regression tasks

Quantification d'incertitude pour les tâches de régression visuelle

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580
Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Sciences du traitement du signal et des images
Graduate School : Informatique et sciences du numérique
Référent : Faculté des sciences d'Orsay

Thèse préparée dans l'unité de recherche **SATIE** (Université Paris-Saclay, ENS Paris-Saclay, CNRS), sous la direction d'**Emanuel ALDEA**, Maître de Conférences, Université Paris-Saclay, et le co-encadrement de **Gianni FRANCHI**, Enseignant chercheur, ENSTA Paris, Institut Polytechnique de Paris.

Thèse soutenue à Paris-Saclay, le 11 décembre 2023, par

Xuanlong YU

Composition du Jury

Membres du jury avec voix délibérative

David PICARD Professeur, École des Ponts ParisTech	Président
Madalina OLTEANU Professeure, Université Paris Dauphine - PSL	Rapporteure & Examinatrice
Frédéric PICHON Professeur, Université d'Artois	Rapporteur & Examinateur
Pauline TROUVÉ-PELOUX Chargée de recherche, ONERA Saclay	Examinatrice
He WANG Professeur Associé, University College London	Examinateur
Jean-Emmanuel DESCHAUD Chargé de recherche, Mines Paris - PSL	Examinateur

THESE DE DOCTORAT

NNT : 2023UPASG094

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Quantification d'incertitude pour les tâches de régression visuelle

Mots clés : Apprentissage profond, quantification d'incertitude, vision par ordinateur, régression

Résumé : Ce travail se concentre sur la quantification de l'incertitude pour les réseaux de neurones profonds, qui est vitale pour la fiabilité et la précision de l'apprentissage profond. Cependant, la conception complexe du réseau et les données d'entrée limitées rendent difficile l'estimation des incertitudes. Parallèlement, la quantification de l'incertitude pour les tâches de régression a reçu moins d'attention que pour celles de classification en raison de la sortie standardisée plus simple de ces dernières et de leur grande importance. Cependant, des problèmes de régression sont rencontrés dans un large éventail d'applications en vision par ordinateur. Notre principal axe de recherche porte sur les méthodes post-hoc, et notamment les réseaux auxiliaires, qui constituent l'un des moyens les plus efficaces pour estimer l'incertitude des prédictions des tâches principales sans modifier le modèle de la tâche principale. Dans le même temps, le scénario d'application se concentre principalement sur les tâches de régression visuelle. En outre, nous fournissons également une méthode de quantification de l'incertitude basée sur le modèle modifié de tâche principale et un ensemble de données permettant d'évaluer la qualité et la robustesse des estimations de l'incertitude.

Title : Uncertainty quantification for vision regression tasks

Keywords : Deep learning, uncertainty quantification, computer vision, regression

Abstract : This work focuses on uncertainty quantification for deep neural networks, which is vital for reliability and accuracy in deep learning. However, complex network design and limited training data make estimating uncertainties challenging. Meanwhile, uncertainty quantification for regression tasks has received less attention than for classification ones due to the more straightforward standardized output of the latter and their high importance. However, regression problems are encountered in a wide range of applications in computer vision. Our main research direction is on post-hoc methods, and especially auxiliary networks, which are one of the most effective means of estimating the uncertainty of main task predictions without modifying the main task model. At the same time, the application scenario mainly focuses on visual regression tasks. In addition, we also provide an uncertainty quantification method based on the modified main task model and a dataset for evaluating the quality and robustness of uncertainty estimates.

Acknowledgements

I am grateful to my best friends and dear advisors, Emanuel Aldea and Gianni Franchi, for their warm supervision, support, advice, and availability. Special thanks to Madalina Olteanu and Frédéric Pichon for having accepted to review this work. I am also thankful to the other committee members: David Picard (president of the jury), Pauline Trouvé-Peloux, He Wang, and Jean-Emmanuel Deschaud, for their comments and evaluation of my work. I also want to thank Professor Philippe Xu and Sao Mai Nguyen for their time listening to my practice presentation.

I acknowledge Université Paris-Saclay for financing this thesis. I thank the laboratory SATIE and U2IS for providing me with comfortable workplaces; the former provides a warm office in winter, and the latter provides a cool one in summer. I thank all my colleagues from those two labs who talked to me because I'm not very outgoing. I also thank everyone who made me join them for lunch from time to time. The ones who did not ask are also nice because sometimes I prefer to eat alone as well. In any case, I want to thank Adrien, Alina, Anis, Clementine, Fatima, Gaël, Joris, Mohamed, Mouin, Olivier, and Rémy. Furthermore, thank you for your gifts, you are really warm and kind.

Next, I really want to thank my friends, including my smartest ex-cat, for accompanying me for different periods of this journey. They are Nice The Cat, Di Yang, Huiqin Chen, Zelong Wang, You Zuo, and Kunpeng Guo. Thank you for your discussion on (including but not limited to) cutting-edge AI techniques, meanings of life, mysteries of the universe, future plans, the content of the next meal, music, games, clothing, interesting videos, and so on.

I want to thank my friends who (including but not limited to) cared about me, communicated in academics, food and music, cooperated on papers, ate together, chatted with me online to reduce my boredom, played and traveled with me, during these years: Jialin Hao, Ziwei Zeng, Jinyuan Fan, Yujian Ma, Lijie Yao, Yangxintong Lv, Yu Zhao, Yueming Yang, Milena Yang, Jindong Gu, Dexiong Chen, Yingyi Chen, Xi Shen, Yuting Li, Yan Chen, Harshil Kanuga, Yajing Feng, Zhuoer Li, Xiaoxuan Hei, Heng Zhang, Shanshan Liu, Shuqi Yu, Siwen Lv, Shiwen Peng, Long Tian, Xina Wang, Qiong Wu, Bingqing Xie, Yunyun Sun, Haoming Zhan, Chengyu Zhao, Shiyang Yan, Zhe Zheng, Yanhao Li, Kaifeng Zou, etc. I thank the friends that I made during conferences. I also thank restaurants e.g. Le bistro d'edgard, Kakdougui, etc., often offering me discounts. I hope I didn't miss names. If I did, please forgive me. I thank all the people who helped me during these years.

Let me once again thank my supervisors. I'm proud and honored to be Gianni's first PhD student, and the first PhD student supervised by Emi as the thesis director. Your warm after-work activity invitations, such as Friday dinners, the year-end movie organized by Gianni, summer jogging, weekend lunches and dinners, big birthday meals, etc., with Emi, are the wealth of my memory.

Thanks to my dear parents, Ping Yu and Fang Yan, who supported my decision to study abroad. You are always my role models. Even though I might not be able to understand all your life philosophy for the moment, I feel like I'm affected by you and building my own world based on your influence. This is a cool thing: I can feel something called inheritance.

I am grateful for the three years of PhD journey and the more than six years of living in France. This is the unique wealth in my life. Finally, due to the long distance, I have not been able to be present at the funeral of elderly family members in the past few years.
This paper is dedicated to my dear grandmothers.

Abstract

This work focuses on uncertainty quantification for deep neural networks, which is vital for reliability and accuracy in deep learning. However, complex network design and limited training data make estimating uncertainties challenging. Meanwhile, uncertainty quantification for regression tasks has received less attention than for classification ones due to the more straightforward standardized output of the latter and their high importance. However, regression problems are encountered in a wide range of applications in computer vision. Our main research direction is on post-hoc methods, and especially auxiliary networks, which are one of the most effective means of estimating the uncertainty of main task predictions without modifying the main task model. At the same time, the application scenario mainly focuses on visual regression tasks. In addition, we also provide an uncertainty quantification method based on the modified main task model and a dataset for evaluating the quality and robustness of uncertainty estimates.

We first propose Side Learning Uncertainty for Regression Problems (SLURP), a generic approach for regression uncertainty estimation via an auxiliary network that exploits the output and the intermediate representations generated by the main task model. This auxiliary network effectively captures prediction errors and competes with ensemble methods in pixel-wise regression tasks.

To be considered robust, an auxiliary uncertainty estimator must be capable of maintaining its performance and triggering higher uncertainties while encountering Out-of-Distribution (OOD) inputs, i.e., to provide robust aleatoric and epistemic uncertainty. We consider that SLURP is mainly adapted for aleatoric uncertainty estimates. Moreover, the robustness of the auxiliary uncertainty estimators has not been explored. Our second work presents a generalized auxiliary uncertainty estimator scheme, introducing the Laplace distribution for robust aleatoric uncertainty estimation and Discretization-Induced Dirichlet pOsterior (DIDO) for epistemic uncertainty. Extensive experiments confirm robustness in various tasks.

Furthermore, to introduce DIDO, we provide a survey paper on regression with discretization strategies, developing a post-hoc uncertainty quantification solution, dubbed Expectation of Distance (E-Dist), which outperforms the other post-hoc solutions under the same settings.

Additionally, we investigate single-pass uncertainty quantification methods, introducing Discriminant deterministic Uncertainty (LDU), which advances scalable deterministic uncertainty estimation and competes with Deep Ensembles on monocular depth estimation tasks.

In terms of uncertainty quantification evaluation, we offer the Multiple Uncertainty Autonomous Driving dataset (MUAD), supporting diverse computer vision tasks in varying urban scenarios with challenging out-of-distribution examples.

In summary, we contribute new solutions and benchmarks for deep learning uncertainty quantification, including SLURP, E-Dist, DIDO, and LDU. In addition, we propose the MUAD dataset to provide a more comprehensive evaluation of autonomous driving scenarios with different uncertainty sources.

Résumé

Ce travail se concentre sur la quantification de l'incertitude pour les réseaux de neurones profonds, qui est vitale pour la fiabilité et la précision de l'apprentissage profond. Cependant, la conception complexe du réseau et les données d'entrée limitées rendent difficile l'estimation des incertitudes. Parallèlement, la quantification de l'incertitude pour les tâches de régression a reçu moins d'attention que pour celles de classification en raison de la sortie standardisée plus simple de ces dernières et de leur grande importance. Cependant, des problèmes de régression sont rencontrés dans un large éventail d'applications en vision par ordinateur. Notre principal axe de recherche porte sur les méthodes post-hoc, et notamment les réseaux auxiliaires, qui constituent l'un des moyens les plus efficaces pour estimer l'incertitude des prédictions des tâches principales sans modifier le modèle de la tâche principale. Dans le même temps, le scénario d'application se concentre principalement sur les tâches de régression visuelle. En outre, nous fournissons également une méthode de quantification de l'incertitude basée sur le modèle modifié de tâche principale et un ensemble de données permettant d'évaluer la qualité et la robustesse des estimations de l'incertitude.

Nous proposons d'abord Side Learning Uncertainty for Regression Problems (SLURP), une approche générique pour l'estimation de l'incertitude de régression via un réseau auxiliaire qui exploite la sortie et les représentations intermédiaires générées par le modèle pour la tâche principale. Le réseau auxiliaire apprend l'erreur de prédiction du modèle pour la tâche principale et peut fournir des estimations d'incertitude comparables à celles des approches des ensembles pour différentes tâches de régression par pixel.

Pour être considéré comme robuste, un estimateur d'incertitude auxiliaire doit être capable de maintenir ses performances et de déclencher des incertitudes plus élevées tout en rencontrant des entrées des examples Out-Of-Distribution (OOD), c'est-à-dire de fournir une incertitude aléatoire et épistémique robuste. Nous considérons que SLURP est principalement adapté aux estimations de l'incertitude aléatoires. De plus, la robustesse des estimateurs auxiliaires d'incertitude n'a pas été explorée. Notre deuxième travail propose un schéma d'estimateur d'incertitude auxiliaire généralisé, introduisant la distribution de Laplace pour l'estimation aléatoire robuste de l'incertitude et le Discretization-Induced Dirichlet pOsterior (DIDO) pour l'incertitude épistémique. Des expériences approfondies confirment la robustesse dans diverses tâches.

De plus, pour présenter DIDO, nous présentons un article d'évaluation des solutions qui appliquent des stratégies de discrétisation aux tâches de régression, développant une solution de quantification d'incertitude post-hoc, baptisée Expectation of Distance (E-Dist), qui surpasse les autres solutions post-hoc dans les mêmes conditions.

De plus, nous étudions les méthodes de quantification de l'incertitude en un seul passage basées sur le modèle de tâche principale ajusté. Nous proposons Latent Discreminant deterministic Uncertainty (LDU), qui fait progresser l'estimation déterministe de l'incertitude évolutive et rivalise avec les Deep Ensembles sur les tâches d'estimation de profondeur monoculaire.

En termes d'évaluation de la quantification de l'incertitude, nous proposons un ensemble de données Multiple Uncertainty Autonomous Driving (MUAD), prenant en charge diverses tâches de vision par ordinateur dans différents scénarios urbains avec des différents exemples OOD difficiles.

En résumé, nous proposons de nouvelles solutions et références pour la quantification de l'incertitude de l'apprentissage profond, notamment SLURP, E-Dist, DIDO et LDU. De plus, nous proposons l'ensemble de données MUAD pour fournir une évaluation plus complète des scénarios de conduite autonome avec différentes sources d'incertitude.

Résumé étendu en français

Ce travail se concentre sur la quantification de l'incertitude pour les réseaux de neurones profonds. Il est devenu essentiel pour les algorithmes d'apprentissage profond de quantifier leurs incertitudes de sortie afin de satisfaire aux contraintes de fiabilité et de fournir des résultats précis. Cependant, les réseaux de neurones profonds sont conçus de manière complexe et ont été entrainés dans la pratique sur des ensembles de donnés finis. Cela rend difficile l'obtention d'estimations d'incertitude pour ces modèles entraînés. Parallèlement, la quantification de l'incertitude pour les tâches de régression a reçu moins d'attention que celles de classification en raison de la sortie standardisée plus simple de ces dernières et de leur grande importance. Cependant, des problèmes de régression sont rencontrés dans un large éventail d'applications en vision par ordinateur. Notre principal axe de recherche porte sur les méthodes post-hoc, et notamment les réseaux auxiliaires, qui constituent l'un des moyens les plus efficaces pour estimer l'incertitude des prédictions des tâches principales sans modifier le modèle de la tâche principale. Dans le même temps, le scénario d'application se concentre principalement sur les tâches de régression visuelle. En outre, nous fournissons également une méthode de quantification de l'incertitude basée sur le modèle modifié de tâche principale et un ensemble de données permettant d'évaluer la qualité et la robustesse des estimations de l'incertitude.

Nous proposons d'abord SLURP, une approche générique pour l'estimation de l'incertitude de régression via un réseau auxiliaire qui exploite la sortie et les représentations intermédiaires générées par le modèle pour la tâche principale. Le réseau auxiliaire apprend l'erreur de prédiction du modèle pour la tâche principale et peut fournir des estimations d'incertitude comparables á celles des approches des ensembles pour différentes tâches de régression par pixel.

Pour être considéré comme robuste, un estimateur d'incertitude auxiliaire doit être capable de maintenir ses performances et de déclencher des incertitudes plus élevées tout en rencontrant des entrées des examples Out-Of-Distribution (OOD), c'est-à-dire de fournir une incertitude aléatoire et épistémique robuste. Nous considérons que le design de SLURP est principalement adapté aux estimations de l'incertitude aléatoires. De plus, la robustesse des estimateurs auxiliaires d'incertitude n'a pas été explorée. Dans le deuxième travail, nous proposons un schéma d'estimateur d'incertitude auxiliaire généralisé pour une quantification plus robuste de l'incertitude sur les tâches de régression. Concrètement, pour obtenir une estimation aléatoire plus robuste de l'incertitude, différentes hypothèses de distribution sont considérées pour le bruit hétéroscédastique et la distribution de Laplace est finalement choisie pour approximer l'erreur de prédiction. Pour l'incertitude épistémique, nous proposons une nouvelle solution nommée Discretization-Induced Dirichlet pOsterior (DIDO), qui modélise le Dirichlet postérieur sur l'erreur de prédiction discrétisée. Des expériences approfondies sur l'estimation de l'âge, l'estimation de la profondeur monoculaire et les tâches de super-résolution montrent que notre méthode proposée peut fournir des estimations d'incertitude robustes face à des entrées bruyantes et qu'elle peut être évolutive à la fois pour des tâches au niveau de l'image et au niveau des pixels.

De plus, pour présenter DIDO, nous présentons un article d'enquête sur les solutions qui appliquent des stratégies de discrétisation aux tâches de régression en passant en revue les travaux d'estimation de profondeur monoculaire utilisant des approches de classification sur la régression. Parallèlement, nous développons une solution de quantification d'incertitude post-hoc basée sur les modèles de ré-

gression basés sur la discrétisation, baptisée Expectation of Distance (E-Dist), qui surpasse les autres solutions post-hoc en utilisant une propagation unique vers l'avant.

En plus des solutions post-hoc, nous étudions également les méthodes de quantification de l'incertitude en un seul passage basées sur le modèle de tâche principale ajusté. Nous proposons l'incertitude de Latent Discreminant deterministic Uncertainty (LDU), qui propose des méthodes d'incertitude déterministe évolutives et efficaces qui assouplissent la contrainte de Lipschitz, ce qui entrave généralement le caractère pratique de telles architectures. Nous apprenons un espace latent discriminant en exploitant une couche de maximisation de distinction sur un ensemble de prototypes entraînables de taille arbitraire. Notre approche permet d'obtenir des résultats compétitifs par rapport aux Deep Ensembles, l'état de l'art en matière de prédiction de l'incertitude, sur les tâches d'estimation de profondeur monoculaire.

En termes d'évaluation de la quantification de l'incertitude, nous fournissons à la communauté un nouvel ensemble de données synthétiques, l'ensemble de données de Multiple Uncertainty Autonomous Driving (MUAD). L'ensemble de données MUAD prend en charge différentes tâches de vision par ordinateur, telles que les tâches de segmentation sémantique et d'estimation de profondeur monoculaire, et contient divers scénarios urbains avec différentes conditions d'éclairage et météorologiques ainsi que différents exemples OOD difficiles sur les ensembles de tests pour une évaluation équitable de la qualité de la quantification de l'incertitude et de sa robustesse.

En résumé, nous proposons de nouvelles solutions et critères d'évaluation pour la tâche de quantification de l'incertitude de l'apprentissage profond. Les solutions se concentrent principalement sur les méthodes post-hoc, et nous avons successivement proposé SLURP, E-Dist et DIDO. De plus, nous avons également proposé la méthode déterministe d'estimation de l'incertitude LDU basée sur la modification du modèle de tâche principale. En termes de références d'évaluation, nous proposons l'ensemble de données MUAD pour fournir une évaluation plus complète des scénarios de conduite autonome avec différentes sources d'incertitude.

Importance of uncertainty quantification in deep learning

In the deep learning era, models based on deep learning techniques [127] have become integral to numerous domains and applications, including the development of breakthrough technologies, such as GPT-4 [183], AlphaGo [214], AlphaFold [111], and advanced video generation [58]. With the remarkable progress witnessed in these areas, deep learning is reshaping the landscape of AI and world development. Within this transformative shift, computer vision stands as a vital domain since vision is the most widely recognized and the most widely studied perceptual modality [218]. Driven by the capabilities of deep learning models, computer vision has made substantial strides in replicating and augmenting human visual perception, being applied to diverse fields like healthcare [144] and autonomous vehicles [20].

However, as these AI models become increasingly sophisticated, it becomes crucial to address not only their main task predictive ability but also their reliability. This brings us to the aspect of model uncertainty quantification [73]. In this thesis, we dive into novel approaches for estimating uncertainty for deep learning models, aiming to meet the increasing need for uncertainty quantification in deep learning. This endeavor has the potential to enhance the precision, dependability, and trustworthiness of AI applications significantly.

Specifically, uncertainty quantification can be applied in some deep learning-based systems. For example, in active learning [210], uncertainty quantification can act as an acquisition function to provide the annotation system with the most uncertain samples of the current model for manual labeling. More importantly, uncertainty quantification is an important supplement to the outputs of the deterministic models, providing valuable decision-making references for human users and making the models more credible.

The uncertainty quantification in deep learning is mostly based on the theory of probabilistic machine learning [174]. However, compared with the rapidly expanding application scenarios and high-performance models with increasing parameters, the application of probabilistic machine learning methods in deep learning is still a process of practice and exploration. When facing more complex models and some pixel-wise tasks, the scalability of uncertainty quantification methods also has to be taken into account.

The main goal of this thesis is to introduce and develop such scalable and practical tools to reason about the uncertainty in deep learning. In the rest of the chapter, I will introduce some of the applications that involve deep learning and uncertainty quantification as one of the most important roles.

Uncertainty quantification often lacks direct and explicit application in computer vision applications. Take image classification or image semantic segmentation, for instance. Typically, users are only aware of the semantic information associated with the input picture, without knowledge of the model's uncertainty level regarding the decision made. However, uncertainty plays a crucial role in deep learning models, as each decision made by the model inherently involves a degree of uncertainty. This uncertainty is as significant as the decision itself and should be provided to the user to aid in making final decisions.

Moreover, uncertainty is not only valuable for users but also crucial on the unseen side, involving model training and data annotation. In this section, we will briefly explore various specific applications where uncertainty becomes relevant.

Active learning

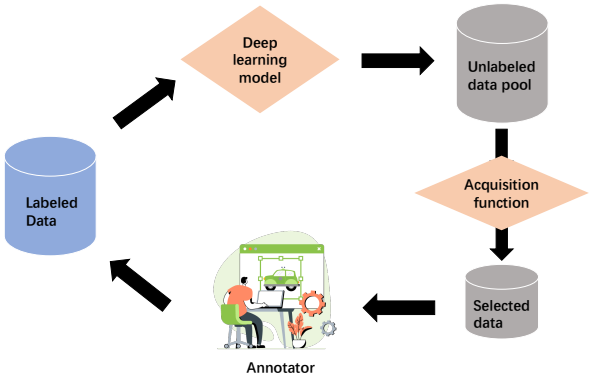


Figure 1: Demonstration of pool-based active learning [135, 210].

Active learning is a powerful method in machine learning that aims to improve the performance of models by actively selecting the most informative samples for training. In deep learning, large labeled datasets are required to train models effectively. However, labeling data can be expensive and time-consuming, especially when dealing with complex tasks or rare categories, such as lesion labeling that requires expert knowledge, emotional fluctuations labeling based on human speech, etc. Active learning selects the instances for the annotators according to the performance of the in-system model, thus making the best use of limited labeled data. At the same time, active learning can also make the model fit faster and achieve similar performance as when training with more labeled data [11]. Figure 1 shows the pool-based active learning procedure [135, 210]. The deep learning model will go through the unlabeled data pool and provide the necessary information on the unlabeled data, e.g., uncertainty estimates, for the acquisition function. The acquisition function will select the data for the annotator to label according to the provided information.

One of the cores of active learning is the acquisition function, which is a key component that drives the information sample selection process. It evaluates the potential of each unlabeled instance to reduce model uncertainty or improve its performance when labeled and included in the training set. Usually, the acquisition function quantifies the uncertainty of the unmarked points, obtains the most uncertain sample set of the currently trained model that the acquisition function considers, and submits it to the annotator for labeling.

A variety of acquisition functions are used in active learning. One of the most commonly used acquisition functions is the estimation of the uncertainty of a single model. It involves selecting instances where the model's predictions are uncertain or have high variance. For example, the model may be uncertain about the class assignment of some data points near the decision boundary. More specifically, we can use some different uncertainty estimation measures, such as entropy [211], energy [147], etc., to rank the uncertainty. By selecting the most ambiguous samples for labeling, the model can better understand the need to improve performance.

Another popular acquisition function is the query-by-committee approach. It involves training multiple models, often referred to as a "committee" or an "ensemble," with different initializations or architectures. The disagreement among the committee members on the labeling of a particular instance indicates uncertainty and, consequently, the sample's potential informativeness. This is very similar to ensemble methods [123, 261]. Instances that caused significant disagreement among committee members were considered informative and prioritized for flagging.

Uncertainty quantification in autonomous driving and healthcare

In the realm of robotics and autonomous navigation, the machine needs to perceive the environment, make decisions, interact with humans, etc. The real world is full of uncertainty, for example, dynamic environments, noise from sensors, and unforeseen obstacles that have never been seen before. Uncertainty quantification emerges as a cornerstone of enhancing the robustness and reliability of autonomous systems in the uncertain real world.

In safe navigation and autonomous driving, machines may encounter unpredictable environmental changes, such as changing weather conditions, sudden appeared objects, changes in terrain, etc. Uncertainty estimation combined with deep learning can continuously evaluate the predictions given by the machine, which allows the machine to adjust its decisions and behaviors in a timely manner, such as slowing down and changing routes [22, 102].

In the field of healthcare, where risks are often a matter of life and death, the combination of deep learning and uncertainty quantification provides many advantages in aspects such as diagnostic decision-making. In the process of using deep learning to quickly analyze and diagnose MRI images, uncertainty quantification can provide a confidence interval that quantifies the likelihood of malignant lesions. Doctors can develop treatment plans based on confidence levels. This method can provide convenience for more patients in remote areas that lack good medical conditions and save money and time. At the same time, the confidence of the diagnosis at each stage of the treatment can be effectively communicated to the patient, thereby ensuring that patients fully understand their current physical condition. Such transparency not only allows patients to better understand their condition, but also reduces the distrust between doctors and patients, reduces the occurrence of disputes, and improves the employment environment for doctors.

There are many more cases where uncertainty quantification is combined with deep learning, for example, learning with imbalanced classes [114, 137] and noisy labels [101, 241]. In short, as an indispensable part of deploying deep learning models, the study of uncertainty estimation is crucial. This thesis will carry out some research, experiments, and discussions around uncertainty quantification, hoping to provide some insights and inspiration for the field of uncertainty estimation.

The problem being addressed in the thesis

This thesis will primarily address uncertainty quantification and aim to devise a method for estimating model output uncertainty without necessitating alterations to the model's architecture and parameters or only making minor structural adjustments. Given that deep neural networks (DNNs) are meticulously tailored for specific tasks, and these networks inherently lack an inherent mechanism for uncertainty estimation, this is particularly relevant for regression tasks where model outputs often lack the necessary information for making reliable judgments. Estimating uncertainty for the model's outputs without introducing substantial modifications to the network is a prudent and efficient approach since it preserves the accuracy and robustness of the model. Concurrently, this thesis also delves into the underexplored domain of uncertainty stemming from distribution disparities between inference and training data in regression tasks and tries to address the challenge of identifying out-of-distribution samples and providing uncertainty estimates in the context of regression problems.

Outline of work

In this thesis, we deal with the uncertainty quantification problem for deep learning models, especially for the ones targeted to regression tasks. We aim to provide more feasible and reliable uncertainty quantification solutions for the deployed or under-deploying deep learning models.

The main parts of the thesis can be summarized as follows:

1. **State-of-the-art uncertainty quantification solutions in deep learning and its evaluation (Part I)**. We study the fundamental problem of uncertainty quantification in deep learning, including the state-of-the-art solutions in the field and the evaluation strategies. The goal is to find inspiration and perspectives for improving the existing solutions and deriving our proposed solutions. At the same time, while introducing the evaluation metrics and the datasets, we propose a new dataset in Chapter 2. This new autonomous driving dataset contains multiple uncertainties, including the different weather conditions (e.g., rain, fog, and snow), lighting conditions (daytime and nighttime), and semantically out-of-distribution objects. This new dataset provides richer comparison possibilities for uncertainty quantification of both classification and regression problems.

2. **Error prediction in pixel-wise regression tasks using an auxiliary network (Part II)**. We deal with uncertainty quantification for regression problems using the auxiliary network. Applying an auxiliary network to the main task model is a kind of post-hoc solution for uncertainty quantification. We propose to train the auxiliary by learning the total uncertainty, i.e., the prediction error, provided by the main task model. The design of the auxiliary network is novel. Specifically, we use both the input and output of the main task model as the inputs to the auxiliary model. For the model architecture, we use the rich features from the image encoder to build a coarse-to-fine decoder with multiple atrous convolutions. Our experiment shows that the proposed auxiliary network is able to provide comparable uncertainty estimates to state-of-the-art solutions.

3. **Introducing classification into uncertainty quantification problems in regression (Part III)**. In this part, we manage to introduce classification strategies into the regression tasks for uncertainty quantification and auxiliary networks. In the first chapter of this part, we present a survey on applying classification approaches to regression for both the main task and the uncertainty estimation. Focusing on the task of monocular depth estimation, we summarize and discuss all solutions that have reused classification methods in this task so far. At the same time, we propose a new post-hoc uncertainty quantification method for this kind of model. Experiments have shown that this method, without modifying the model and using only one forward propagation, has better uncertainty estimation quality than other methods under the same conditions. In the next step, we introduce the classification approaches to auxiliary network learning. We are the first to apply the discretization to the prediction error and model the distributional uncertainty split from predictive uncertainty by applying the Dirichlet distribution to the discretized prediction error. Our solution is scalable and able to be applied to both image-level and pixel-wise vision tasks.

4. **Single-forward-propagation uncertainty quantification methods (Part IV)**. Apart from the auxiliary networks, we also deal with the robust uncertainty quantification by modifying the main task model. We introduce a distinction maximization layer into the penultimate layer of the main task model to provide bi-lipschitz features to the final classification and uncertainty quantification layers. With a single-forward propagation, the uncertainty estimator can provide more robust out-of-distribution example detection results on semantic segmentation tasks and image classification tasks. This method is scalable to the regression tasks. We observe an improved main task prediction performance and a comparable uncertainty quantification quality compared to the benchmark methods.

Publications

The publications during this PhD work are listed as follows:

- **SLURP: Side Learning Uncertainty for Regression Problems [252]** (see Chapter 3)
  Xuanlong Yu, Gianni Franchi, Emanuel Aldea
  British Machine Vision Conference (BMVC), 2021

- **On Monocular Depth Estimation and Uncertainty Quantification using Classification Approaches for Regression [253]** (see Chapter 4)
  Xuanlong Yu, Gianni Franchi, Emanuel Aldea
  IEEE International Conference on Image Processing (ICIP), 2022 (Oral)
- **Latent Discriminant deterministic Uncertainty [66]** (see Chapter 6)
  Gianni Franchi*, Xuanlong Yu*, Andrei Bursuc, Emanuel Aldea, Séverine Dubuisson, David Filliat
  * benchmark design for supervised MDE task, loss design, writing, editing
  European Conference on Computer Vision (ECCV), 2022
- **MUAD: Multiple Uncertainties for Autonomous Driving, a benchmark for multiple uncertainty types and tasks [67]** (see Chapter 2)
  Gianni Franchi*, Xuanlong Yu*, Andrei Bursuc, Angel Tena, Rémi Kazmierczak, Séverine Dubuisson, Emanuel Aldea, David Filliat
  * benchmark design for supervised/unsupervised MDE task, writing, editing, website creation and maintenance
  British Machine Vision Conference (BMVC), 2022
- **The Robust Semantic Segmentation UNCV2023 Challenge Results [255]**
  Xuanlong Yu, et al.
  International Conference on Computer Vision Workshop (ICCVW), 2023
- **Discretization-Induced Dirichlet Posterior for Robust Uncertainty Quantification on Regression [254]** (see Chapter 5)
  Xuanlong Yu, Gianni Franchi, Jindong Gu, Emanuel Aldea
  The 38th Annual AAAI Conference on Artificial Intelligence (AAAI), 2024

*equal contribution

Contents

List of Figures

List of Tables

Glossary

AUPR - *area under the precision-recall curve*

AUROC - *area under the receiver operating characteristic*

AUSE - *area under the sparsification error curve*

AuxUE - *auxiliary uncertainty estimator*

BNN - *Bayesian deep neural network*

CAR - *classification approaches for regression*

DEns. - *deep ensembles*

DNN - *deep neural network*

DUM - *deterministic uncertainty methods*

EPE - *end-point error*

FPR95 - *false positive rate at 95% recall*

ID - *in-distribution*

MC - *monte-carlo*

MDE - *monocular depth estimation*

MLP - *multi-layer perceptron*

NLL - *negative log-likelihood*

OOD - *out-of-distribution*

SR - *super-resolution task*

Part I

Background

Chapter 1

Approaches in uncertainty quantification in deep learning

Contents

Deep learning has emerged as a powerful and versatile solution for addressing complex problems across diverse domains, spanning image classification to natural language processing. However, a notable hurdle in deep learning models lies in their ability to capture and quantify uncertainty in their predictions effectively. The accurate estimation of uncertainty is essential in applications that rely on reliable uncertainty estimates, such as medical diagnosis or autonomous systems, as it has been mentioned in the previous chapter. This chapter aims to comprehensively explore and analyze various approaches to uncertainty quantification within the realm of deep learning. Specifically, we will delve into three distinct categories of approaches: distribution learning on the output space, ensembles and sampling-based methods, and post-hoc techniques. Each of these approaches will illuminate unique techniques and methodologies employed to tackle the formidable challenge of quantifying uncertainty in deep learning models. By gaining a profound understanding of these approaches and comparing their strengths, limitations, and potential applications, we can pave the way for improved decision-making and enhance the robustness of deep learning systems.

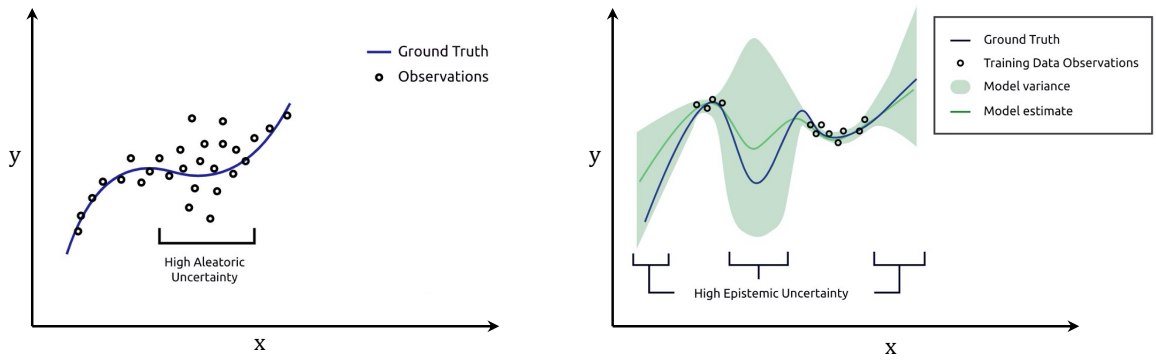


Figure 1.1: Visualization of two types of uncertainty on 1D regression tasks. Left: Visualizations on high aleatoric uncertainty. Right: Visualizations on high epistemic uncertainty.

1.1 Uncertainty modeling on Deep Neural Networks

1.1.1 Preliminary

We define a training dataset $\mathscr{D} = \{\mathbf{x}_i, y_i\}_i^{\mathrm{N}}$ where N is the number of samples/images depending on the task at hand. We consider that $\mathbf{x}, \mathbf{y}$ are drawn from a joint distribution $\mathrm{P}(\mathbf{x}, \mathbf{y})$. We denote a DNN $f_{\boldsymbol{\omega}}$ with parameters $\boldsymbol{\omega}$ applied to fit the samples to the corresponding labels in the dataset $\mathscr{D}$.

1.1.2 Predictive uncertainty and its decomposition

Most of the distribution-related solutions for DNN uncertainty quantification [5, 33, 34, 70, 108, 113, 123, 156, 157, 180, 209] start from the predictive uncertainty in the Bayesian framework. Given a new input $\mathbf{x}^*$, consider the corresponding ground truth is y^*, the predictive uncertainty of $f_{\boldsymbol{\omega}}$ will be described by $\mathrm{P}(y^*|\mathbf{x}^*, \mathscr{D})$ and it can be decomposed as aleatoric uncertainty and epistemic uncertainty:

$$\mathrm{P}(y^*|\mathbf{x}^*, \mathscr{D}) = \int \mathrm{P}(y^*|\mathbf{x}^*, \boldsymbol{\omega}) \mathrm{P}(\boldsymbol{\omega}|\mathscr{D}) d\boldsymbol{\omega} \tag{1.1}$$

The $\mathrm{P}(y^*|\mathbf{x}^*, \boldsymbol{\omega})$ is the likelihood of the prediction target given the input and the model. It is about how probable this prediction is compared to the ground truth, and normally, we need to assume the predictions follow a certain type of distribution. When we only talk about $\mathrm{P}(y^*|\mathbf{x}^*, \boldsymbol{\omega})$, the model parameters can be regarded as fixed, the probability here is then interpreted as the data uncertainty or aleatoric uncertainty. The $\mathrm{P}(\boldsymbol{\omega}|\mathscr{D})$ is a probability only about model parameters given the dataset. It describes model uncertainty or epistemic uncertainty, which represents the level at which the model learns the dataset. Figure 1.1 illustrates two types of uncertainty on a 1D regression task.

Based on the above formula, many works have been proposed providing solutions to measure these two types of uncertainties. In the following sections, we group these schemes into distributional learning approaches [5, 33, 34, 108, 113, 156, 157, 180, 209, 228], sampling and ensembling approaches [70, 123, 167, 237], and post-hoc approaches [40, 41, 92, 197, 229, 252]. Among them, the distribution learning and the post-hoc methods are mainly oriented to the estimation of aleatoric uncertainty, and the sampling and ensembling methods involve epistemic uncertainty estimation. In particular, post-hoc methods are mainly aimed at DNNs, which can estimate the aleatoric uncertainty of DNN predictions without updating model parameters and adjusting DNN structure. One of the themes of the thesis is post-hoc methods and their applications in regression tasks.

1.2 Distribution learning on the output space

As we introduced in the previous section, the prediction is the most likely result given an output distribution. Yet, in practice, when we use DNNs, we often choose point estimates as the final decision

estimates, especially on regression tasks. So if we only focus on the output of the model, we tend to ignore this important distribution. In short, under the Bayes framework, the prediction given by the DNN is a distribution, and we choose the most likely value from this distribution as the final result. Given the model parameters $\boldsymbol{\omega}$, or more ideally, assuming that $\boldsymbol{\omega}$ are perfect parameters given by an ideal oracle, the distribution still exists and will not (always) be a Dirac. This is due to aleatoric uncertainty, i.e., due to the uncertain component of the given observations acquired in the real world under different conditions, the results will contain the noise.

Therefore, we consider both network prediction and the noise in the forecast as the main factor that makes up the outcome distribution $\mathrm{P}(y^*|\mathbf{x}^*, \boldsymbol{\omega})$. For regression tasks, a widely used distribution is the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. Specifically, μ corresponds to the DNN point estimation. The noise is a zero mean Gaussian random variable with the variance denoted as σ^2. When σ^2 is the same for all the input, we call the noise homoscedastic. For different inputs, if the variance of the noise distribution changes, we call the noise heteroscedastic. Back in 1994, Nix and Weigend [180] introduced heteroscedastic noise prediction into artificial neural networks and 2D data regression tasks. Recently, Kendall and Gal [113] adapted the principle to the DNNs and more complex computer vision tasks, which turns to be a strong baseline in output space distribution learning.

1.2.1 Gaussian distribution learning on the output space

Training a DNN with parameters $\boldsymbol{\omega}$ involves maximizing the likelihood $\mathrm{P}(y|\mathbf{x}, \boldsymbol{\omega})$ given the dataset $\mathscr{D}$. When the above-mentioned Gaussian assumption is applied, we can achieve negative Gaussian likelihood minimization:

$$\begin{aligned}
\hat{\boldsymbol{\omega}} = \underset{\boldsymbol{\omega}}{\operatorname{argmax}} \mathrm{P}(\mathscr{D}|\boldsymbol{\omega}) &= \underset{\boldsymbol{\omega}}{\operatorname{argmax}} \sum_{i=1}^{\mathrm{N}} \log(\mathrm{P}(y_i|x_i, \boldsymbol{\omega})) \\
&= \underset{\boldsymbol{\omega}}{\operatorname{argmax}} \sum_{i=1}^{\mathrm{N}} \log(\mathcal{N}(y_i|\mu_i, \sigma_i^2)) \\
&= \underset{\boldsymbol{\omega}}{\operatorname{argmax}} \sum_{i=1}^{\mathrm{N}} -(\frac{1}{2}\log 2\pi\sigma_i^2 + \frac{(y_i - \mu_i)^2}{2\sigma^2}) \\
&= \underset{\boldsymbol{\omega}}{\operatorname{argmin}} \sum_{i=1}^{\mathrm{N}} \frac{1}{2}\log 2\pi\sigma_i^2 + \frac{(y_i - \mu_i)^2}{2\sigma_i^2}
\end{aligned} \tag{1.2}$$

If only homoscedastic noise is considered, i.e., the same variance σ_{const}^2 construct the noise distributions, we have:

$$\hat{\boldsymbol{\omega}} = \underset{\boldsymbol{\omega}}{\operatorname{argmin}} \sum_{i=1}^{\mathrm{N}} \frac{1}{2}\log 2\pi\sigma_{\text{const}}^2 + \frac{(\boldsymbol{x}_i - \mu_i)^2}{2\sigma_{\text{const}}^2} \tag{1.3}$$

The first thing based on the above equations, is the DNN architecture design, especially the output space. The μ and σ^2 are generated by the DNN with parameters $\hat{\boldsymbol{\omega}}$, which is related to the architecture of the DNN. Since μ solves the main task, the output space will be adjusted according to different settings on σ^2:

1. We ignore the σ^2 and keep the DNN output only the μ. This is equivalent to taking the σ^2 as σ_{const}^2 and without estimating them. Thus, based on Eq. 1.3, we can obtain the well-known mean square error (MSE) loss for regression tasks [17].

2. We take the noise as homoscedastic, and estimate both μ and σ_{const}^2. The DNN output space will be adjusted to additionally output a trainable parameter, which represents σ_{const}^2 and is not affected by the previous features but only trained by the Eq. 1.3.

3. We take the noise as heteroscedastic, and estimate μ and σ based on Eq. 1.2. Different from estimating σ_{const}^2, the estimation of σ^2 here is more similar to the μ estimation. Normally, in

practice, the feature map on the penultimate layer (or the pre-logit layer) will be chunked into two maps with equivalent shapes according to the feature channel. Some implementations will also duplicate these feature maps. There is no exact optimal implementation for this dual-task architecture design, yet they all follow the Eq. 1.2 principle. These two feature maps will then be the inputs of the main task and the uncertainty estimator on top of the DNN.

The third design direction follows Nix and Weigend [180] and is applied widely in the following works such as in Kendall and Gal [113], lightweight probabilistic DNN [72], mono-uncertainty [191], evidential regression [5] and so on. It also leads to and influences the creation and design of the auxiliary networks for uncertainty quantification. We will go into the details of the auxiliary networks in Section 1.4.

It is easy to see from Eq. 1.2 that the optimal σ^2 is the prediction error given the fixed main task prediction μ. We can thus consider the other symmetric distributions, such as the Laplace distribution in lightweight probabilistic DNN [72], which has a similar property, to replace the Gaussian distribution in the assumption. In practice, different distribution assumptions might cause different aleatoric uncertainty quantification performance. We investigate this phenomenon in the Chapter III.5.

1.2.2 Prior/Posterior distribution learning on the output space

In the previous section, we model the output of the DNN as a Gaussian distribution, which originally comes from the likelihood in Eq. 1.1. Furthermore, the Eq. 1.1 can be further decomposed. By introducing the prior distribution $\mathrm{P}(\boldsymbol{\alpha})$ of the likelihood, we can disentangle $\mathrm{P}(y^*|\mathbf{x}^*, \boldsymbol{\omega})$ to two parts:

$$\begin{aligned}\mathrm{P}(y^*|\mathbf{x}^*, \mathscr{D}) &= \int \mathrm{P}(y^*|\mathbf{x}^*, \boldsymbol{\omega})\mathrm{P}(\boldsymbol{\omega}|\mathscr{D})d\boldsymbol{\omega} \\ &= \iint \mathrm{P}(y^*|\boldsymbol{\alpha})\mathrm{P}(\boldsymbol{\alpha}|\mathbf{x}^*, \boldsymbol{\omega})\mathrm{P}(\boldsymbol{\omega}|\mathscr{D})d\boldsymbol{\omega}d\boldsymbol{\alpha}\end{aligned} \tag{1.4}$$

where $\mathrm{P}(\boldsymbol{\alpha}|\mathbf{x}^*, \boldsymbol{\omega})$ can describe the distributional uncertainty [157]. More specifically, the parameters of the prior distribution can be updated during model training and become the posterior given the dataset and the model parameters. The final distribution of the prediction target y^* is drawn from its posterior distribution. The diversity of the posterior distribution will show how the final distribution is distributed. Note that, the previous-defined aleatoric uncertainty is disentangled, we call $\mathrm{P}(y^*|\boldsymbol{\alpha})$ the new data or aleatoric uncertainty, and the $\mathrm{P}(\boldsymbol{\alpha}|\mathbf{x}^*, \boldsymbol{\omega})$ distributional uncertainty, according to the literature such as Malinin and Gal [157].

This 'distribution of distribution' modeling is also called evidential deep learning [228], which is one of the single-pass uncertainty quantification solutions. There are other single-pass uncertainty quantification methods, such as geometry-based or prototype-based discriminative uncertainty methods (DUM) [194]. We will introduce them in the last section of this chapter, and also provide a novel solution in Chapter IV.6.

The reason why it is also called evidential deep learning is mainly according to Subjective logic [110]. The author Jøsang considers that subjective logic provides a principled way to connect beliefs to Dirichlet distributions, and Dirichlet distribution is equivalent to a multinomial opinion. Beliefs are the expressed subjective opinions on the truth with certain degrees of uncertainty. In Dempster-Shafer Theory (DST) [47], the authors offered a systematic way to combine and update evidence to form beliefs in situations where uncertainty and incomplete information are prevalent. Denoeux [49] first uses DST to combine the evidence provided by a radial basis function layer [192] and applies it to the image classification task using a shallow artificial neural network. Recently, Jøsang [110] introduced Dirichlet distribution to multinomial belief learning. Since DST is also called evidence theory or the theory of belief functions, and the first ones who applied this theory to DNNs named the technique Evidential classification [209], such methods of using prior distributions of the distribution over the final prediction in DNNs are sometimes loosely referred to as evidential deep learning. Different from the works applying DST on the top of the model using DS layers and utility layers [224], the

evidential learning presented in this work mainly discusses the modeling of the output space, such as the conjugate prior distribution of the assumed distribution of the training data.

Dirichlet distribution is the prior distribution of the categorical distribution. Yet in the regression tasks, since we make Gaussian assumption on the final outputs, we can find a proper prior of Gaussian distribution to model the distributional uncertainty of the output, such as the Normal Inverse-Gamma distribution proposed in Evidential regression [5].

In the regression tasks, since we make Gaussian assumption on the final outputs, we can find a proper prior of Gaussian distribution to model the distributional uncertainty of the output, such as the Normal Inverse-Gamma distribution proposed in Evidential regression [5] and Normal-Wishart distribution in Regression Prior Networks [156]. The former is used for modeling univariate Gaussian distributions, and the latter is used for modeling multivariate Gaussian distributions. We will mainly introduce Normal Inverse-Gamma distribution in the following paragraphs, yet the principles are the same using different prior distributions.

We consider that given an input image, the DNN will output multiple Gaussian distributions, and these Gaussian distributions are sampled from a distribution over them, which is a Normal Inverse-Gamma (NIG) distribution $\mathrm{NIG}(\gamma, \nu, \alpha, \beta)$, where the mean of the Gaussian distributions follows a Gaussian distribution $\mathcal{N}(\gamma, \sigma^2\nu^{-1})$ and the variance follows an Inverse-Gamma distribution $\Gamma^{-1}(\alpha, \beta)$. The goal is to obtain the posterior distribution $\mathrm{P}(\mu, \sigma^2)$. Follow the Evidential regression [5] and Statistical field theory [185], the approximation can be obtained as $\mathrm{P}(\mu, \sigma^2) = \mathrm{P}(\mu)\mathrm{P}(\sigma^2)$. Then, this approximation can take the Normal Inverse-Gamma distribution, which is the conjugate prior of the Gaussian distribution:

$$\mathrm{P}(\mu, \sigma^2|\gamma, \nu, \alpha, \beta) = \frac{\beta^\alpha\sqrt{\nu}}{\Gamma(\alpha)\sqrt{2\pi\sigma^2}}\left(\frac{1}{\sigma^2}\right)^{\alpha+1}\exp\left\{-\frac{2\beta + \nu(\gamma-\mu)^2}{2\sigma^2}\right\} \tag{1.5}$$

The conjugate prior distribution can provide 'virtual- observations' to interpret the parameters [168], which is similar to the Energy [147] in classification DNNs. For NIG distribution, Amini et al. [5] define the total evidence Φ as the sum of all inferred virtual-observations counts $\Phi = 2\nu + \alpha$.

To combine the virtual multiple Gaussian distributions sampled from the NIG distribution, we can use some statistics about μ and σ^2:

$$\mathbb{E}[\mu] = \gamma, \quad \mathbb{E}[\sigma^2] = \frac{\beta}{\alpha - 1}, \quad \mathrm{Var}[\mu] = \frac{\beta}{\nu(\alpha - 1)} \tag{1.6}$$

where the $\mathbb{E}[\mu]$ represents the main task prediction, $\mathbb{E}[\sigma^2]$ represents aleatoric uncertainty, and $\mathrm{Var}[\mu]$ represents distributional uncertainty. In the original paper [5], the authors dub the $\mathrm{Var}[\mu]$ as the epistemic uncertainty, yet we could also state that the uncertainty here comes from the scale of the posterior distribution over the final distribution, which should represent the distributional uncertainty. However, sometimes the two names are used interchangeably because the distributional uncertainty might also be interpreted as the uncertainty caused by the model not having learned data with a different distribution than the training data. At the same time, in previous works [70, 209], both kinds of uncertainties were proved to be feasible for out-of-distribution detection, which also led to the confusion of their names.

Figure 1.2 provides the visualizations of the desired behaviors of the regression result, where we assume that the final result is given by a set of potential outputs drawn from different Gaussian distributions. The distributions behind the potential outputs indicate different types of uncertainty that the final output has. When the final output has low uncertainty, the virtual Gaussian distributions are more concentrated and identical, with smaller variances for each distribution, as shown in the right sub-figure. When output has a big irreducible aleatoric uncertainty and low epistemic or distributional uncertainty, the virtual distributions should have more identical means and variances but the variances should be big for all the distributions. This is because all the distributions show similar average opinions on the final decision due to the low epistemic uncertainty, yet big randomness around

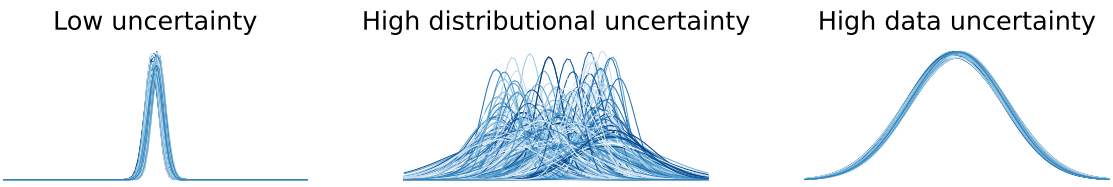


Figure 1.2: Visualizations on desired behaviors of regression results. Three different cases on the distribution of the Gaussian distributions for the output with different types of uncertainty.

this decision, as shown in the left sub-figure. The middle sub-figure shows the case that the high distributional uncertainty occurs in the final output. Facing input sampled from a different distribution to the ones of the training data, the virtual distributions given by the model should be more diverse and provide different opinions on the final decision, i.e., high Var[μ] across the virtual distributions, which is also shown by Amini et al. [5].

However, since the distributional uncertainty should consider the diversities between the virtual Gaussian distributions, both μ and σ^2 of each distribution should be considered. In Chapter III.5, we will introduce the observation that when the virtual Gaussian distributions share the same mean, the difference in the variances of virtual Gaussian distributions can also serve as a measure of distribution uncertainty.

In the next paragraphs, we follow Amini et al. [5] to describe the DNN training based on NIG prior assumption, especially the loss functions, and provide our insights on the training procedure.

The loss function consists of two parts. The design of the first one follows the likelihood function, yet we change the Gaussian parameters to the NIG parameters, and the likelihood of the training target turns to be $P(y^*|\gamma,\nu,\alpha,\beta)$. The DNN parameters $\boldsymbol{\omega}$ are placed in the NIG parameters. According to [5], in this case, the likelihood follows a Student-t (St) distribution:

$$P(y^*|\gamma,\nu,\alpha,\beta) = St\left(y^*;\gamma,\frac{\beta(1+\nu)}{\nu\alpha},2\alpha\right) \tag{1.7}$$

Then, we can follow the negative log-likelihood to build the first part of the loss function.

The second part is a regularization term, which regularizes the virtual counts according to the prediction error. The lower the prediction error is, the lower the regularization is attached to the virtual observations, and vice versa.

Then we have the loss function for training DNNs based on NIG assumption:

$$\mathscr{L}_1 = \frac{1}{2}\log\left(\frac{\pi}{\nu}\right) - \alpha\log(\Omega) + \left(\alpha+\frac{1}{2}\right)\log((y-\gamma)^2\nu+\Omega) + \log\left(\frac{\Gamma(\alpha)}{\Gamma(\alpha+\frac{1}{2})}\right) \tag{1.8}$$

$$\mathscr{L}_2 = |y-\gamma|\cdot(2\nu+\alpha) \tag{1.9}$$

$$\mathscr{L} = \mathscr{L}_1 + \lambda_{\text{NIG}}\cdot\mathscr{L}_2 \tag{1.10}$$

where $\Omega = 2\beta(1+\nu)$ and λ_{NIG} is the hyperparameter for the regularization. There are two following works on the reporting observations for the ineffectiveness of evidential regression and provided the adjustments. Oh and Shin [182] observed the gradient conflict when optimizing γ and $\{\nu,\alpha,\beta\}$ in the same time. Meinert et al. [164] observed that this solution offers only a heuristic approximation of epistemic uncertainty. Moreover, the evidential regression modeling is also over-parameterized, which makes the efficiency of the evidential regression seam almost unreasonable. Respectively, Oh and Shin [182] provide an evidential regularization during training, which uses a Lipschitz-modified MSE loss as the additional training objective to ensure that the uncertainty estimation of the negative log-likelihood is not disturbed by the main task training. On the other hand, Meinert et al. [164] propose to modify the evidential regression loss function. The new loss function prevents a too-fast

convergence and stabilizes the convergence for uncertainty-related parameters during optimization. In short, according to the following works, we can see that the design of the evidential regression loss function can not make the DNNs effectively converge, and the uncertainty estimates provided also do not fully convince the readers. We will not go through the details for these modifications, but we will provide some new points of view here and go into the details in Chapter 5.

As we mentioned before, the $\mathscr{L}_1$ in Eq. 1.8 is the negative log-likelihood (NLL) loss, which uses the Student-t assumption, rather than the Gaussian assumption. We argue that the main difference between this work and simple NLL is the penalization term $\mathscr{L}_2$ on the total evidence based on the prediction error. This idea is very similar to the design of Sensoy et al. [209] about penalizing the total evidence based on classification errors in evidence classification. However, one intuitive problem will arise in this design. When the structure of DNN becomes more and more refined and well-designed, the main task performance becomes much better than before. It results in lower prediction errors, which corresponds to the $|y-\gamma|$ term in Eq. 1.9. Lower prediction errors will cause the regularization to be hard to be effective since the current data samples are biased, and most of the samples have very small prediction errors.

The perturbation on the total evidence normally is a constraint on the combination of some parameters in the DNNs, which could represent the total evidence according to the evidential theory. The perturbation of the parameter magnitude in the classification will affect the final classification decision less, because the final decision is often to choose the category with the most evidence. At the same time, the total magnitude of evidence can provide information on the uncertainty from another perspective. In regression tasks, the regularization of the magnitude of the parameters or their combinations might hurt the main task predictions. This point of view is similar to Oh and Shin [182], while they modified the loss to avoid the main task performance reduction, which is a remedy in our opinion, but it is not guaranteed to work all the time.

For the uncertainty measure, the epistemic uncertainty or the distributional uncertainty is defined as the variance of the means of the virtual Gaussian distributions $\mathrm{Var}[\mu]$. However, the variance of the means alone lacks a concrete description of these Gaussian distributions, and we admit that capturing the variance of these distributions is difficult in practice because they are not truly sampled. We can only observe them through some statistics of their conjugate prior distribution. This point of view is similar to Meinert et al. [164], where the authors considered the epistemic uncertainty estimation to be a heuristic approximation. Some more recent works also show the drawbacks on using the prior/posterior distribution learning for epistemic uncertainty estimation, such as loss minimisation does not work for such second-order predictors [12], Dirichlet distribution is not robust to the adversarial attacks [119], and so on. These works are important supplements to evidential learning, and we can further improve the robustness and effectiveness of evidential learning-based models regarding to these works.

In conclusion, according to our observations and the following works [164, 182], we argue that the current evidential regression strategies on DNNs might cause the main task performance reduction, may not effectively constrain the total evidence on better and larger DNNs, and cannot provide a good definition of cognitive uncertainty. Based on these, we will introduce an alternative solution in Chapter 5, dubbed Discretization-Induced Dirichlet pOsterior (DIDO) [254].

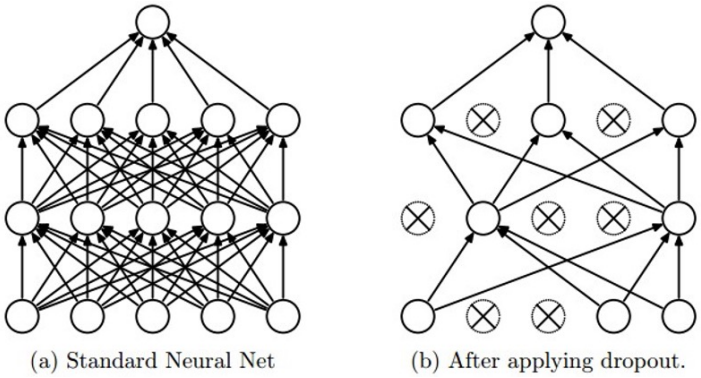


Figure 1.3: Illustration on MC-Dropout [70].

1.3 Ensembles and Sampling approaches

In the previous section, we talked about the solutions for achieving $P(y^*|\mathbf{x}^*, \boldsymbol{\omega})$ in Eq. 1.1. In this section, we will talk about some solutions for achieving $P(\boldsymbol{\omega}|\mathscr{D})$ which can be used to represent the epistemic uncertainty. Bayesian Deep Learning involves modeling the uncertainty in DNNs by treating model parameters as probability distributions. However, calculating the exact posterior distribution of these parameters becomes intractable due to the complex and high-dimensional nature of DNNs. Sampling and ensembling are used to approach the distribution of the parameters. They are more general and scalable with respect to the tasks and the model architecture.

1.3.1 Sampling-based approaches

DNNs are typically trained using deterministic optimization algorithms, which result in a single set of learned parameters. This deterministic nature will leave the DNN's predictions devoid of uncertainty estimates. Sampling methods, such as Monte Carlo Dropout (MC-Dropout) [70] and Bayesian Neural Networks [19], introduce stochasticity into the prediction process. By sampling from different configurations of the model, these techniques generate a distribution of predictions, allowing for the quantification of uncertainty.

Rather than regarding model parameters as fixed values, Bayesian Neural Networks (BNNs) regard them as probability distributions. Referring to Eq. 1.1, the integral over $\boldsymbol{\omega}$ is intractable, yet it can be estimated by taking the average of a limited set of parameter configurations $\{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_J\}$ in practical terms, where there are J sets sampled from the posterior distribution. The approximation for the predictive distribution can be expressed as follows:

$$\begin{aligned} P(y^*|\mathbf{x}^*, \mathscr{D}) &= \int P(y^*|\mathbf{x}^*, \boldsymbol{\omega}) P(\boldsymbol{\omega}|\mathscr{D}) d\boldsymbol{\omega} \\ &\approx \frac{1}{J} \sum_{j=1}^{J} P(y^*|\mathbf{x}^*, \boldsymbol{\omega}_j) \end{aligned} \tag{1.11}$$

BNNs are elegant and easy to formulate, yet their inference is non-trivial. In recent years, the progress in variational inference [109] has enabled a recent revival of BNNs. The variational approach takes a known distribution $Q(\boldsymbol{\omega}|\boldsymbol{\phi})$ and finds the parameters $\boldsymbol{\phi}$ that approximates $Q(\boldsymbol{\omega}|\boldsymbol{\phi})$ to the true Bayesian posterior distribution of the weights $P(\boldsymbol{\omega}|\mathscr{D})$. This process is optimized by minimizing the Kullback-Leibler (KL) divergence between variational and true posteriors, which is also known as the expected lower bound (ELBO) loss:

$$\mathscr{L}_{\text{ELBO}} = \sum_{(\mathbf{x},y)\in\mathscr{D}} D_{\text{KL}}(Q(\boldsymbol{\omega}|\boldsymbol{\phi})||P(\boldsymbol{\omega})) - \mathbb{E}_{Q(\boldsymbol{\omega}|\boldsymbol{\phi})}(\log P(y|\mathbf{x}, \boldsymbol{\omega})) \tag{1.12}$$

However, the integral over $\boldsymbol{\omega}$ is still unavoidable when taking derivatives with respect to $\boldsymbol{\phi}$. To optimize this loss function, Blundell et al. [19] proposed Bayes by Backprop, by leveraging the re-

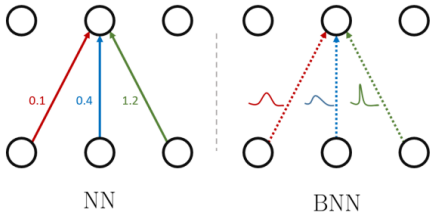


Figure 1.4: Illustration on Bayesian Neural Network (BNN) [19]. In a standard NN, each weight has a fixed value. In most BNNs, the weights follow a known distribution, such as the Gaussian distribution, and are assumed to be mutually independent.

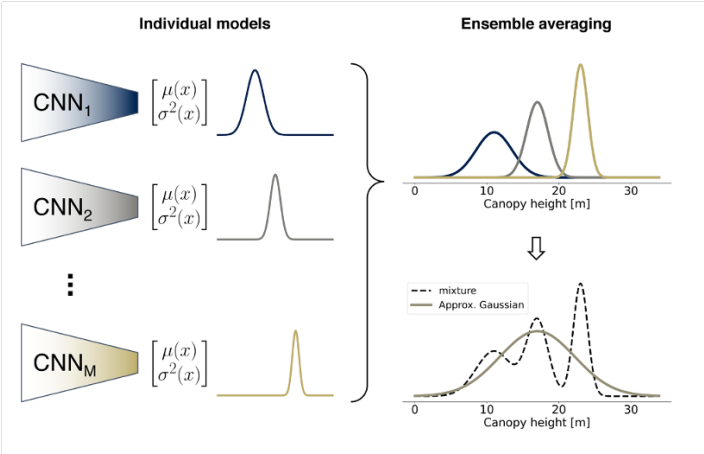


Figure 1.5: Illustration on Deep ensembles [123]. Here is a case of the regression task.

parameterization trick [115] and assume Q($\boldsymbol{\omega}|\boldsymbol{\phi}$) a Gaussian distribution. A simple illustration in Figure 1.4 shows the comparison between normal NN and BNN. During inference, these parameter distributions are sampled to generate a distribution of predictions.

MC-Dropout is another approach to approximate the true posterior. Dropout [217] is a regularization technique commonly used during training to prevent overfitting. Dropout layers apply a form of Bernoulli distribution with a certain probability, i.e., the dropout rate, to sample the input feature maps. In MC-Dropout, the dropout layers remain activated during inference, and we can do forward propagations multiple times to have different predictions, as shown in Figure 1.3.

This process introduces stochasticity into the prediction process, effectively simulating the idea that different predictions are generated by different neuron subsets of the DNN. We can achieve the mean and the variance of the sampled predictions, and the latter can be regarded as the epistemic uncertainty, which represents the uncertainty of DNN's internal structure and parameters.

1.3.2 Ensemble-based approaches

Ensembles can mimic and attain the properties of BNNs [61]. Deep Ensembles (DE) [123] is a popular and pragmatic alternative to BNNs. Figure 1.5 demonstrates the DE of the DNNs with modified output space using Gaussian distribution assumption. The idea of the DE is to train multiple DNNs using different random seeds. The stochasticity comes from the random initialization and the stochastic gradient descent during training, and results in diverse DNN parameters after the training [61]. These separate networks explore different regions of the parameter space, effectively sampling from the space of possible model configurations. Moreover, Allen-Zhu and Li [4] recently proved that the ensemble can learn the multi-view of training data and provide better test accuracy and diverse knowledge of the given data. By combining the predictions of these different models, we can obtain a more reliable estimate of uncertainty, even if we cannot directly compute the posterior distribution.

However, despite the high accuracy and good uncertainty quantification performance the DE provides, its need for high computational cost during both training and inference makes the deployment difficult. Many following works try to decrease the cost and preserve the performance. Snapshot ensembles take the intermediate checkpoints or the final checkpoints using adjusted learning rates in order to save the training-time cost [71, 99]. TRADI [64] computes the posterior distribution of the weights by tracking their trajectory during training. Multi-head networks are also applied to introduce the diversity of the parameters within one DNN [104, 133]. The ensembles composed of small models can also achieve performances comparable to a single large model [117, 150]. OVNNI [65] shows that the ensembles using one-vs-all models can outperform the deep ensembles with the same amount of model components. Ensemble Distillation [159] distills the distribution of the predictions from an ensemble into a single model and retains both the improved accuracy and the information about the diversity of the ensemble. BatchEnsembles [237] addresses computational and memory overhead through multiple low-rank weights tied to a backbone network. Packed-Ensembles [124] leverage grouped convolutions to parallelize the ensemble into a single shared backbone and forward pass to improve training and inference speeds. It is a recently proposed strategy to design and train lightweight structured ensembles by carefully modulating the dimension of their encoding space. More and more solutions manage to mirror and approach the main task and uncertainty quantification quality of DE in a computationally efficient manner close to a single DNN in terms of memory usage, number of forward passes, and image throughput.

1.4 Post-hoc uncertainty quantification solutions

The DNNs have more and more parameters, and the structure becomes more and more complex. For specific vision tasks, the researchers design sophisticated architecture for the DNNs in order to achieve better accuracy on the corresponding benchmarks. Even though for classification-based models, the normalized outputs, such as the Softmax outputs, can be regarded as the confidence measure [88] of the final decision, for the regression models, it lacks outputting more information on uncertainty besides a point estimation. Adjusting the main task model to achieve uncertainty estimates is not guaranteed not to bring negative effects on, for example, changing the robustness and the accuracy of the main task model.

For example, some modifications to existing models may lead to instability in model training, difficulty in searching for hyperparameters, or a decrease in the main task accuracy, etc. Post-hoc solutions can avoid adjusting or re-training the DNNs to achieve uncertainty estimates.

We classify post-hoc solutions into supervised-learning-based and non-learning-based methods. The former is based on auxiliary neural networks, while the latter is based on observations such as gradients during backpropagation, mostly achieving uncertainty estimates based on empirical experience. Both types of solutions have pros and cons. Supervised-learning-based solutions need to train an auxiliary DNN, yet their performance is usually better than the non-learning-based solutions. The non-learning-based ones do not need to design or train extra DNNs, but they usually require more inference time due to the sampling or other operations.

1.4.1 Supervised-learning-based post-hoc methods

Despite the various DNN architectures existing in supervised-learning-based solutions, this type of solution is also based on the Bayesian framework. Given a trained main task DNN, we assume the $\boldsymbol{\omega}$ in Eq. 1.1 is fixed and optimized, and we denote it as $\hat{\boldsymbol{\omega}}$. Therefore, we can re-write $P(\boldsymbol{\omega}|\mathscr{D})$ in Eq. 1.1 as:

$$P(\boldsymbol{\omega}|\mathscr{D}) = \delta(\boldsymbol{\omega} - \hat{\boldsymbol{\omega}}) \tag{1.13}$$

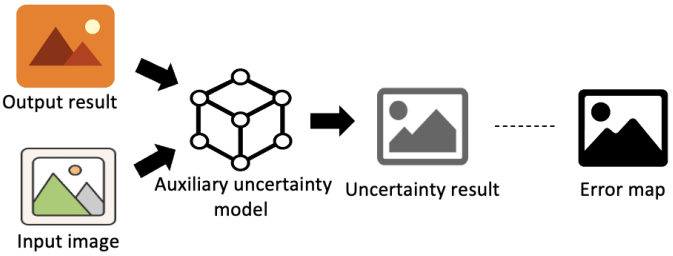


Figure 1.6: Illustration on SLURP procedure [252].

where $\delta(\cdot)$ is the Dirac function. In this case, the Eq. 1.1 and Eq. 1.4 can be re-written as follows in Eq. 1.14 and Eq. 1.15, as also provided by Malinin and Gal [157]:

$$\mathrm{P}(y^*|\mathbf{x}^*,\mathscr{D}) \approx \mathrm{P}(y^*|\mathbf{x}^*,\hat{\boldsymbol{\omega}}) \tag{1.14}$$

$$\begin{aligned}\mathrm{P}(y^*|\mathbf{x}^*,\mathscr{D}) &= \iint \mathrm{P}(y^*|\boldsymbol{\alpha})\mathrm{P}(\boldsymbol{\alpha}|\mathbf{x}^*,\boldsymbol{\omega})\mathrm{P}(\boldsymbol{\omega}|\mathscr{D})d\boldsymbol{\omega}d\boldsymbol{\alpha} \\ &= \int \mathrm{P}(y^*|\boldsymbol{\alpha})\mathrm{P}(\boldsymbol{\alpha}|\mathbf{x}^*,\mathscr{D})d\boldsymbol{\alpha}\end{aligned} \tag{1.15}$$

As the assumption shown in Eq. 1.13, the distribution given by the DNN parameters $\boldsymbol{\omega}$ is transferred as point estimates, which is the basis of post-hoc solutions, i.e., no adjustment and re-training on the given trained main task DNN. In theory, these solutions omit the epistemic uncertainty estimates during modeling. Yet during training, total uncertainty could be involved, and according to the empirical observations in the experiments, we can see that post-hoc solutions are already able to achieve high performance on OOD examples detection.

Aleatoric uncertainty modeling solutions Normally, the prediction errors on the training set are the most important training targets for the auxiliary DNNs. The prediction error[1] $\epsilon = (y - f_{\hat{\boldsymbol{\omega}}}(\mathbf{x}))^2$ has two perspectives to explain. Firstly, we can take it as the total uncertainty. The aleatoric and epistemic uncertainty principally corresponds to the data noise and the imperfect model design (w.r.t. the impossible perfect model parameters $\boldsymbol{\omega}^*$). Another one is just the aleatoric uncertainty. According to the Eq. 1.2, the ϵ can replace the $(\mathbf{x}-\mu)^2$ term. In this case, the auxiliary DNN will only provide the estimation for σ_i^2, and the ground truth for σ^2 is the corresponding prediction error ϵ. Thus, the ϵ can be regarded as the aleatoric uncertainty.

In practice, during training the main task DNN, the epistemic uncertainty of the model's outputs will decrease. This also makes it difficult for the auxiliary DNN to learn useful information and generalize to other scenes or patterns, but can only provide uncertainty estimates for cases where aleatoric uncertainty may occur, such as the intersection of the objects in semantic segmentation and depth estimation tasks. We consider that aleatoric uncertainty makes up the vast majority of the prediction error ϵ, and we do not expect epistemic uncertainty in ϵ learned by the auxiliary DNN to generalize well.

Based on Eq. 1.14 and the above descriptions, we proposed Side Learning Uncertainty for Regression Problems (SLURP) [252]. Figure 1.6 shows the general architecture of the SLURP, which takes both the input image and the output of the main task DNN as the inputs of the auxiliary network. We will go through the details in Chapter 3. Additionally, there are some previous works for classification tasks, for instance, the ConfidNet [41, 42] as shown in Figure 1.7. Different from SLURP, ConfidNet only uses the intermediate features as the inputs of the auxiliary networks. Yet in SLURP, we observe that taking the output of the main task DNN as the input can greatly improve the uncertainty quantification

[1] We can also take the absolution error here, i.e., $\epsilon = |y - f_{\hat{\boldsymbol{\omega}}}(\mathbf{x})|$. For simplicity in the later descriptions, we use the square error.

performance, especially on pixel-wise tasks. In short, a task-specific auxiliary DNN is designed to capture better the uncertainty, especially for pixel-wise tasks. The auxiliary DNN learns the prediction errors, which is the total uncertainty in theory, yet we cannot ensure that the learned total uncertainty can be well generalized to other scenarios, but only the empirical observations are provided.

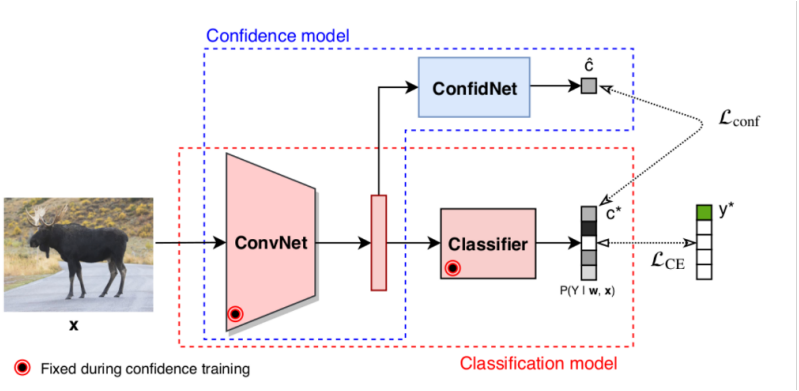


Figure 1.7: Illustration on ConfidNet. ConfidNet takes the features from a trained main task DNN as the input. $\mathscr{L}_{\text{conf}}$ represent the loss between the predicted confidence score and the true class probability given by the main task DNN [41].

There are some other works based on the error learning principle, and we are going to go through them here. Mono Uncertainty [191] aims at uncertainty quantification for self-supervised monocular depth estimation. The idea is to take the main task DNN as the teacher network and an identical DNN as the student network. The student network takes the output of the teacher network as the depth ground truth and models the output space the same as Kendall and Gal [113]. Hu et al. [96] also take the student-teacher strategy, take the ensemble of the main task DNNs as the teacher, and let the student DNN learn the prediction error given by the teacher networks. The main problem with these methods is that they do not model epistemic or distributional uncertainty. In applications, the auxiliary DNNs cannot guarantee to provide correct uncertainty estimates for OOD objects that appear.

Some solutions are proposed to modify the auxiliary DNNs. BayesCap [229] replaces the Gaussian distribution assumption with the generalized Gaussian distribution since the generalized Gaussian distribution is a long-tailed distribution, which can help the auxiliary DNN fit the biased prediction error and provide better uncertainty on the patterns or samples with higher prediction error. Yet, there is still no modeling on the distributional or epistemic uncertainty estimation. Qu et al. [197] proposed a training strategy for auxiliary DNNs using meta-learning. By building virtual training and validation sets during training, the new training procedure highly improved the performance of the auxiliary DNNs. The following work provided by the same group [196] extends the proposed strategy by introducing an additional loss function to help auxiliary DNNs fall on a flattened loss landscape when encountering hard examples. This new strategy shows an improved OOD detection performance. Besnier et al. [13] proposed another improved auxiliary DNN training strategy that can be applied to semantic segmentation. Specifically, they performed data augmentation on the input images for training the auxiliary DNN, and the content of the data augmentation was the adversarial noise oriented to the main task DNN. We consider that these new training procedures are more general and can be applied to the auxiliary DNNs both with or without distributional/epistemic uncertainty modeling.

Epistemic/Distributional uncertainty modeling solutions Epistemic uncertainty modeling is not common in post-hoc solutions, since the basis of this solution is assuming the model parameters to be fixed. Yet, distributional uncertainty modeling is still possible in auxiliary DNNs and provides well OOD example detection performance [108, 157, 209]. Shen et al. [212] proposed an auxiliary network to model the distributional uncertainty for the trained main task DNN. They extract and fuse the feature maps from different layers of the main task DNN, and send the fused features to the uncer-

tainty estimation head. The output space is modeled as a Dirichlet distribution, as introduced in the previous section. For the regression tasks, we modified the auxiliary DNN proposed in SLURP [252] to Discretization-Induced Dirichlet pOsterior (DIDO) [254]. We will go through the details in Chapter III.5.

Furthermore, there are some other auxiliary DNNs. KLoS [40] learn the true class evidence for the evidential classification networks based on the fact that according to Joo et al. [108], the upper bound of the evidence of the true class can be calculated. Thus, similarly to ConfidNet, which learns the true class probability, it is also possible to learn the true class evidence but oriented only to the evidential classification models. Direct Epistemic Uncertainty Prediction (DEUP) [107] uses the density estimator [201] and the model variance estimator to construct the auxiliary DNN. The authors reported good uncertainty quantification performance for the sequential model and also for reinforcement learning models. However, it is hard to adjust the principle to the pixel-wise task because of the density estimator.

In conclusion, in supervised-learning-based post-hoc solutions, various auxiliary DNNs are proposed to obtain the uncertainty estimates without adjusting and re-training the main task DNN. The development of these solutions is similar to that based on the output space of the main task models. Auxiliary networks have also evolved from simple aleatoric uncertainty modeling to modeling simultaneously aleatoric uncertainty and distributional uncertainty. There are also some works that improve the training methods of the auxiliary DNNs so that they can learn better on imbalanced samples and generalize better to more complex scenarios.

1.4.2 Non-learning-based post-hoc methods

There are two general types of solutions for non-learning-based post-hoc methods, namely sampling-based and gradient-based. Both of them are scalable to both image-level and pixel-wise tasks.

Sampling-based post-hoc solutions The representative work for the sampling-based solutions is the Sensitivity as a Surrogate of uncertainty estimation [167], which aims at uncertainty quantification for regression tasks. In this work, the authors proved that two Pearson correlations are proportionally related. One correlation is between the true prediction error and the variance of the model output. Another correlation is between the variance of the model output given a clean input and the variance of the model output given a perturbed input or perturbed model layers. This relation between the two correlations represents the link between the sensitivity and the uncertainty. In this case, without having the ground truth, we can also achieve the uncertainty estimates according to the sensitivity of the model by perturbations. Specifically, there are three ways of the perturbation. In the case of having a black-box trained model, i.e., one cannot get access to the architecture of the model, we can apply simple data augmentation to the input images, for instance, rotations [39] and flipping. For the grey-box trained models, i.e., one can get access to the model architectures, we can apply the dropout layers [217] or the Gaussian noise layers to the intermediate layers of the given DNN. After multiple forward propagations, the variance of the outputs can be regarded as the sensitivity of the model. Using the relation between the sensitivity and the uncertainty, we can consider this variance as the uncertainty estimation of the output. Figure 1.8 illustrates the inference procedures.

Gradient-based post-hoc solutions The gradient-based solution is another uncertainty quantification strategy. With once or twice forward and backward propagations, one can achieve uncertainty quantification with an empirical good quality. These approaches are based on a basic consensus that gradients can provide valuable insights into a model's familiarity and certainty about the input. Oberdiek et al. [181] and Lee and AlRegib [129] provided solutions for the image-level classification task. Both of them use one forward and backward propagation to collect the gradients given by different layers. Oberdiek et al. [181] proposed to use the gradient of the NLL at the predicted class label, i.e., take the current prediction result as the true class label 1, then calculate the NLL

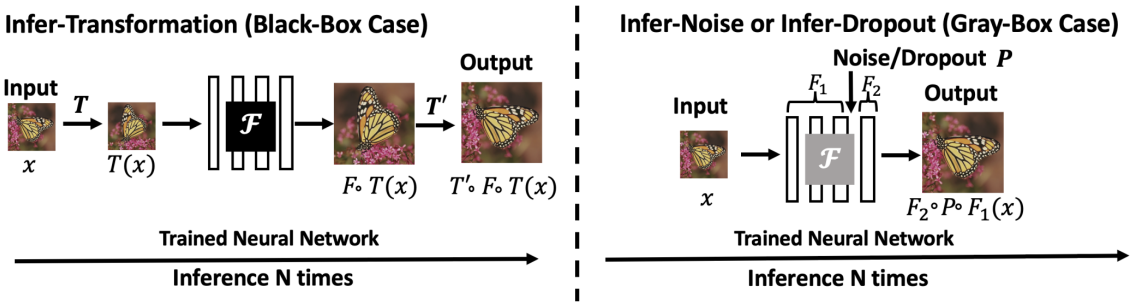


Figure 1.8: Method description of Sensitivity as a Surrogate of uncertainty estimation [167]. Applying Infer-transformation T (left) and infer-noise or infer-dropout P (right) to a trained DNN F during inference.

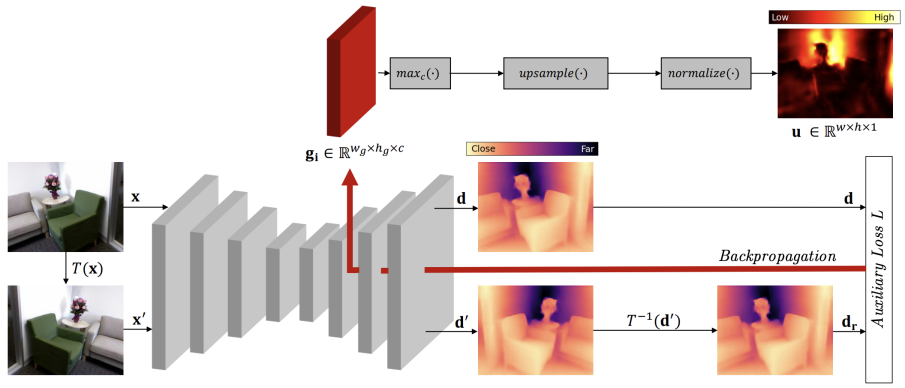


Figure 1.9: Inference procedure of gradient-based uncertainty for monocular depth estimation [92]. $T(\cdot)$ and $T^{-1}(\cdot)$ are the data transformation and its inverse operation. Here the flipping is chosen. **x**, **d**, **g**, and **u** represent the input image, depth map, gradient map and uncertainty map, respectively.

loss. Yet, the reported results underperform the entropy over the Softmax outputs. We argue that the OOD examples might provide high confidence, resulting in fake gradient information, which shows the high familiarity from the model given this input. Lee and AlRegib [129] modified the solution by proposing the so-called confusing label to form the loss function. In detail, the confounding labels are a kind of multi-hot or zero-hot vectors, which can include multiple classes or none. Given the number of class C, we have the new target $\mathbf{y}^{[\text{new}]} = \{0,1\}^{\text{C}}$ with n number of 1, and $n \in \{0,\ldots,\text{C}\}\backslash\{1\}$. Then, the loss function given the input $\mathbf{x}$ becomes:

$$\text{L} = \frac{1}{\text{C}} \sum_{c=1}^{\text{C}} (y_c^{[\text{new}]} \cdot \log(\hat{y}_c) + (1 - y_c^{[\text{new}]}) \cdot \log(1 - \hat{y}_c)) \tag{1.16}$$

where $\hat{\mathbf{y}}$ is the DNN output vector.

The authors argue that by using confounding labels and familiar inputs, the model only needs to learn the relationship between the features it has learned and the confounding labels. Yet, if the DNN is unfamiliar with the input, it needs to learn new features to represent the input and associate it with confounding labels correctly. The squared L_2 norm of the gradient is calculated, then the results from different intermediate layers are concatenated to represent the given input.

Closer to the topic of this thesis is the gradient-based uncertainty for monocular depth estimation [92]. Figure 1.9 illustrates clearly the procedure. The input images consist of the original one and the augmented one. After forward propagation of the DNN, we can achieve two different outputs, and the outputs construct the loss. The DNN backpropagation minimizes this loss. We can extract the gradients from the intermediate layers and finally obtain the uncertainty estimation map after maximization, pooling, and normalization. On the main task model of unsupervised monocular depth

estimation, the quality of uncertainty estimations obtained by this method outperforms that of methods using auxiliary networks [191]. Mainly based on previous work in classification tasks, the author provides extensions and empirical conclusions of this method on monocular depth estimation tasks. The research also discusses what data augmentation methods produce the best uncertainty estimates, and it shows that simple image flipping works best.

The non-learning-based post-hoc method is a scalable uncertainty quantification method. This method usually requires more than one forward or backward propagation, and more effective methods usually require the main task DNN to be a gray-box model rather than a black-box model, i.e., the intermediate features can be obtained. In addition, there is currently no clear connection between this method and Bayesian methods, and the results are usually empirical. We believe that finding a reasonable way to combine this method with Bayesian-based methods will be a valuable research direction.

1.5 Challenges in uncertainty quantification for regression tasks

We consider that there are two main challenges in uncertainty quantification for regression tasks. The first one is the uncertainty quantification benchmarks on the In-distribution dataset. We argue that learning-based and non-learning-based solutions should be considered and compared separately. Based on the ID dataset, learning-based solutions can get access to the ground truth and the corresponding prediction errors, which are strongly related to metrics such as AUSE, which will be introduced in the next chapter. Merging the two types of solutions will cause an unfair comparison. Even though some benchmarks are provided in the literature [86], the community still lacks the ones for pixel-wise tasks and the corresponding robustness analysis.

The second challenge is about OOD example detection in regression tasks. OOD example detection solutions in classification tasks are developing rapidly. The proposed solutions are far more than the types we have mentioned in the above sections. For example, the solutions based on the feature space [131], based on the output space [140, 147], and based on the combination of the feature and the output space [55, 240], etc. However, it is not intuitive to know whether these solutions could be applied to regression tasks. In conclusion, we consider that the fair and standard evaluation of the ID dataset and exploration of the OOD example detection for regression tasks are the current challenges.

Chapter 2

Evaluation for uncertainty quantification

Contents

In this chapter, we introduce evaluation metrics to assess the quality of uncertainty quantification. We also go through the commonly used datasets in uncertainty quantification for different computer vision tasks. At the same time, we present a new synthetic dataset, dubbed multiple uncertainty for autonomous driving (MUAD) dataset. We provide the main task, and the uncertainty quantification benchmarks for the proposed dataset, and some insights from the results. The fair and complete uncertainty quantification evaluation is still a challenge, especially for regression tasks.

2.1 Evaluation for uncertainty quantification

There are usually two perspectives for evaluating the uncertainty or confidence of model outputs. The first part is to evaluate the calibration degree on the in-distribution dataset, such as the test set or validation set. The second part is to evaluate the uncertainty performance on the out-of-distribution (OOD) dataset.

2.1.1 Evaluation on In-Distribution dataset

Evaluating the uncertainty quantification on the ID dataset is more about the model calibration [83, 121]. Guo et al. observed that modern DNNs, such as the ResNet [87], are over-confident in their predictions, which provide uncalibrated results. For a calibrated model, its confidence should match its accuracy. For instance, if we have a DNN output with class cat and confidence 0.2, it means that,

among all the outputs with confidence 0.2, the average accuracy of being a cat should also be 0.2. We call the prediction overconfident if the average accuracy is lower than the corresponding confidence, and vice versa. Formally, we have:

$$\mathrm{P}(\hat{y} = y|\hat{p} = p) = p \quad \forall p \in [0,1] \tag{2.1}$$

where the $\hat{y}$ is the DNN class prediction, $\hat{p}$ is the corresponding confidence. However, Eq. 2.1 is a perfect case. To measure the gap between the calibrated confidence and the current one, we can use the Expected Calibrated Error (ECE) to describe this degree of matching:

$$\mathrm{ECE} = \mathbb{E}_{\hat{p}}[|\mathrm{P}(\hat{y} = y|\hat{p} = p) - p|] \tag{2.2}$$

In practice, we have to binarize the confidence scores into M sets, and we denote B_m as the mth set. The expectations in Eq. 2.2 can be re-written as:

$$\mathrm{ECE} = \sum_{m=1}^{\mathrm{M}} \frac{|\mathrm{B}_m|}{\mathrm{N}} |acc(\mathrm{B}_m) - conf(\mathrm{B}_m)| \tag{2.3}$$

$$\text{where:} \quad acc(\mathrm{B}_m) = \frac{1}{|\mathrm{B}_m|} \sum_{i \in \mathrm{B}_m} \mathbb{I}(\hat{y}_i = y_i)$$

$$conf(\mathrm{B}_m) = \frac{1}{|\mathrm{B}_m|} \sum_{i \in \mathrm{B}_m} \hat{p}_i$$

where $\mathbb{I}(\hat{y}_i = y_i)$ is the Indicator function which is equal to one if $\hat{y}_i = y_i$ and zero otherwise.

In terms of classification, the calibration diagnosis method is relatively intuitive, the Softmax or Sigmoid output can be seen as the confidence level, and it needs to be consistent with the accuracy obtained by the output set with this confidence [83].

Regression calibration evaluation In the regression setting, Kuleshov et al. [121] consider that to study the regression calibration, we need to evaluate the cumulative distribution function (CDF) of the DNN outputs. Then we evaluate if quantiles of the CDF have a proportion of data corresponding to the confidence score. Following the classification settings, to achieve the CDF, we construct the Gaussian distributions with the main task prediction $\hat{y}$ as means, and the uncertainty estimation values $\hat{p}$ as variances. Note that, we use $\hat{p}$ to make the notations consistent, yet here the $\hat{p}$ could be any positive value, rather than bounded between 0 and 1. We consider that the ground truth y is sampled from the corresponding Gaussian distribution. We also let p be a real value between 0 and 1, representing a confidence score. We denote CDF^{-1} the inverse CDF such that $\mathrm{CDF}^{-1}(p) = \inf\{y|p \leq \mathrm{CDF}(y|\hat{y}, \hat{p})\}$. Formally, a regression model is calibrated if:

$$\frac{\sum_{i=1}^{\mathrm{N}} \mathbb{I}(y_i \leq \mathrm{CDF}^{-1}(p))}{\mathrm{N}} \rightarrow p \quad \forall p \in [0,1] \tag{2.4}$$

where $\mathbb{I}(y_i \leq \mathrm{CDF}^{-1}(p))$ is the Indicator function which is equal to one if $y_i \leq \mathrm{CDF}^{-1}(p)$ and zero otherwise.

In the case of an ideally calibrated model for which the observed confidence level matches the expected confidence level, the calibration curve will be a diagonal line. To define the calibration degree of an uncertainty estimation method, we compute the area between the curves made by different uncertainty estimation approaches, and the ideal one will be a good idea. We call the sum of the distances for each confidence level the calibration error.

Area Under the Sparsification Error curve in pixel-wise regression tasks The Sparsification curve and the corresponding Area Under the Sparsification Error curve (AUSE) are commonly used as an uncertainty evaluation approach for pixel-wise regression tasks. It was first proposed by Aodha

et al. [152] for evaluating the uncertainty quantification quality in the optical flow task. The following works, such as MonoUncertainty [191], extend this metric to the monocular depth estimation task. The core of AUSE is to measure the consistency between the predicted uncertainty and the ground truth error. Thus this metric can only be applied on the ID dataset. To build the predicted sparsification curve, first sort the data points in descending order of predicted uncertainty. Then, a certain proportion of the pixels with the highest current uncertainty are removed, and the current average prediction error is recalculated to construct the y-axis of the curve, while the proportion of removed pixels is the x-axis of the curve. When the pixels are removed according to their ground truth prediction error, we can obtain the oracle curve. The AUSE is then defined as the area between the oracle curve and the predictive curve. Depending on the different prediction error metrics, such as root mean square error (RMSE), absolute relative error (AbsRel), etc., different AUSE can be obtained, and we can denote them as AUSE-RMSE, AUSE-REL, etc.

In addition to AUSE, there are some other metrics, such as Uncertainty Calibration Error (UCE) [125] and Area Under the Random Gain (AURG) [191]. We will introduce them in Chapter 5, and we also apply them to evaluate the uncertainty quantification.

2.1.2 Evaluation on Out-Of-Distribution examples

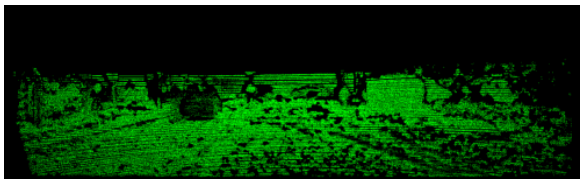
OOD example detection task and its evaluation are more well-known tasks in classification than in regression. Following the classification tasks [89, 247], we separate the evaluation of uncertainty quantification for regression tasks on OOD data into image-level and pixel-wise.

Image-level uncertainty quantification evaluation of OOD examples For image-level tasks, the definition of the OOD examples is similar to the one proposed in classification tasks [223]. In detail, Techapanurak and Okatani categorized the OOD examples into three types: irrelevant inputs, examples with novel classes, and examples with domain shift. We argue that this definition also works in image-level regression tasks. For example, in the age estimation, the DNNs are trained on a face dataset composed of people from ages 5 to 10. In this case, the irrelevant inputs can be the MNIST handwritten digits dataset [126], the examples with novel classes can be older people images, and the last case could be some cartoon children images.

The image-level uncertainty quantification evaluation on OOD data is essentially a binary-class classification. When we want to evaluate the uncertainty, we take the in-distribution dataset as class zero and the OOD dataset as class one. Then, we can apply the metrics that work for evaluating classification performance. Area Under the Receiver Operating Characteristic Curve (AUROC), Area Under Precision-Recall curve (AUPR) and the False Positive Rate at 95% Recall (FPR95) are three widely used metrics for evaluating the OOD example detection task. In practice, we need to find a threshold of the predicted uncertainty to indicate whether the sample is an OOD example. In this case, we need to find an OOD dataset to build the receiver operating characteristic curve and find the optimal point on the Pareto front as the threshold.

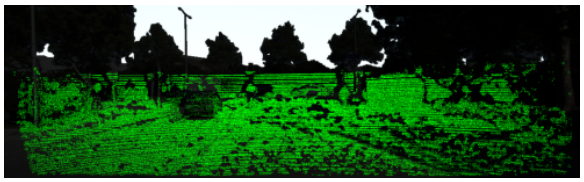
Pixel-wise uncertainty quantification evaluation of OOD examples For pixel-wise tasks, the definition of the OOD examples is relatively vague. Different from the semantic segmentation task, where the OOD examples are semantically far from the ID examples, the OOD examples in regression tasks are not necessarily to be semantically different. For example, on the MUAD benchmark [67], we observe that the examples that are OOD in the semantic segmentation task will not make the main task or the uncertainty quantification a big difference in the monocular depth estimation task. We argue that there are two ways to define the OOD examples on pixel-wise tasks, the images from the OOD dataset and the OOD patterns from the same distribution. The former is similar to the image-level tasks. For instance, our monocular depth estimator is trained on the MUAD dataset [67], which is a synthetic urban dataset, and the OOD data can be sampled from the NYU in-door dataset [176]. Concerning the OOD patterns from the same distribution, they are very specific and need to be picked

(a) RGB image example from KITTI dataset.



(b) The corresponding ground truth depth map.

(c) The corresponding semantic segmentation map.



(d) Overlapped ground truth map for uncertainty evaluation.

Figure 2.1: Visualization on the ground truth building for evaluating OOD example detection on the monocular depth estimation task. On the sub-figure (d), green points represent the depth ground truth, i.e., the in-distribution part. White parts represent the sky pattern, which is the OOD part.

according to the task. For example, in monocular depth estimation, we define the sky pattern in the image as the OOD pattern, as in our proposed work DIDO [254]. The depth of the sky is undefined and the DNN never sees this pattern or the positions of this pattern during training. We are sure that the depth prediction on this pattern is unreliable, and the uncertainty quantification on this pattern should be consistent and numerically highest on the prediction map. Even though we find a case of the OOD example on a pixel-wise task, the definition is still unclear, and it is also useful to find more cases like this in other tasks, such as optical flow estimation and the crowd counting task.

The evaluation of the uncertainty estimates for OOD examples on pixel-wise tasks has two types according to the two definitions in this last paragraph. For the OOD dataset case, we can take the uncertainty quantification ground truth on the ID dataset as 0 and the ones on the OOD dataset as 1. Then, we use AUROC, AUPR, and FPR95 to evaluate per-pixel uncertainty and take the average as the final score. For the OOD pattern case, we take the uncertainty quantification ground truth on the sky area as 1 and 0 for the pixels with the main task ground truth. Figure 2.1 provides an example taken from the KITTI dataset, where the sky part is annotated. Thus, we can take this image for evaluation.

Overall, the uncertainty quantification evaluation for OOD examples is different on image-level and pixel-wise tasks in regression. We use binary-class classification metrics for both cases, while the difference is in the definition of the OOD examples. On pixel-wise tasks, the definition of the OOD examples is different from the semantic segmentation, making this type of OOD example detection evaluation uncommon.

2.2 Evaluation datasets for uncertainty quantification

2.2.1 Datasets for calibration evaluations

For classification tasks, evaluation usually includes the calibration degree of the model and the OOD example detection performance. The former is measured on the in-distribution dataset, i.e., the test set or the validation set of the training dataset. ImageNet [48], CIFAR10, and CIFAR100 [120] are the commonly used in-distribution datasets. The robustness of the calibration degree can also be evaluated by applying the image corruption techniques to these datasets. For instance, the following eighteen perturbations with five severities can be applied to the validation or the test set: Gaussian noise, shot noise, impulse noise, iso noise, defocus blur, glass blur, motion blur, zoom blur, frost, fog, snow, dark, brightness, contrast, pixelated, elastic, color quantization, and JPEG. Hendrycks and Dietterich [90] provide a benchmark for the model robustness based on the previous-mentioned

datasets.

Similarly, on the semantic segmentation task and the object detection task, the perturbations can also be applied to the images. Michaelis et al. [169] provide benchmarks for the robustness of the object detection task. The authors follow the image corruption settings in [90] to build Pascal-C, Coco-C, and Cityscapes-C based on the original clean datasets [43, 59, 143]. For the semantic segmentation task, except for the above-mentioned corruptions, Sakaridis et al. [206] and Hu et al. [97] also provide the synthesized foggy and rainy weather conditions on the original Cityscapes dataset, named FoggyCityscapes and RainyCityscapes, respectively.

For regression tasks, the consistency between the predicted uncertainty and the prediction error is evaluated on in-distribution datasets. For the optical flow task, Ilg [104] and Yu et al. [252] evaluate this consistency on Sintel [24], KITTI [227] and FlyingChairs [54] datasets. For the monocular depth estimation task, Malini and Gal [156] and Amini et al. [5] use NYU indoor dataset [176]. Poggi et al. [191] and Yu et al. [252] use KITTI [75]. Moreover, in the RoboDepth challenge [118], similar corruptions as in [90] are applied to the KITTI and NYU datasets, while the goal is to measure the robustness of the main task prediction and test images are in a smaller resolution. In our proposed DIDO [254], we apply the same corruptions to the KITTI dataset and keep the original resolution for the uncertainty quantification evaluation.

2.2.2 Datasets for out-of-distribution example detection evaluations

For image classification tasks, according to Techapanurak and Okatani [223], based on the in-distribution dataset, we can use irrelevant inputs, examples with novel classes, and examples with domain shift to build the OOD datasets. For instance, the models trained on CIFAR10 are often evaluated on MNIST [126], iNaturalist [233], SUN [243], Places [259], and Textures [38], which are the irrelevant inputs. The evaluation can also be applied based on the CIFAR100 dataset, which belongs to a similar color space distribution yet contains novel classes. The domain shift is to turn the dataset to a closer domain. For example, CIFAR10 is a real-world tiny image dataset, while a dataset with a closer domain could be the one that has similar content but is composed of stick figures or comic-style pictures. In the semantic segmentation task, OOD examples are defined as objects with different semantics from the training dataset. For instance, in our proposed MUAD dataset [67], the trains, bicycles, motorcycles, and animal patterns are not in the training and validation dataset but only exist in the OOD test set. More datasets will be mentioned in Section 2.3.

In regression tasks, we provide the KITTI Seg-Depth dataset for evaluating the OOD pattern detection performance using both the KITTI depth dataset and KITTI segmentation dataset [3]. This design follows the examples with novel classes setting in [223]. The evaluation based on the dataset change OOD detection can be applied to the two different datasets with different scenarios. For example, as we mentioned in Section 2.2.2, we can choose two datasets with outdoor and indoor scenarios, respectively, based on the irrelevant input setting, for example, NYU [176] and KITTI [75] datasets. Meanwhile, we can also follow the setting of the examples with domain shift, such as the synthetic MUAD [67] dataset and the real-world KITTI dataset.

As we have seen, perturbations applied to images (such as different weather conditions) are often post-added. At the same time, we believe that the community lacks a dataset that has OOD objects, different weather conditions with the same distribution as the dataset and supports both regression and classification tasks. In the next section, we present the MUAD dataset [67]. It contains the previous-mentioned conditions and aims to have a fair and complete evaluation for uncertainty quantification solutions.

2.3 Multiple uncertainty for autonomous driving (MUAD) dataset

2.3.1 Introduction

For autonomous driving, uncertainty estimation and reliability are essential for safely deploying DNNs in real-world conditions. Here, DNNs are expected not only to reach high predictive performance and real-time inference speed but also to deal effectively with the two types of uncertainty under various forms (noise, distribution shift, out-of-distribution samples, sensor degradation, etc.). In the last years, numerous works have moved the needle towards more reliable predictive uncertainty for DNNs [13, 19, 63, 70, 113, 123, 171, 172, 237]. However, evaluating such methods is not obvious as there is no ground truth for uncertainty, and the different sources of uncertainty are conflated due to prior data curation.

Most datasets aim to improve the predictive performance of DNNs [43, 77, 178, 251], only recently datasets addressed the robustness of DNNs under unseen weather conditions [45, 207, 208] or objects [18, 30, 89]. However, these datasets are either limited to only one task, typically semantic segmentation, or only focus on a single type of uncertainty, or are not precise enough in the different levels of uncertainties.

We introduce a new dataset to study uncertainty estimation methods for perception in autonomous vehicles and address these limitations in our dataset. Our dataset, MUAD (Multiple Uncertainties for Autonomous Driving), is composed of 3,420 images for training, 492 for validation, and 6,501 for testing. The training and validation sets contain only the daytime and nighttime clear-weather urban scenarios. Yet the images in test sets not only cover day and night conditions, but they also contain different weather conditions and multiple OOD objects, which allows us to quantify all levels of uncertainty.

Contributions

1. We introduce MUAD: a new automotive dataset with annotations for multiple tasks and multiple uncertainty sources.
2. We perform a wide range of benchmarks on the MUAD dataset for multiple computer vision tasks and settings (semantic segmentation, depth estimation, object detection) to further support research in this area.
3. We conduct an extensive study on uncertainty quantification for pixel-wise classification and regression tasks.

2.3.2 Related work

A variety of real-world datasets for autonomous driving have been recently released [26, 32, 43, 77, 93, 198, 221, 234, 251]. They have enabled tremendous progress in the area but they typically focus on a single task, e.g., semantic segmentation [43, 198, 251], object detection [26, 77, 221], motion prediction [32, 93] and do not have evaluation tracks for uncertainty and out-of-distribution detection. Synthetic datasets, e.g., GTA-V [202], SYNTHIA [204], virtual KITTI [69] can provide abundant training data alleviating the need for costly annotation of real images as well as privacy preservation concerns in the case of real data. Currently, they are mostly designed and used for domain adaptation, typically imitating the content and classes from a given real dataset. Several datasets have emerged towards meeting the reliability requirement for self-driving vehicles [18, 30, 89, 189] and evaluate the performance of semantic segmentation DNNs when facing out-of-distribution objects (OOD). Other datasets investigate the robustness against different weather conditions, e.g., night [45, 46, 208], rain [208, 226], fog [206, 208], however, they are often acquired in different locations and conditions leading to a performance drop that overlaps with the one from the difficult weather conditions.

In order to provide images of the same locations, to address the lack of diversity in real environments

Dataset	Adversarial annotations	Fog	Night	Rain	Snow	Classes	Out of distribution	Depth	Object detection	2D/3D	Instance segmentation
Foggy Driving[206]	101	✓	-	-	-	19	-	-	✓		-
Foggy Zurich [45]	40	✓	-	-	-	19	-	-	-		-
Nighttime Driving [46]	50	-	✓	-	-	19	-	-	-		-
Dark Zurich [207]	201	-	✓	-	-	19	-	-	-		-
Raincouver [226]	326	-	✓	✓	-	3	-	-	-		-
WildDash [257]	226	✓	✓	✓	✓	19	-	-	-		-
BDD100K [251]	1346	✓	✓	✓	✓	19	-	-	-		-
ACDC [208]	4006	✓	✓	✓	✓	19	-	✓	✓		-
Virtual KITTI 2 [25]	21260	✓	-	✓	-	14	-	✓	✓		✓
Fishyscapes [18]	373	-	-	-	-	19+2	✓	-	-		-
LostAndFound [189]	1203	-	-	-	-	19+9	✓	-	-		-
RoadObstacle21 [30]	327	-	✓	-	✓	19+1	✓	-	-		-
RoadAnomaly21 [30]	100	-	-	-	✓	19+1	✓	-	-		-
Streethazard [89]	6625	-	-	-	-	13+250	✓	-	-		-
BDD anomaly [89]	810	✓	✓	✓	✓	17+2	✓	-	-		-
MUAD	10413	✓	✓	✓	✓	16+9	✓	✓	✓		✓

Table 2.1: Comparative overview of the different datasets for uncertainty on autonomous driving.

and to evaluate better the impact on the epistemic uncertainty, some works promoted inpainting of virtual objects [89] or synthesized weather conditions [225]. In this setting, however, questions may be raised about the veracity of the result. Therefore, the recent ACDC dataset [208] is composed entirely of real images taken from the same locations, and includes multiple sources of aleatoric uncertainty. However, not having any control over the noise level makes it harder to quantify the link between noise and uncertainty. Acquiring images with uncertainty corner cases is problematic as these cases are rare (long tail) and also costly to annotate, e.g., 3.3 hours/image [208]. Given this scarcity, such images are better used for validation as a small test set to assess the reliability of DNNs before deployment. These system validation stages can be seen as stress tests with corner cases to mirror challenging real-world conditions. It is thus interesting, even from a more applied standpoint, to have a synthetic dataset that mimics these rare conditions with some good fidelity constraint to quantify the robustness of DNNs. Synthetic data is abundant and can allow us to measure finer drifts in the input distribution. In addition, most such datasets mainly focus on semantic segmentation, while we propose to address multiple tasks (semantic segmentation, monocular depth, object detection, and instance segmentation).

In Table 2.1, we provide a summary of the main existing uncertainty datasets. Here, we propose a fully synthetic dataset called MUAD, integrating different weather conditions with various intensities and suitable for a multitude of vision tasks and for the comprehensive characterization of their uncertainty.

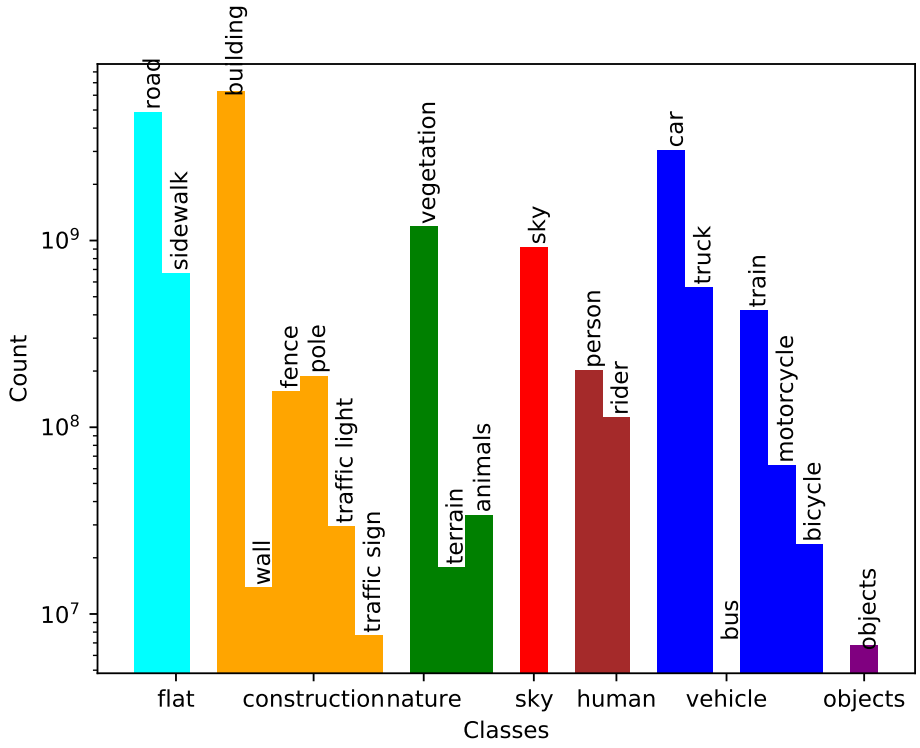


Figure 2.2: Number of annotated pixels per class in MUAD.

2.3.3 Multiple Uncertainties for Autonomous Driving dataset description

According to the categorization of the uncertainty in line with the current works of the community [73], we propose to use the dataset to better evaluate the results and uncertainty estimations given by the DNNs in the context of autonomous driving. Let us link the two main types of uncertainty - aleatoric and epistemic - to the specific context of our application. In the scenario of autonomous driving, we believe that the aleatoric uncertainty of the DNNs will occur due to different weather conditions than the ones present in the training set. The epistemic uncertainty of the DNNs should arise when the class or the appearance of objects in the picture differ from those of the data provided in the training set. The design of the MUAD dataset is based on this hypothesized relationship between uncertainty and autonomous driving scenarios. In the remainder of this section, we will detail the composition of the MUAD dataset.

The goal of MUAD is to confront DNNs in uncertain environments and to characterize numerically their robustness in adverse conditions, more specifically in the presence of rain, fog, and snow. Photorealism is essential for guaranteeing that synthetic datasets are challenging with respect to real-world conditions and also for keeping them relevant for use in industrial applications. This is particularly important for accommodating weather artifacts [138, 225, 236]. Our dataset is generated using a physics-based synthetic image rendering engine to produce high-quality realistic images and sequences. The engine uses an accurate light transport model [155, 235] and provides a physics description of lights, cameras, and materials. This allows for a detailed simulation of the amount of light that is reaching the camera sensor. The camera sensor itself is simulated, converting the energy coming from the scene in the form of photons into electrons. Electrons are finally converted into a voltage that is digitized to produce the digital values that represent the color image. We provide the photorealistic rendering descriptions for different weather conditions in Section 2.3.3.3. For each sample in MUAD, the corresponding ground truth information contains the semantic segmentation, the depth map, and for some specific classes (pedestrian, car, van, traffic light, traffic sign) the instance segmentation with the corresponding bounding boxes. We follow the standard data split strategy. However,

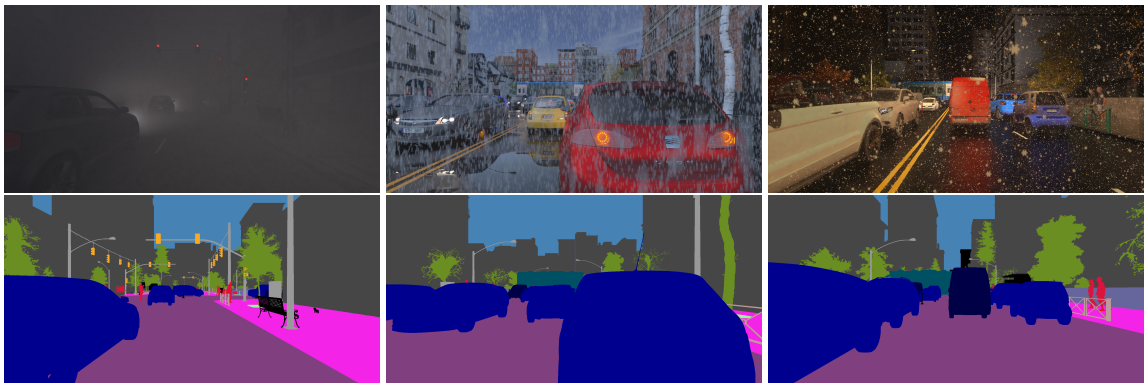


Figure 2.3: Illustration of semantic segmentation images of MUAD dataset. The first row is composed of the original images of the **high adv. set**. The second row is their corresponding ground truth.

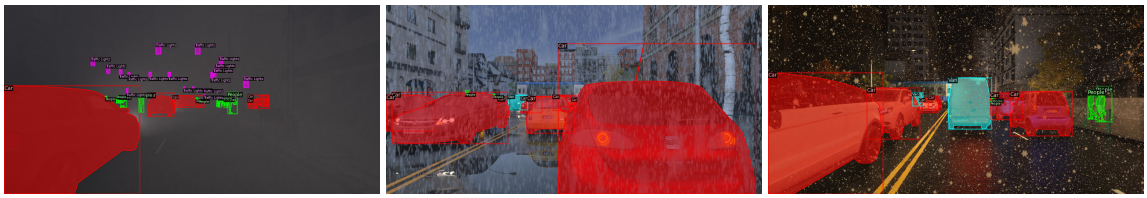


Figure 2.4: Illustration of instance segmentation images of MUAD dataset. The three images are selected from the **high adv. set**. We illustrated fog, rain, and snow conditions.

the training and validation set contains only images with normal weather conditions and without some specific classes, which are denoted as OOD. The test set is organized into seven subsets following the intensity of the adverse weather conditions:

- **normal set:** images without OOD objects nor adverse conditions.
- **normal set overhead sun:** images without OOD objects nor adverse conditions, in which we simulate the sun with a zenith angle of 0°, that we denote for the sake of simplicity as overhead sun.
- **OOD set:** images with OOD objects and without adverse conditions.
- **low adv. set:** images with medium intensity adverse conditions (fog, rain or snow).
- **high adv. set:** images containing high intensity adverse conditions (fog, rain or snow).
- **low adv. with OOD set:** images containing both OOD objects and medium intensity adverse conditions (fog, rain or snow).
- **high adv. with OOD set:** images containing both OOD objects and high intensity adverse conditions (fog, rain or snow).

In Figure 2.3 and 2.4, we illustrate the instance segmentation and the semantic segmentation of three images. In Figure 2.5, we show three images from the OOD test set.

The adverse weather conditions are realistic and challenging as they bring a mix of difficult (unknown during training) environment conditions and perturbation of the visibility in the scene. We argue that such settings are helpful for autonomous driving since the autonomous system must face and be robust against a variety of weather conditions and situations.

2.3.3.1 MUAD statistics

Our dataset contains 3,420 images in the train set and 492 in the validation set. The test set is composed of 6,501 images divided as follows: 551 in the **normal set**, 102 in the **normal set no shadow,**

Figure 2.5: Illustration of images with OOD examples from MUAD dataset. The three images are selected from the **OOD set**. The `animal`, `rocks`, `trash bags`, and `food track` are the OOD examples.

Cityscapes classes	**MUAD classes**	**nb. of images with the annotations**
Road	Bots, Tram Tracks, Crosswalk, Parking Area, Garbage - Road, Road Lines, Sewer Longitudinal Crack, Transversal Crack, Road, Asphalt hole, Polished Aggregate, Vegetation - Road, Sewer - Road, Construction Concrete	9,055
Sidewalk	Lane Bike, Kerb Stone, Sidewalk, Kerb Rising Edge	8,948
Building	House, Construction Scaffold, Building, Air Conditioning, Construction Container, TV Antenna, Terrace, Water Tank, Pergola Garden, Stairs, Dog House, Sunshades, Railings, Construction Stock, Marquees, Hangar Airport	9,089
Wall	Wall	1,101
Fence	Construction Fence, Fences	8,622
Pole	Traffic Signs Poles or Structure, Traffic Lights Poles, Street lights, Lamp	8,984
Traffic light	Traffic Lights Head, Traffic Cameras, Traffic Lights Bulb (red, yellow, green)	8,222
Traffic sign	Traffic Signs	2,672
Vegetation	Vegetation	9,072
Terrain	Terrain, Tree Pit	8,377
Sky	Sky	8,591
Person	Walker, All colors of Construction Helmet, All colors of Safety Vest, Umbrella, People	8,843
Rider	Cyclist, Biker	3,470
Car	Car, Beacon Light, Van, Ego Car	9,026
Truck	Truck	5,533
Bus	Bus	0
Train	Train, Subway	2,240
Motorcycle	Motorcycle, Segway, Scooter Child	2,615
Bicycle	Bicycle, Kickbike, Tricycle	2,816
Animals	Cow, Bear, Deer, Moose	603
Objects anomalies	Food Stand, Trash Can, Garbage Bag	352
Background	Others	-

Table 2.2: Overview of annotated classes

1,668 in the **OOD set**, 605 in the **low adv. set**, 602 in the **high adv. set**, 1,552 in the **low adv. with OOD set** and 1,421 in the **high adv. with OOD set**. All of these sets cover day and night conditions

with 2/3 of day images and 1/3 of night images. Test datasets address diverse weather conditions (rain, snow, and fog with different levels), and various OOD objects. The resolution of all images is 1024×2048.

The dataset aims to provide general and consistent coverage for a typical urban and suburban environment under different times of day and weather conditions. Ego-vehicle poses are drawn randomly within a complex environment, and in the second stage, the field of view is populated stochastically with dynamic objects of interest following distributions in compliance with their expected behavior. The pose and context changes, as well as the variation of the models for the objects of interest, ensure that content diversity is high, in addition to images being photorealistic. The simulator makes use of approximately 300 different person models and 150 different vehicle models, which are sampled while varying their visual characteristics.

2.3.3.2 Class labels

The class ontology of MUAD is presented in Table 2.2. MUAD comprises 155 different classes that we have regrouped into 21 classes. The first 19 classes are similar to the CityScapes classes [43], then we added object anomalies and animals to have more diversity in the anomalies. In addition to ensuring high content diversity, this ontology facilitates the mapping of MUAD to specific environments that require or impose a lower number of more generic classes. Consequently, trained models are easily transferable for existing datasets, and we provide the mapping towards the 21 classes widely used by the community, e.g., [36, 43, 202, 204]. The dataset statistics for the 21 classes are presented in Figure 2.2. For the evaluation of OOD detection, we have excluded nine classes (train, motorcycle, bicycle, bears, cow, deer, moose, food stand, and garbage bags) from the training and validation sets. These classes are present in the test set as OOD objects. DNNs that process samples belonging to one of these nine classes are expected to have a low confidence score.

2.3.3.3 Photorealistic rendering

Our physically based approach simulates the weather conditions taking into consideration the amount of ozone and humidity, among other factors. **Regarding the sky**, the renderer uses a physical model of the light coming from the sky. The amount of ozone and humidity in the atmosphere changes the emissive spectral profile of the sky, impacting the color of the objects in the scene. Apart from ozone and humidity, there are other factors that the render takes into account, for instance, turbidity and scattering asymmetry. **Regarding the rain and the snow**, the simulation of every raindrop allows us to model physical dispersion. For improved realism, we choose the falling speed and size of raindrops according to observed real rain [9, 173]. For snow, the same principle applies, but changing, in this case, the material and the dynamics. **Regarding the fog**, we use a full volumetric approach for the simulation where scattering effects are considered. **Regarding the level of noise**, to the best of our knowledge, there is no standard procedure to measure the intensity of adverse weather conditions for driving scenarios. We empirically selected the number of raindrops, snowflakes, and fog intensity from a human point of view. All the efforts mentioned above improved our dataset realism. A study [163] was performed that confirmed that our render enhances the realism of MUAD compared to SYNTHIA.

2.3.4 Experiments

We will provide the semantic segmentation and object detection experiments in the Appendix. Here we only present the experiments on monocular depth estimation tasks.

2.3.4.1 Supervised monocular depth estimation

We provide results for monocular depth using NeWCRFs [256], which is one of the SOTA on the KITTI dataset [77]. NeWCRFs does not output uncertainty by default. Similarly to [104, 113, 180],

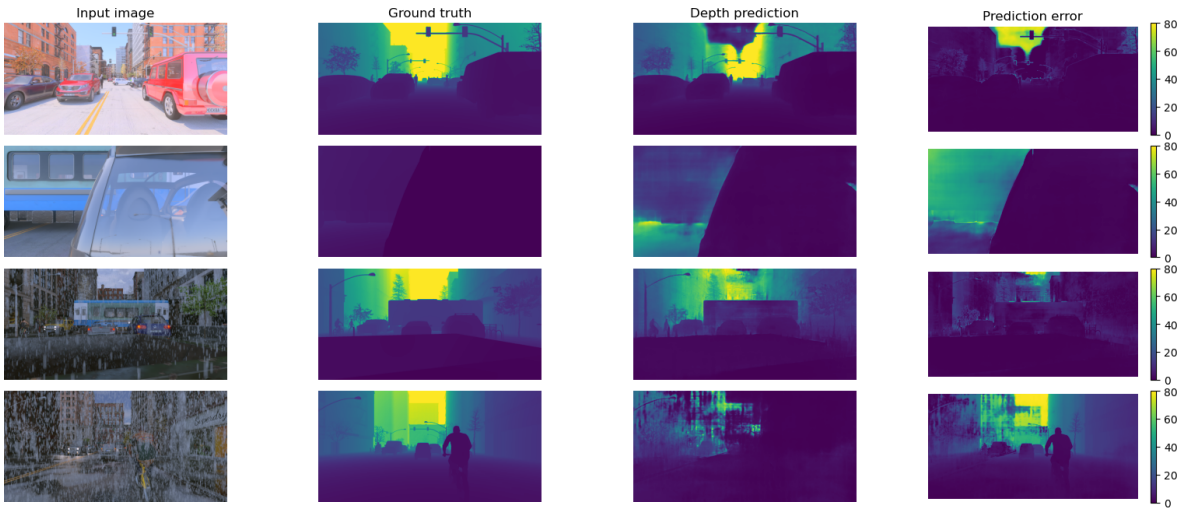


Figure 2.6: Visualization of monocular depth estimates on the testing images given by the NeWCRFs model. The first row and the second rows show the clear weather cases, and the rest show the rainy cases. The `train` and `bicycle` on the last three inputs are semantically OOD examples, which do not exist on the training and validation set.

we modify the DNN to output the parameters of a Gaussian distribution (i.e., the mean and variance). We denote the result as single predictive uncertainty (Single-PU). Based on this modification, we train a Deep Ensembles [123] with three DNNs. We also provide the results from SLURP [252], which needs two DNNs to predict the depth and the uncertainty, respectively, and MC-Dropout [70]. For depth evaluation, we use the same metrics as Eigen et al. [56], which are used in many following works [130, 256]. For uncertainty quality evaluation, we follow the implementation of Poggi et al. [191]. Table 2.3 lists some of the depth and uncertainty results of the above techniques on our dataset due to the space limit. We provide the full results of each test set in the Appendix.

We observe that in the presence of OOD, the uncertainty results of Deep Ensembles are comparatively better, while MC-Dropout provides more robust depth estimations under different perturbations. Figure 2.6 shows the qualitative results of the depth predictions given by the NeWCRFs model. We can see that the depth model is more robust on the semantically OOD examples, i.e., even though the train and bicycle in the images have different semantics than other objects in the training and validation sets, the predictions given by the depth model for these objects are not much different from other objects. In comparison, different weather has a greater impact on depth models. As shown in Table 2.3, the result difference between the OOD set and the normal set is smaller than the difference between the normal set and the low/high adv. sets. The difference between low/high adv. with OOD sets and low/high adv. without OOD sets follows the tendency of the one between normal and normal OOD sets.

We also propose a baseline method for depth domain adaptation from MUAD to KITTI and report its performance in Table 2.4. Compared to the direct adaptation from Virtual KITTI2 [69], which is specifically designed based on the target dataset KITTI, the model trained on MUAD can achieve competitive performance.

2.3.4.2 Self-supervised monocular depth estimation

In this section, we provide the self-supervised monocular depth results for MUAD. In order to provide a wider variety of urban scenarios, there are no consecutive frames in MUAD, but still provide pictures taken by the left and right cameras. We provide self-supervised monocular depth results on MUAD in Table 2.5 using DIFFNet [260] and left-right consistency [79] strategy. DIFFNet is one of the SOTA on the KITTI dataset. We train a DIFFNet model with 12 images as the batch size, randomly crop the image to 512*1024, and train 20 epochs in total.

Methods	normal set					low adv. without OOD set				
	Depth results			Uncertainty results		Depth results			Uncertainty results	
	d1 ↑	AbsRel ↓	RMSE ↓	AUSE RMSE ↓	AUSE Absrel ↓	d1 ↑	AbsRel ↓	RMSE ↓	AUSE RMSE ↓	AUSE Absrel ↓
Baseline	0.922	0.114	3.357	-	-	0.786	**0.147**	5.005	-	-
Deep Ensembles [123]	**0.929**	**0.111**	**3.199**	**0.291**	0.060	0.767	0.156	4.892	**0.740**	0.105
MC Dropout [70]	0.919	0.119	3.209	0.634	0.061	**0.798**	0.151	**4.580**	1.063	0.098
Single-PU [113]	0.905	0.132	3.230	0.313	0.081	0.773	0.159	4.865	0.789	0.112
SLURP [252]	0.922	0.114	3.357	0.467	**0.048**	0.786	**0.147**	5.005	1.167	**0.090**

Methods	high adv. without OOD set					normal set overhead sun				
	Depth results			Uncertainty results		Depth results			Uncertainty results	
	d1 ↑	AbsRel ↓	RMSE ↓	AUSE RMSE ↓	AUSE Absrel ↓	d1 ↑	AbsRel ↓	RMSE ↓	AUSE RMSE ↓	AUSE Absrel ↓
Baseline	0.632	**0.207**	6.989	-	-	0.951	0.090	3.646	-	-
Deep Ensembles [123]	0.566	0.243	7.498	**1.182**	0.153	**0.955**	**0.083**	3.479	**0.336**	0.055
MC Dropout [70]	**0.657**	**0.207**	**6.278**	1.382	**0.128**	0.948	0.092	**3.407**	0.786	0.058
Single-PU [113]	0.571	0.248	7.680	1.740	0.171	0.946	0.105	3.546	0.358	0.079
SLURP [252]	0.632	**0.207**	6.989	1.707	**0.128**	0.951	0.090	3.646	0.525	**0.033**

Methods	OOD set					low adv. with OOD set					high adv. with OOD set				
	Depth results			Uncertainty results		Depth results			Uncertainty results		Depth results			Uncertainty results	
	d1 ↑	AbsRel ↓	RMSE ↓	AUSE RMSE ↓	AUSE Absrel ↓	d1 ↑	AbsRel ↓	RMSE ↓	AUSE RMSE ↓	AUSE Absrel ↓	d1 ↑	AbsRel ↓	RMSE ↓	AUSE RMSE ↓	AUSE Absrel ↓
Baseline	0.896	0.125	3.616	-	-	0.713	2.637	4.764	-	-	0.555	0.459	6.916	-	-
Deep Ensembles [123]	**0.903**	**0.114**	3.447	**0.427**	0.074	0.709	**1.810**	4.707	**0.692**	**0.129**	0.521	**0.331**	7.411	**1.072**	**0.151**
MC Dropout [70]	0.893	0.145	**3.432**	0.724	0.080	**0.744**	3.925	**4.364**	0.927	0.206	**0.610**	0.545	**6.176**	1.245	0.314
Single-PU [113]	0.888	0.132	3.463	0.447	0.095	0.714	4.349	4.716	0.744	0.482	0.529	0.351	7.627	1.347	0.156
SLURP [252]	0.896	0.125	3.616	0.721	**0.068**	0.713	2.637	4.764	1.072	0.212	0.555	0.459	6.916	1.564	**0.151**

Table 2.3: Comparative results for monocular depth on MUAD. We use NeWCRFs [256] as the based DNN for the monocular depth task.

We observe that OOD objects have less impact on the results of monocular depth estimation in the Self-supervised monocular depth. According to [52], monocular depth estimation based on left-right coherence is sensitive to illumination conditions, particularly to object shadows. However, our results on the *Normal set* and *Overhead sun set* do not seem to confirm this point. We believe that DNNs learn depth without necessarily paying much attention to shadows; hence they have no impact on the performance of the self-supervised monocular depth model.

2.3.5 Discussion

The experiments show that the best main task contender might not always be the most suited against different sources of uncertainty. Thus, it is important to test thoroughly and adapt the processing pipeline to the expected type of perturbations. The similar ranking of methods on our synthetic dataset and on real data (see [62, 89]) is encouraging as it allows us to generalize the analysis performed on MUAD to actual scenarios. An additional benefit of synthetic datasets is related to the reduced data privacy concerns and regulations that typically affect real-world datasets, in particular in urban settings that include pedestrians. All these traits allow for faster validation of new algorithms before their deployment in real-world settings. Finally, a potential different usage of MUAD concerns unsupervised domain adaptation from synthetic to real domains. Our preliminary results are encouraging.

	KITTI							
Training set	d1↑	d2↑	d3↑	Abs Rel↓	Sq Rel↓	RMSE↓	RMSE log↓	SILog↓
KITTI [77]	0.975	0.997	0.999	0.052	0.148	2.072	0.078	6.9859
Virtual KITTI 2 [25]	0.835	0.957	0.989	0.129	0.706	4.039	0.177	15.534
MUAD	0.731	0.927	0.983	0.187	1.059	4.754	0.227	18.581

Table 2.4: Comparative results for monocular depth estimation simple domain adaptation from MUAD to KITTI eigen-split [56]. The first row is the original baseline, and the second and the third rows are the performance of the model trained directly on Virtual KITTI 2 [69] and MUAD, respectively.

Evaluation sets	AbsRel ↓	log10 ↓	RMSE ↓	SqRel ↓	log_RMSE ↓	d1 ↑	d2 ↑	d3 ↑
Normal	0.365	0.111	5.646	2.234	0.350	0.638	0.874	0.919
Overhead sun	0.174	0.079	5.875	1.426	0.249	0.693	0.953	0.978
low adv. without OOD	0.312	0.185	10.472	3.951	0.586	0.442	0.716	0.824
high adv. without OOD	0.510	0.432	15.578	8.513	1.194	0.227	0.417	0.531
OOD	0.312	0.101	6.170	2.663	0.331	0.648	0.899	0.941
low adv. with OOD	1.462	0.192	9.356	6.054	0.601	0.431	0.697	0.807
high adv. with OOD	1.141	0.415	14.415	25.281	1.194	0.236	0.426	0.543

Table 2.5: Self-supervised monocular depth results on all test sets given by DIFFNet [260]. Since the results are provided by a single solution, we do not make the comparisons here.

2.3.6 Conclusion

In this section, we propose a novel dataset on autonomous driving scenarios named MUAD. In addition to different urban scenarios, the dataset also contains test images with different lighting conditions and weather conditions. Furthermore, the various semantically OOD examples are also provided in the test sets. The dataset is synthetic, which means it has more precise and correct annotations than human annotations. This work covers semantic segmentation, depth estimation and object detection tasks, making multi-modal domain transfer possible. More importantly, since the OOD examples and the weather conditions come from the same data distribution as the other parts, we argue that this dataset can provide a complete and fair evaluation of uncertainty estimates and their robustness.

Part II

Error prediction in pixel-wise regression tasks using an auxiliary network

Chapter 3

SLURP: Side Learning Uncertainty for Regression Problems

Contents

3.1 Introduction

Side learners [41, 107], also called auxiliary networks [42] or meta-models [212], are a kind of training-based post-hoc approach for uncertainty quantification. A side learner could be regarded as a post-processing operation applied to the main task model. 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, such as monocular depth estimation and optical flow estimation, 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.

SLURP is not only a specific network architecture facing regression tasks but also a general novel idea to solve uncertainty quantification problems for pixel-wise tasks. More specifically, the previous design only uses the single brunch, taking either the image or the intermediate features of the main task model as the input of the auxiliary networks. We use both the input and the output of the main task model to feed the auxiliary network in SLURP, which is proven more suitable for pixel-wise tasks in the experiments.

Contributions

1. With SLURP, we were the first to solve the uncertainty estimation problem for general pixel-wise regression tasks with an auxiliary network. Since then, multiple works such as [197] have proposed modifications and alternative strategies and extended the SLURP idea.
2. We proposed a transposable architecture that 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.

3.2 Related Works

Apart from the major uncertainty quantification approaches, we here provide a quick review of the methods related to side learners.

Side learning for uncertainty estimation

With the development of deep learning, DNNs are becoming more sophisticated, and the training of DNNs and the selection of hyperparameters have also become essential, such as in optical flow estimation task [220]. As it avoids changing the main task model structure, a side learner is, in this respect, a convenient approach to estimating uncertainty.

Most previous works are applied initially to image-level tasks. Yoo et al. [250] design a side learner for learning the loss supplied by the main task model. However, this work aims to provide a single predicted loss for the main task, which is unsuitable for pixel-wise uncertainty estimation. DEUP [107] not only uses a side learner to predict the loss but to obtain the epistemic uncertainty better, 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.

Like SLURP, which is mainly oriented to or can be extended to pixel-wise tasks, Lee et al. [132] use a conditional GAN [106] as an auxiliary model. The conditional GAN learns to project the images to the uncertainty map provided by MC-Dropout [70] on the main task model. The goal is to reduce the time cost of the MC-Dropout sampling, but its performance can be limited to the uncertainty quantification quality given by MC-Dropout. Based on [123] and [113], Hu et al. [95] trains sequentially a network identical to the main task model to fit the prediction error of the main task model ensembles. 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. Corbière et al. [41] introduces ConfidNet, which can conveniently provide the uncertainty by adding an auxiliary network after the main task encoder. It shows a good performance, but it initially works only for classification tasks and takes only

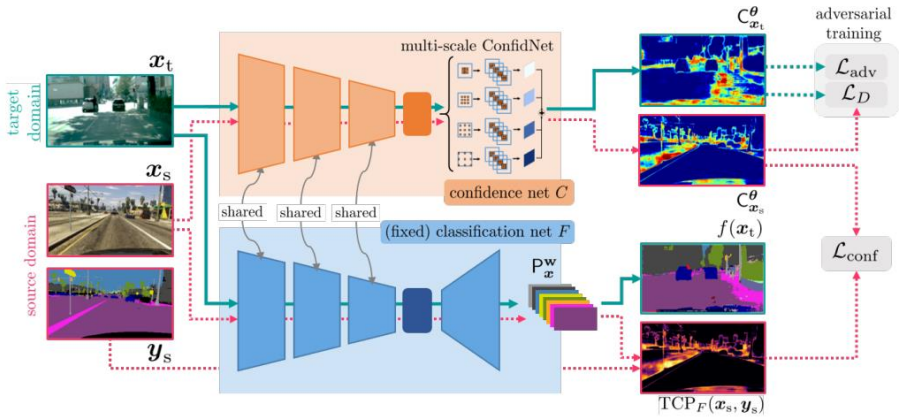


Figure 3.1: Multi-scale ConfidNet applied in domain adaptation for semantic segmentation.

Figure 3.2: An example for the uncertainty quantification in optical flow task.

the image as the input. 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. The follow-up work, published later on [42], extended ConfidNet to domain adaptation tasks. In domain adaptation for semantic segmentation, the authors propose the concept of multi-scale ConfidNet as shown in Fig 3.1, which is similar to SLURP. Yet, our work still differs, such as feeding outputs of pixel-wise tasks into an auxiliary network.

The number of data points is vast for pixel-wise regression tasks, and data preparation will be costly. SLURP does not need to modify the main task model and does not require data preparation. It faces general pixel-wise regression tasks, and differs 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 training codes. Using only the in-distribution data, we can get better quality and robust uncertainty without touching the main task model. Fig. 3.2 provides an illustration of the uncertainty quantification of our work. The predicted flow is made by FlowNetS [54]. The prediction error is the end-point-error between ground truth flow and predicted flow. Our uncertainty quantification 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.

3.3 Auxiliary uncertainty estimator SLURP

3.3.1 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 the second stage, the parameters in f are fixed and will not be updated anymore. Based on a primary DNN f with trained parameters $\hat{\boldsymbol{\omega}}$, we train a second DNN σ with trainable parameters Θ 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

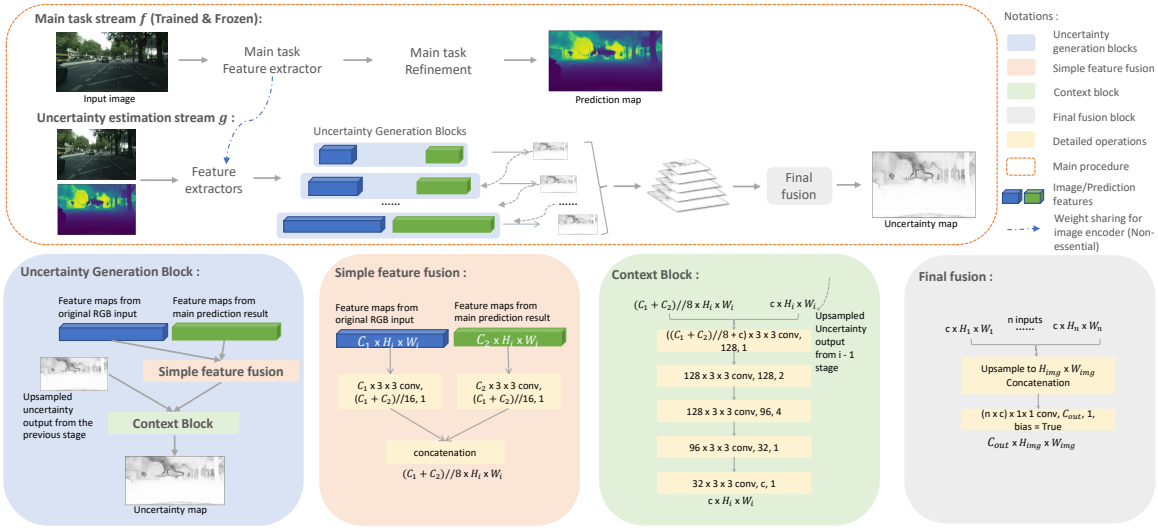


Figure 3.3: General solution for pixel-wise uncertainty estimation. We take monocular depth estimation as an example. The convolutional layers are described as (shape, number, and dilation ratio).

trained by minimizing the objective function $\mathscr{L}_{\boldsymbol{\omega}}(f_{\boldsymbol{\omega}}(\mathbf{x}), y)$ over the dataset.

Task 2 Once the DNN f is trained and $\hat{\boldsymbol{\omega}}$ is obtained, 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 $\mathscr{L}_{\hat{\boldsymbol{\omega}}}(f_{\hat{\boldsymbol{\omega}}}(\mathbf{x}), y)$. Note that the loss does not need to be the same as $\mathscr{L}_{\hat{\boldsymbol{\omega}}}$, since in some cases, $\mathscr{L}_{\hat{\boldsymbol{\omega}}}$ could follow a specific design such as in focal loss [142], in scale-invariant error [57], etc. Hence we aim to predict the error loss $\mathscr{L}_u(f_{\hat{\boldsymbol{\omega}}}(\mathbf{x}), y)$. As is the case with regression tasks, a sensible choice for $\mathscr{L}_u$ is the mean square or absolute error. Our uncertainty DNN σ with trainable parameters Θ, will have the following training objective:

$$\underset{\Theta}{\operatorname{argmin}}\, \mathscr{L}\Big(\mathscr{L}_u(f_{\hat{\boldsymbol{\omega}}}(\mathbf{x}), y), \sigma_\Theta(f_{\hat{\boldsymbol{\omega}}}(\mathbf{x}), \mathbf{x})\Big) \tag{3.1}$$

where $\mathscr{L}$ denotes an objective function that 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 $\mathrm{U}(f, \mathbf{x})$ may then be interpreted as the sum of prediction errors according to [107]:

$$\mathrm{U}(f_{\hat{\boldsymbol{\omega}}}, \mathbf{x}) = \int \mathscr{L}_u(f_{\hat{\boldsymbol{\omega}}}(\mathbf{x}), y)\, d\mathrm{P}(y|\mathbf{x}) \tag{3.2}$$

In subsection 3.3.2, we explain the structure of the side learner σ, and in subsection 3.3.3, we present the design of the loss $\mathscr{L}$ to learn to predict the uncertainty.

3.3.2 Side learner's architecture

The side learner σ_Θ takes two inputs: $\mathbf{x}$ and $f_{\hat{\boldsymbol{\omega}}}(\mathbf{x})$. The combination of image features and prediction results in uncertainty maps 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 by the following two properties:

1. Edges of connected domains in the prediction map;
2. Hard predictable areas that 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 [148]. The concatenated convolutional features are followed by a context block [219].

We structure the side learner σ in three parts described as follows:

1. Feature encoders The feature encoders aim to learn and extract the richer convolutional feature pyramids [148] from raw input (RGB data) and from the final prediction map preparing for the next steps. We choose the widely used backbone DenseNet [100]. Note that the architecture is agnostic, and DenseNet could be replaced by the other backbones like ResNet [87], depending on the context. The image encoder in σ 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 by using feature extractors with similar capabilities. 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 caused by focusing on specific patterns. The goal is to find the similarities and the differences 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 [219]. With various dilation ratios, the convolutional layers can take the features from different sizes 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 the 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.

Figure 3.3 shows the general design for our pixel-wise uncertainty estimator. As we can see from the figure, in the main procedure, **Main task stream f is first trained and frozen.** Then we use the input and the output of $f_{\hat{\omega}}$ to train σ_Θ. 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.

3.3.3 Loss design

1. Natural loss A straightforward way is to use mean square error loss (MSE) for $\mathscr{L}_u$ in Eq. 3.1 and Eq. 3.2. Since the main task model f is trained, most of the predictions on the training dataset will have smaller prediction errors, which is also stated in the later work [229]. In order to prevent the distribution of the training target, i.e., the prediction errors, from being too sparse, we take absolute error $\mathscr{L}_u(f_{\hat{\omega}}(\mathbf{x}), y) = |f_{\hat{\omega}}(\mathbf{x}) - y|$.

2. Target scaling The uncertainty of the regression task is expressed across 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 of $[0, 1]$. Hence we process $\mathscr{L}_u(f_{\hat{\omega}}(\mathbf{x}), y)$ 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:

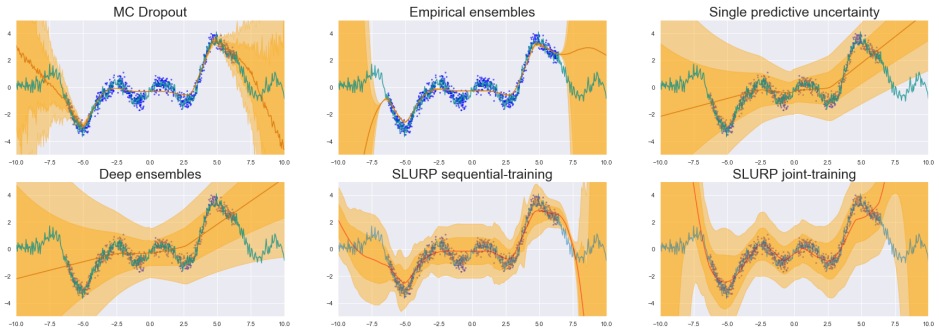


Figure 3.4: 1D synthetic regression task comparison example. X-axis: spatial coordinate of the Gaussian process. Green curve: ground truth; Blue points: training samples; Red curves: main task prediction; Orange zooms: the uncertainty coverage (1-sigma for inner interval, 2-sigma for outer interval).

$$\tilde{\mathscr{L}}_u(f_{\hat{\omega}}(\mathbf{x}), y) = \tanh\left(\lambda * \mathscr{L}_u(f_{\hat{\omega}}(\mathbf{x}), y)\right) \tag{3.3}$$

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

3. Cross-Entropy Loss After having normalized the uncertainty $\mathscr{L}_u$, we select cross-entropy loss to describe the probability distance between training target $\tilde{\mathscr{L}}_u$ and our predicted uncertainty. Since $\tilde{\mathscr{L}}_u \in [0, 1]$, we choose a binary cross entropy (BCE) loss as $\mathscr{L}$ in Eq. 3.1:

$$\begin{aligned}\mathscr{L}(\Theta) = -\sum_{i=0}^{N} \Big[& \tilde{\mathscr{L}}_u\left(f_{\hat{\omega}}(\mathbf{x}_i), y_i\right) \cdot \log\left(\delta(\sigma_\Theta(f_{\hat{\omega}}(\mathbf{x}_i), \mathbf{x}_i))\right) \\ & + \left(1 - \tilde{\mathscr{L}}_u(f_{\hat{\omega}}(\mathbf{x}_i), y_i)\right) \cdot \log\left(1 - \delta(\sigma_\Theta(f_{\hat{\omega}}(\mathbf{x}_i), \mathbf{x}_i))\right) \Big]\end{aligned} \tag{3.4}$$

where $\delta(.)$ denotes 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. 3.4 with respect to $\delta(\sigma_\Theta)$, we can find that the optimum will be $\delta(\sigma_\Theta) = \tilde{\mathscr{L}}_u$.

3.4 Experiments

Our main focus is on obtaining high-quality uncertainty maps on pixel-level regression tasks. However, in order to illustrate our approach more comprehensively 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 and monocular depth estimation. 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) [70], Single predictive uncertainty (Single-PU) [113], Empirical ensembles (EEns.), ConfidNet (Confid) [41] and Deep ensembles (DEns.) [123] to the main task as the comparisons. Due to its particularity, we reproduce the multi-hypothesis prediction network (MHP) [104] only for the optical flow task. Specifically, to transfer the 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 the main task + uncertainty estimator system from two perspectives:

Hyper-parameters	MC	EEns.	DEns. (Single-PU)	SLURP joint-training	SLURP sequential-training
number of main task latent features	3000				
learning rate for main task model	1e-1	1e-1	1e-2	1e-1	/
learning rate for side learner feature extractor	/	/	/	1e-1	1e-4
learning rate for side learner uncertainty generation blocks	/	/	/	1e-4	1e-4
batch size	50				
number of training epoch	50				
weight decay for main task model	1e-2	1e-2	1e-2	1e-2	/
weight decay for side learner feature extractor	/	/	/	1e-2	1e-3
weight decay for side learner uncertainty generation blocks	/	/	/	1e-3	1e-2
Model structure and other settings					
loss	MSE	MSE	Gaussian NLL	Gaussian NLL	MSE
dropout rate	0.4	/	/	/	/
ensemble size M	1	3	3 (1)	1	1

Table 3.1: 1D regression task model settings.

Runtime and Number of parameters. The specific formulas as well as the implementation details for the methods we compare with, are provided in the Sec. 3.5.

3.4.1 1D regression task toy example

We compare our proposed solution with MC [70], EEns., Single-PU [113] and DEns. [123] on a 1D regression task dataset. The toy dataset is generated by the Gaussian process. Our spatial coordinate range is $x_i \in [-10, 10]$. For generating y_i, we first define the RBF kernel and compute the covariance matrix associated with inputs $\mathbf{x}$. Then, we generate separate samples from a Gaussian with mean 0 and covariance. From $x_i \in [-7, 7]$, we cross-select 875 data points as the training set and 175 data points as the validation set. From $x_i \in [-10, 10]$, we randomly select 400 points as the test set. The main task predictor f with single output has only one hidden layer, the Single-PU, and DEns. are implemented based on a modified dual-output f.

The detailed model and training settings for all the methods are detailed as follows. Because of the simplicity of the data, the main task predictor we use is a neural network composed of one hidden layer and 3000 neurons. 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 results 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 [113], i.e., joint training. Because of the data dimension, the side learner shown in Fig 3.3 is modified. For the side learner, following the general SLURP solution, we use the same hidden layer as the prediction result feature extractor and three hidden layers with 128, 64, and 16 neurons, respectively, as the context block in the uncertainty generation block, since we have only one stage, there is no fusion block in the end. The training details for all uncertainty estimation approaches are listed in Table 3.1.

As illustrated in Fig 3.4, the results of MC [70] and EEns. give a good uncertainty on the unseen area but little on the training part. While DEns. [123] and Single-PU [113] can give 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 the joint-training one. We can give an insight that the SLURP strategy can also work on 1D-regression tasks. In addition, the structure of the SLURP side learner is variable, and other uncertainty estimation methods are limited to the structure of the main task model.

3.4.2 Evaluation protocols for pixel-wise tasks

monocular depth estimation and optical flow are two fundamental regression tasks that have significant implications in a wide range of applications. In terms of inputs, the main difference is that monocular depth estimation requires a single RGB image, while optical flow requires an image pair. For each pixel, the monocular depth estimation output is a depth value $y_d \in \mathscr{R}^+$, while the optical flow output is a 2-channel displacement vector $y_o \in \mathscr{R}^2$. We introduce below the evaluation criteria.

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)** [23, 104, 116, 122, 191]. 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 that 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 estimation, we choose square error and absolute relative error [57]. 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**.

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 monocular depth estimation to set the data as reliable if they check the inlier metrics threshold d1 criterion proposed in [98], and for optical flow 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.

Monocular depth estimation evaluation metrics Let us consider a monocular depth dataset $\mathrm{D} = \{(\mathbf{x}_i, d_i)\}_i$ where $d_i \in \mathscr{R}^+$ is the ground truth depth value for pixel $\mathbf{x}_i$. Below, $\hat{d}_i$ represents the depth prediction. The metrics we used in the evaluations are as follows:

- Root mean square error (RMSE)

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\mathrm{D}|} \sum_{d_i \in |\mathrm{D}|} ||\hat{d}_i - d_i||^2} \tag{3.5}$$

- Absolute relative error (Absrel)

$$Absrel = \frac{1}{|\mathrm{D}|} \sum_{d_i \in \mathrm{D}} |\hat{d}_i - d_i| / d_i \tag{3.6}$$

- Threshold δk
  Inlier metrics as proposed in [57], k in δk indicates the power of the threshold (thr), we take

$thr = 1.25$. In this case, δ1: $thr = 1.25$; δ2: $thr = 1.25^2$; δ3: $thr = 1.25^3$, and δk represents the proportion of pixels that meet the threshold condition:

$$\delta k = max(\frac{\hat{d}_i}{d_i}, \frac{d_i}{\hat{d}_i}) = \delta < thr^k \tag{3.7}$$

$$\delta k = \frac{|\mathrm{A}|}{|\mathrm{D}|}, \text{where A} = \left\{ \mathbf{x}_i, \text{such that } \delta_i = \max(\frac{\hat{d}_i}{d_i}, \frac{d_i}{\hat{d}_i}) \text{ and } \delta_i < thr^k \right\} \tag{3.8}$$

- Squared relative difference (SqRel)

$$\mathrm{S}qRel = \frac{1}{|\mathrm{D}|} \sum_{d_i \in \mathrm{D}} ||\hat{d}_i - d_i||^2 / d_i \tag{3.9}$$

- Root mean square log error (RMSElog)

$$\mathrm{RMSE}log = \sqrt{\frac{1}{|\mathrm{D}|} \sum_{d_i \in |\mathrm{D}|} ||\log \hat{d}_i - \log d_i||^2} \tag{3.10}$$

- Average log10 error (log10)

$$log10 = \frac{1}{|\mathrm{D}|} \sum_{d_i \in |\mathrm{D}|} |\log_{10} \hat{d}_i - \log_{10} d_i| \tag{3.11}$$

These six metrics measure the performance of the MDE models we use and are shown in Table 3.4. Additionally, for AUROC, we choose $k = 1$ when applying threshold δk metric.

We will keep these metrics in the next sections and parts for evaluating depth estimates.

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.

3.4.3 Monocular depth estimation

In this section, we introduce uncertainty estimations for monocular depth estimation. We choose BTS [130] 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 monocular depth estimation benchmarks [74, 213]. As an encoder-decoder-based network, it is well suited for the extraction of latent image features, as shown in Fig 3.3. However, note that our strategy is agnostic to the main task architecture. In accordance with the default setting, we choose DenseNet161 [100] 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 the 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 [74, 227] Eigen-split training set [57]; *Fine-tuning set:* Cityscapes training set [43]; *Test set:* KITTI Eigen-split test set, Cityscapes test set, Foggy Cityscapes-DBF test set [205] with three severity levels and Rainy Cityscapes test set [97] with three parameter sets which indicate three severities. The KITTI depth dataset and the 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 models on the KITTI Eigen-split training set and then evaluate the uncertainty on the KITTI Eigen-split test set and the Cityscapes test set. After that, we fine-tune the models on the Cityscapes training set and evaluate the uncertainty on all Cityscapes test sets listed above.

Training settings In the monocular depth estimation task, we choose to use a sequential training strategy for single predicted uncertainty (Single-PU) [113], deep ensembles (DEns.) [123] and our SLURP side learner. In other words, we first complete the training of the main task models (BTS [130]) and then train different uncertainty predictors according to the settings. Specifically, for Single-PU, we use an identical BTS model to estimate the uncertainty by using the output of the main task and its corresponding main task model in the loss. For DEns., it is a mixture of multiple Single-PUs, so we just repeat the previous procedures. Because of the ensemble property of empirical ensembles (EEns.) and DEns., EEns. and DEns. can share the same main task predictors. In the same sense, the main task predictor of our side learner is chosen from one of EEns. (DEns.)'s main task predictors, which is the same one as the main task model of Single-PU. This method can ensure that the prediction accuracy of the main task will not be affected by the training of the uncertainty predictor. The ConfidNet [41] (Confid) implementation for BTS references its implementation on SegNet. We build our side learner according to the SLURP solution shown in Fig 3.3 for BTS, and here are some supplements. We directly use the frozen RGB feature maps from the encoder of the main task BTS model. To convert a 1-channel predicted depth map to a 3-channel input, we expand it three times. The detailed settings for different uncertainty estimation models are listed in Table 3.2. All main task models are trained identically according to the original BTS [130] model training settings.

Uncertainty quantification and main task results Table 3.3 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.
Table 3.4 shows the main task model performance for different uncertainty estimation approaches [70, 113, 123]. We have noticed that after the model is trained on KITTI, it cannot obtain reasonable accuracy on Cityscapes. This is because the ground truth of the KITTI dataset is sparse, and only the lower half of the content is present. At the same time, the scene of Cityscapes is more complicated. Therefore, we fine-tune all models on Cityscapes to obtain reasonable accuracy.

3.4.4 Optical flow

FlowNetS [54] is a popular optical flow 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 [103, 105, 191, 219, 245], 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 [100] 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 [54]; *Testing set:* FlyingChairs test set, synthetic Sintel training set [24] and real-world KITTI 2015 training set [74, 165, 166]. The train-test split file for the FlyingChairs dataset is provided officially. Sintel has more complex moving objects and movements than FlyingChairs, while KITTI is taken from real-world, and exhibits larger movement magnitude.

Hyper-parameters	MC	EEns.	DEns. (Single-PU, Confid)	Ours
learning rate for main task model (Training on KITTI)	1e-4	1e-4	/	/
number of training epoch (Training on KITTI)	50	50	50	8
learning rate for side learner (Training on KITTI)	/	/	/	1e-4
learning rate for main task model (Fine-tuning on Cityscapes)	5e-5	5e-5	/	/
number of training epoch (Fine-tuning on Cityscapes)	30	30	30	16
learning rate for side learner (Fine-tuning on Cityscapes)	/	/	/	8e-5
learning rate for identical uncertainty estimator	/	/	5e-5	/
learning rate for side learner	/	/	/	1e-4
batch size	4			
number of training epoch	50	50	50	8
weight decay for main task model	1e-2	1e-2	/	/
weight decay for identical uncertainty estimator	/	/	1e-2	/
weight decay for side learner feature extractor	/	/	/	1e-3
weight decay for side learner uncertainty generation blocks	/	/	/	4e-4
Model structure and other settings				
encoder backbone for main task model, identical uncertainty estimator and side learner feature extractor	Densenet 161 [100]			
loss	same as BTS [100]	same as BTS [100]	Laplacian NLL	BCE $\lambda = 0.0125$
number of latent stages n	/	/	/	5
number of latent stage output channel c	/	/	/	1
number of final uncertainty output channel C_{out}	/	/	/	1
dropout rate p_d	0.4	/	/	/
ensemble size M	1	3	3 (1)	1
during inference time number of forward propagation	8	3	3 (1)	1

Table 3.2: monocular depth estimation model settings for MC, EEns., DEns., Single-PU, Confid and Ours.

Training settings In the optical flow task, for EEns., we directly train multiple main task prediction models FlowNetS [54], and our side learner selects one of the models as our main task predictor. For Single-PU, because FlowNetS is relatively simple, we directly modify the original model to output two values for each pixel, one representing the predicted value of the main task and the other the uncertainty value. For DEns., we train multiple Single-PU models. For multi-hypothesis prediction network (MHP) [104], we modified FlowNetS so that it can output eight (number of hypotheses) pairs of main task-uncertainty results. Furthermore, we use another FlowNetS as the MergeNet. It should be noted that the authors did not mention the structural information about MergeNet in the paper. We choose FlowNetS based on the use of model stacking in the article.
For our SLURP side learner, since the encoder in FlowNetS is designed for capturing the object movement for two images and the total uncertainty will reflect only the semantics from the first

Datasets	Criteria	MC	EEns.	Single PU	DEns.	Confid	Ours
KITTI	AUSE-RMSE	8.14	3.17	1.89	**1.68**	1.76	**1.68**
	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
CityScapes	AUSE-RMSE	**9.42**	11.56	9.91	11.47	10.48	9.48
	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							
CityScapes	AUSE-RMSE	7.72	8.20	4.35	**3.03**	4.05	3.05
	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 3.3: monocular depth estimation uncertainty performance. Bold value: result with the best performance, Blue value: second performance. s (*e.g* s=1) indicates severity, the higher the s value, the higher the severity.

image, we use two DenseNet161 backbones [100] as RGB and prediction map encoders respectively. We also used two DenseNet121 backbones for the lighter version. In order to transfer the 2-channel flow prediction to a 3-channel input, we just add one convolution layer to expand the channel number before the RGB feature extractor. All uncertainty model training settings are shown in Table 3.6.

Uncertainty quantification and main task results Table 3.5 shows the uncertainty estimation performance on the various optical flow 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.
Table 3.7 shows the main task precision for different uncertainty estimation strategies. Our SLURP

Models	Datasets	higher is better			lower is better				
		δ1 ↑	δ2 ↑	δ3 ↑	AbsRel ↓	SqRel ↓	RMSE ↓	RMSElog ↓	log10 ↓
MC	KITTI	0.945	0.992	0.998	0.072	0.287	2.902	0.107	0.031
	Cityscapes	0.103	0.255	0.453	1.051	18.942	18.986	0.842	0.324
EEns. (DEns.)	KITTI	0.957	0.993	0.999	0.059	0.233	2.688	0.093	0.026
	Cityscapes	0.214	0.430	0.560	0.837	14.459	18.441	0.845	0.298
Ours	KITTI	0.955	0.993	0.998	0.060	0.249	2.798	0.096	0.027
(Single-PU, Original [130])	Cityscapes	0.183	0.386	0.519	0.963	17.230	18.948	0.896	0.321
After fine-tuning on Cityscapes									
MC	Cityscapes	0.882	0.974	0.992	0.117	0.917	5.625	0.169	0.049
EEns. (DEns.)	Cityscapes	0.920	0.983	0.995	0.098	0.635	4.889	0.149	0.043
Ours (Single-PU)	Cityscapes	0.906	0.980	0.993	0.104	0.711	5.216	0.159	0.046

Table 3.4: The performance on KITTI and the performance on the Cityscapes dataset before and after fine-tuning the models. Before fine-tuning: Training set: KITTI Eigen-split training set, Test set: KITTI Eigen-split test set and Cityscapes test set; After fine-tuning: Fine-tuning training set: Cityscapes training set, Test set: Cityscapes test set. The three main task models used in EEns. are also used in DEns. and Single-PU also shares the same main task model with our approach.

Datasets	Criteria	MC	EEns.	Single PU	DEns.	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 3.5: optical flow uncertainty performance. Bold value: result with the best performance. Blue value: second performance.

side learner picks one of the models from EEns. as our main task predictor. The main task models are trained only on FlyingChairs training set with official split, and the KITTI dataset we choose for main task precision evaluation and also uncertainty estimation/evaluation is KITTI 2015 with occlusions. In the original FlowNetS paper [54], the precision evaluation is based on KITTI 2012 [76].

3.4.5 Model efficiency

Datasets and settings We select to use Sintel training set [24] and KITTI [74] Eigen-split validation set [57] as the testing sets for Runtime evaluation. We use three models to form EEns. and DEns. and eight forward propagations for MC. In monocular depth estimation, due to the instability of training, we use sequential training for Single-PU and DEns.

Results Table 3.8 shows the model efficiency in two tasks. In the optical flow 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 monocular depth estimation, due to the complexity of implementation for joint uncertainty generation, the side-learner-based methods have a fixed computational footprint and take significant advantage of the cost. In addition, our solution can be reused without re-tuning the main task encoder, so it is lighter and faster than the other auxiliary networks.

Hyper-parameters	MC	EEns.	DEns. (Single-PU, Confid)	Ours
learning rate for (modified) main task model	1e-4	1e-4	1e-4	/
learning rate for side learner	/	/	/	1e-4
batch size			8	
number of training epoch	216	216	216	30
weight decay for (modified) main task model	4e-4	4e-4	4e-4	/
weight decay for side learner feature extractors	/	/	/	1e-4
weight decay for side learner uncertainty generation blocks	/	/	/	4e-4
Model structure and other settings				
loss	same as FlowNetS [54]	same as FlowNetS [54]	Laplacian NLL	BCE $\lambda = 0.05$
number of latent stages n	/	/	/	5
number of latent stage output channel c	/	/	/	2
number of final uncertainty output channel C_{out}	/	/	/	1
dropout rate p_d	0.4	/	/	/
ensemble size M	1	3	3 (1)	1
during inference time number of forward propagation	8	3	3 (1)	1

Table 3.6: optical flow model settings for MC, EEns., DEns., Single-PU, Confid, MHP and Ours. MHP training setting is followed by the same schedule provided by its original paper.

Datasets	EEns.	MC	Single-PU	DEns.	Ours	Orginal [54]
FlyingChairs test	1.79	3.71	2.04	1.93	1.96	2.71
KITTI 2015 occ	18.36	16.53	21.21	20.78	19.39	/
KITTI 2012 noc	6.77	19.02	8.34	8.40	7.65	8.26
Sintel clean train	5.10	6.31	5.12	5.00	5.20	4.50
Sintel final train	6.50	6.97	6.53	6.41	6.62	5.45

Table 3.7: The main task accuracy for the uncertainty estimators in optical flow task. The values present the end-point error (EPE). Training set: FlyingChairs training set [54], Test set: FlyingChairs test set, KITTI 2012 noc [76] which was used in the original FlowNetS paper, KITTI 2015 occ [74, 165, 166] and Sintel full training set [24]

.

Task	Criteria	MC	EEns.	Single-PU	DEns.	Confid	MHP	Ours-light	Ours
monocular depth estimation	Runtime (ms)	386	144	98	286	106	-	-	**88**
	# Param. (M)	**47.0**	141.0	94.0	282.0	94.7	-	-	87.2
optical flow	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 3.8: 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.

3.5 Ablation study

As a goal, we want to highlight the impact of the two considered inputs on the final performance (namely the image features and the prediction results features) and the impact of the considered loss (binary cross entropy loss and mean square error loss).

Conditions					
Input source	RGB Input	✓	✗	✓	✓
	Prediction map Input	✗	✓	✓	✓
Loss	MSE	✗	✗	✓	✗
	BCE	✓	✓	✗	✓
Datasets	Criteria	Ours RGBOnly	Ours PredOnly	Ours MSE	Ours BCE
FlyingChairs	AUSE-EPE	1.82	1.24	1.41	**1.20**
	AUROC	0.944	0.972	0.967	**0.974**
KITTI	AUSE-EPE	8.40	4.87	5.40	**4.69**
	AUROC	0.586	**0.800**	0.793	**0.800**
Sintel Clean	AUSE-EPE	7.43	**2.73**	3.19	2.91
	AUROC	0.639	**0.898**	0.883	0.896
Sintel Final	AUSE-EPE	8.24	**2.71**	3.11	2.86
	AUROC	0.575	**0.907**	0.889	0.906

Table 3.9: Ablation study for the optical flow task. Bold value: result with the best performance. Blue value: second performance.

Settings The ablation study for the monocular depth estimation task is implemented on KITTI [74, 227] Eigen-split test set [57], Cityscapes test set [43], Foggy Cityscapes-DBF test set [205] and Rainy Cityscapes test set [97]. The ablation study for the optical flow (optical flow) task is implemented on FlyingChairs test set [54], Sintel training set [24] and KITTI 2015 training set [74, 165, 166]. We use the same uncertainty evaluation metrics (AUSE and AUROC) as in Section 3.4.2.

Results The models with different inputs and different loss functions are presented as follows. Table 3.9 presents the model performance on the optical flow task, and Table 3.10 illustrates the results of the monocular depth estimation task. Note that in the tables BCE and MSE denote binary cross entropy loss and mean square error loss, respectively, PredOnly and RGBOnly denote the models taking only prediction map as input and the models taking only RGB image as input, respectively. No special note means that the model will use both RGB and prediction results as input and BCE as the loss function (the default behavior).

Discussions Firstly, regarding the performance of the different loss functions, we found that the results obtained with the BCE loss are almost systematically better than those provided when using MSE loss. We think this is because when we have a correctly trained predictor for the main task, most of the data points have minor errors, while a small number of data points have high errors. Using the MSE loss will amplify the more significant prediction errors and reduce the minor errors, making the model unable to fit well. Our target scaling uses a soft clipping strategy to centralize the distribution of data for better fitting.

For different inputs, we found that it is essential to use the prediction map as the input through the evaluation results. The input of the RGB image sometimes affects the generalization ability of uncertainty estimation if the main task model can generate already very good prediction results. According to the visualizations and evaluation results, we can see that the influence of the input of the prediction map is dominant because the uncertainty map of dual input and the one with only the prediction map input are similar. On the other hand, the RGB image can supplement some missing semantics of the prediction map, such as the Fig 3.5 FlyingChairs where RGB input can supplement the lack of chair legs and in the Fig 3.6 where RGB input supplement the uncertainty of the sky (although the sky does not have ground truth of depth, it should have a high degree of uncertainty).

Conditions					
Input source	RGB Input	✓	✗	✓	✓
	Predinction map Input	✗	✓	✓	✓
Loss	MSE	✗	✗	✓	✗
	BCE	✓	✓	✗	✓
Dataset	Criteria	Ours RGBOnly	Ours PredOnly	Ours MSE	Ours BCE
KITTI	AUSE-RMSE	1.84	1.76	1.74	**1.68**
	AUSE-Absrel	4.45	4.31	**4.19**	4.36
	AUROC	0.879	0.890	0.894	**0.895**
Cityscapes	AUSE-RMSE	9.95	**9.40**	9.82	9.48
	AUSE-Absrel	10.68	**9.23**	10.29	10.90
	AUROC	0.344	**0.446**	0.414	0.400
After fine-tuning on Cityscapes					
Cityscapes	AUSE-RMSE	3.47	3.45	4.77	**3.05**
	AUSE-Absrel	6.71	**6.47**	6.93	6.55
	AUROC	0.837	0.844	0.766	**0.849**
Cityscapes Rainy s=1	AUSE-RMSE	3.97	3.43	4.80	**3.39**
	AUSE-Absrel	6.95	**5.52**	7.30	5.62
	AUROC	0.739	**0.795**	0.68	0.788
Cityscapes Rainy s=2	AUSE-RMSE	3.98	3.39	4.92	**3.36**
	AUSE-Absrel	6.68	**5.16**	7.09	5.28
	AUROC	0.747	**0.801**	0.689	0.794
Cityscapes Rainy s=3	AUSE-RMSE	4.11	**3.41**	5.07	**3.41**
	AUSE-Absrel	6.77	**4.85**	7.06	5.05
	AUROC	0.748	**0.811**	0.694	0.801
Cityscapes Foggy s=1	AUSE-RMSE	3.55	3.42	4.92	**3.04**
	AUSE-Absrel	6.40	**6.15**	6.92	6.25
	AUROC	0.835	0.841	0.763	**0.847**
Cityscapes Foggy s=2	AUSE-RMSE	3.51	3.39	5.03	**3.01**
	AUSE-Absrel	6.23	**5.98**	6.89	6.06
	AUROC	0.838	0.845	0.767	**0.852**
Cityscapes Foggy s=3	AUSE-RMSE	3.48	3.36	5.24	**3.08**
	AUSE-Absrel	5.97	**5.72**	6.75	5.80
	AUROC	0.845	0.852	0.773	**0.857**

Table 3.10: Ablation study for the monocular depth estimation task. Bold value: result with the best performance. Blue value: second performance. s (*e.g* s=1) indicates severity, the higher the s value, the higher the severity.

Figure 3.5: Uncertainty estimation examples in ablation study for optical flow task. The first row of each dataset block represents the input image pair, ground truth and predicted optical flow, and the prediction error. The prediction map and error map are made by a single FlowNetS model as an example. The second row of each dataset block represents the uncertainty results in using the SLURP side learner with different inputs and different loss functions. Black indicates higher uncertainty, and white indicates lower uncertainty.

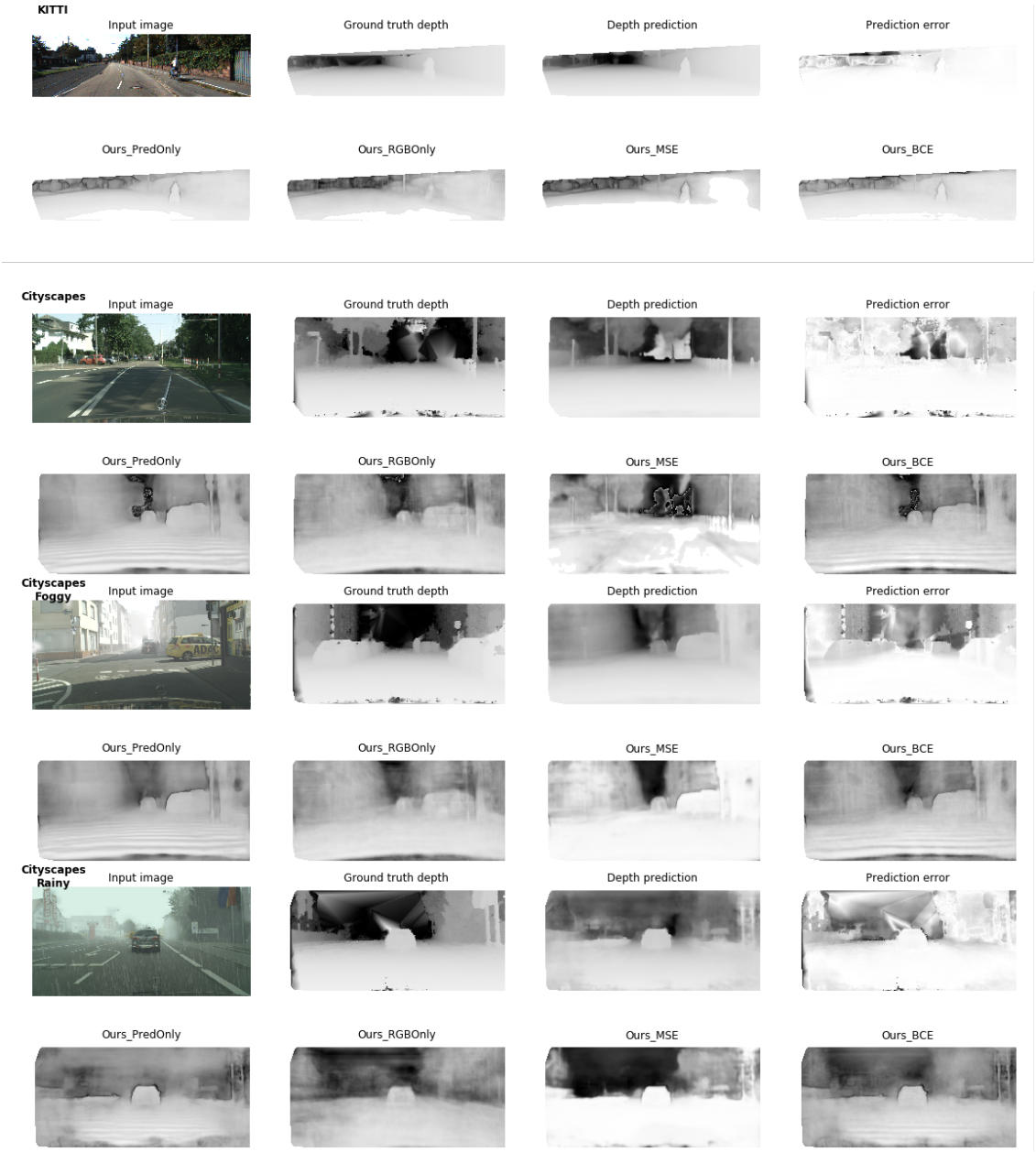


Figure 3.6: Uncertainty estimation examples in ablation study for monocular depth estimation task. The first row of each dataset block represents the input image, ground truth depth map, depth prediction map, and the prediction error. Since the ground truth is sparse, we use interpolation to rebuild the ground truth map just for visualization. The predicted depth map and error map are made by a single BTS model as an example. The second row of each dataset block represents the uncertainty results in using the SLURP side learner with different inputs and different loss functions. For uncertainty maps, black indicates higher uncertainty, and white indicates lower uncertainty. For depth maps, black represents deeper depth, and white represents shallower depth.

3.6 More visualization

In this section, we provide more visualization results. For monocular depth estimation, as shown in 3.7, the depth prediction map and the error map are generated by a single BTS model as an example. MC-Dropout uncertainty maps are obtained by eight forward propagation, Deep ensembles and Empirical ensembles uncertainty maps are obtained from three model ensembles. The results for optical flow are shown in Figure 3.8. The optical flow prediction maps and the error maps are generated by a single FlowNetS model as an example. MC-Dropout uncertainty maps are obtained by eight forward propagation, Deep ensembles, and Empirical ensembles uncertainty maps are obtained from three models ensembles.

Figure 3.7: Uncertainty estimation results for monocular depth estimation task. The ground truth maps are rebuilt by interpolation just for visualization. For uncertainty maps, black indicates higher uncertainty, and white indicates lower uncertainty. For depth maps, black represents deeper depth, and white represents shallower depth.

Figure 3.8: Uncertainty estimation results for optical flow task. For uncertainty maps, black indicates higher uncertainty, and white indicates lower uncertainty.

3.7 Conclusion

In this work, we introduced SLURP, a learning-based side learner for uncertainty quantification in regression problems, especially on pixel-wise regression tasks. It is a post-hoc solution in which the main task model will not be modified or re-trained. The network architecture design is based on the observations on the prediction error maps. In the experiment, we discover that using both images and the main task predictions as the inputs of the auxiliary network is more effective and can be a general concept in pixel-wise uncertainty quantification tasks.

However, although the prediction errors can be regarded as the total uncertainty in the training set, SLURP has no guarantee to provide higher uncertainty to the rarely seen or even unseen patterns. This is because, first, the training dataset only provides the limited cases of patterns that have higher prediction errors, and SLURP will only learn these patterns and generalize them to similar cases. However, when there is a bigger dataset shift, for instance, from the outdoor training scenario to the unseen indoor one, SLURP has a risk of missing the alert. With limited data resources, the uncertainty estimators should explicitly better model the epistemic uncertainty to provide higher uncertainty to the out-of-distribution patterns.

In the next chapter, we will focus on how to better model the aleatoric and epistemic uncertainty, respectively, and introduce a general solution based on the auxiliary uncertainty estimators. Specifically, we will introduce a new epistemic uncertainty quantification approach called Discretization-Induced Dirichlet pOsterior (DIDO). Before introducing the general uncertainty estimator and DIDO, we will first present a pilot work that discusses how to use discretization methods in regression tasks.

Part III

Introducing classification into uncertainty quantification problems in regression

Chapter 4

Classification approach for regression in monocular depth estimation

Contents

4.1 Introduction

In machine learning, regression tasks predict a continuous output based on a given input. Yet, if the ground truth (prediction target) is within a specific range, *e.g.,* in the case of age estimation [134], one can quantize the ground truth and cast regression into a classification problem. As an example, monocular depth estimation (MDE), which is an ill-posed problem [215], consists of predicting the scene depth given only an RGB image as the input. Deep Neural Networks (DNNs) learn the mapping between the single RGB images and their corresponding depth maps to solve MDE, and show good performance on indoor and outdoor benchmarks [176, 227].

We refer to these techniques as Classification Approaches for Regression (CAR), and we explore CAR techniques applied to monocular depth estimation.

Classification Approaches for Regression (CAR) [15, 28, 51, 68, 136, 244] have emerged recently in the spotlight among MDE algorithms. The core idea is to transfer regression to a classification problem using quantization (or discretization) strategies. The classification models can natively provide confidence for prediction results, which also has the potential to improve the prediction accuracy [51].

As discussed previously, DNNs are prone to two kinds of uncertainty: aleatoric uncertainty and epistemic uncertainty [113], thus is crucial to study the uncertainty of DNNs if we want to rely on their predictions. Some works proposed to estimate the uncertainty of MDE DNNs by using an auxiliary network [191, 252], or ensembling [123]. Here we gain access to the uncertainty directly using the CAR DNN.

This work will investigate and show the complete picture of the CAR MDE methods. The contributions are as follows:

Contributions

1. We systematically summarize and formalize all fundamental CAR MDE mechanisms. The goal is to find the similarities and differences between these methods and find the best CAR solution under the same settings.
2. We propose a new, effective uncertainty estimation method named Expectation of Distance for CAR MDE models, which is a weighted variance of the network outputs. It outperforms the previous solutions, such as entropy based on CAR.

We implement these mechanisms on top of two different backbones, comparing depth prediction and uncertainty quality on various evaluation metrics.

4.2 Related works

To better unify the terms and make it easier to grasp the differences in contributions of CAR strategies, we propose to decompose the CAR problems into three key components: *discretization*, *loss function* and *post-processing*. Table 4.1 offers an overview of the specific strategies used in the previous works. The details are provided in the following sections.

The contributions of previous CAR-based MDE solutions fall into two main groups:

1. Novel strategies in the three components mentioned above [15, 28, 51, 68, 136, 244];
2. Architecture and/or loss modifications based on the previous strategies [139, 141, 149, 188, 200, 249]

In most papers, CAR can improve model accuracy, making it outperform its regression version [15, 28, 51, 68, 136, 139, 244], or may improve model performance as part of multi-task learning [141, 200]. We will summarize them according to the most distinguished designs with respect to the three key components in the following sections. The visual summary is shown in Fig. 4.1 for a more direct understanding.

CAR-MDEs		**Discretization**				**Post processing**
		Fully Handcrafted			Adaptive	
		Handcrafted + One-hot	Handcrafted + Ordinal	Handcrafted + Smooth		
Loss function	KL divergence/	-	-	SORN [51]	-	Argmax
	Weighted CE loss	-	-	Cao et al. [28] ([249])	-	Soft weighted sum
	CE loss	Li et al. [136] ([139, 141])	-	-	-	
	Multiple BCE loss	-	-	Yang et al. [244]	-	
	Regression loss	DS-SIDENet [200]	-	-	Adabins [15]	
	Ordinal Regression loss	-	DORN [68] ([149, 188])	-	-	Ordinal sum

Table 4.1: Summary on typical CAR-MDE solutions. In parentheses are methods that use the corresponding schemes as part of their solutions.

4.3 Classification Approaches for Regression

4.3.1 General notations

Let us first consider a monocular depth dataset $\mathrm{D} = \{(\mathbf{x}_i, \mathbf{d}_i)\}_i$, where $\mathbf{x}_i \in \mathbb{R}^{3\times \mathrm{H}\times \mathrm{W}}$, and $\mathbf{d}_i \in (\mathbb{R}^+)^{\mathrm{H}\times \mathrm{W}}$ represents the ground truth depth $\mathbf{d}_i$ for the image $\mathbf{x}_i$. We denote $\{d_{i,j}\}_j^{\mathrm{N}}$ all the pixel values in $\mathbf{d}_i$, where N is the number of pixels with valid ground truth. a, b are two real values representing the minimum and maximum depth values for the dataset, respectively.

For CAR strategies, we denote K as the number of classes, which represents the level of discretization. Additionally, to simplify the notations, we use log as the logarithm with base e for all papers except for [139, 200], where the log refers to the logarithm with base 10. We denote $f_{\boldsymbol{\omega}_1}$ the DNN with parameters $\boldsymbol{\omega}_1$. Given $\mathbf{x}_i$, the prediction of $f_{\boldsymbol{\omega}_1}$:

$$\hat{\mathbf{y}}_i = \frac{e^{f_{\boldsymbol{\omega}_1}(\mathbf{x}_i)}}{\sum_{p=0}^{c-1} e^{[f_{\boldsymbol{\omega}_1}(\mathbf{x}_i)]_p}} \tag{4.1}$$

where $f_{\boldsymbol{\omega}_1}(\mathbf{x}_i) \in \mathbb{R}^{c\times \mathrm{H}\times \mathrm{W}}$ is the logit map and $[f_{\boldsymbol{\omega}_1}(\mathbf{x}_i)]_p$ its p-th coefficient, and $\hat{\mathbf{y}}_i$ is the Softmax output. The number of its channels $c = \mathrm{K}$ by default, otherwise equals to the specific settings as in DORN [68] (2K) and Adabins [15] (128). Table. 4.2 lists some of the notations we denote and will use in the following sections.

4.3.2 Discretization: Fully Handcrafted

The discretization function will output two components given $\mathbf{d}$: a *depth table* $\overline{\mathbf{d}} = \{\overline{d}_p\}_p^{\mathrm{K}} \in \mathbb{R}^{\mathrm{K}}$ where the possible discrete depth values are set ordinally, and an indicator map equivalent to a *classification map* $\mathbf{y}_i = \{\{y_{i,j,p}\}_p^{\mathrm{K}}\}_j^{\mathrm{N}} \in (\mathbb{R}^+)^{\mathrm{K}\times \mathrm{H}\times \mathrm{W}}$ which points for each pixel the closest discrete depth value. This closest depth value can be considered as a class, leading to a classification task. Both $\overline{\mathbf{d}}$ and $\mathbf{y}_i$ are *handcrafted*, and the goal becomes the learning of $\mathbf{y}_i$.

Handcrafted $\overline{\mathbf{d}}$ $\overline{\mathbf{d}}$ contains K values representing the centers of intervals $[\overline{\overline{d}}_0, \overline{\overline{d}}_1[, .., [\overline{\overline{d}}_{\mathrm{K}-1}, \overline{\overline{d}}_{\mathrm{K}}[$ with an interval width q:

$$\begin{aligned} \overline{\mathbf{d}} &= \left\{\overline{d}_p\right\}_p^{\mathrm{K}} = \left\{(\overline{\overline{d}}_0 + \overline{\overline{d}}_1)/2, ..., (\overline{\overline{d}}_{\mathrm{K}-1} + \overline{\overline{d}}_{\mathrm{K}})/2\right\} \\ \overline{\overline{d}}_k &= \log a + k\cdot q, k \in [0, \mathrm{K}], q = (\log b - \log a)/\mathrm{K} \end{aligned} \tag{4.2}$$

Types	Notations	Meanings
sub/super-scripts	i	subscript for image/depth index
	j	subscript for pixel index
	p	subscript for channel index
	[method name]	superscript for indicating different methods
common capital letters	N	number of pixels with valid ground truth
	K	number of bins, the level of discretization
some other notations	$\mathbf{d}_i = \{d_{i,j}\}_j^{\mathrm{N}}$	ground truth depth values
	$\overline{\mathbf{d}} = \{\overline{d}_p\}_p^{\mathrm{K}}$	handcrafted logarithm depth table
	$\hat{\overline{\mathbf{d}}}_i = \{\hat{\overline{d}}_p\}_p^{\mathrm{K}}$	learned (adaptive) depth table for a given image
	q	width between two side-by-side bins in the depth table
	$f_{\boldsymbol{\omega}_1}(\mathbf{x}_i)$	output logits of the DNN $f_{\boldsymbol{\omega}_1}$
	$\hat{\mathbf{y}}_i = \{\{\hat{y}_{i,j,p}\}_p^{\mathrm{K}}\}_j^{\mathrm{N}}$	softmax output of the DNN $f_{\boldsymbol{\omega}_1}$

Table 4.2: Reminder for the notations

Handcrafted $\overline{\mathbf{d}}$ + One-hot $\mathbf{y}_i$ Given $\overline{\mathbf{d}}$, constructing $\mathbf{y}_i$ is done using one-hot encoding, as applied in [136, 139, 141, 200]:

$$\mathbf{y}_{i,j}^{[\text{onehot}]} = \begin{bmatrix} y_{i,j,0} & \cdots & y_{i,j,k} & \cdots & y_{i,j,\text{K}-1} \end{bmatrix} \in \mathbb{R}^{\text{K}} \tag{4.3}$$
$$\text{with } y_{i,j,k} = 1, \text{ if } k = \lfloor (\log(d_{i,j}/a))/q \rceil, \text{ and 0 otherwise.}$$

in which, $\lfloor . \rceil$ is a rounding operator, q is defined in Eq. 4.2.

Handcrafted $\overline{\mathbf{d}}_i$ + Ordinal $\mathbf{y}_i$ Furthermore, there are several variants of Eq 4.3. Ordinal properties can be applied on it as presented in [68] and the followed works [149, 188]:

$$\mathbf{y}_{i,j}^{[\text{ordi}]} = \begin{bmatrix} y_{i,j,0} & \cdots & y_{i,j,k} & \cdots & y_{i,j,\text{K}-1} \end{bmatrix} \in \mathbb{R}^{\text{K}} \tag{4.4}$$
$$\text{with } y_{i,j,k} = 1, \text{ if } k \leq \lfloor (\log(d_{i,j}/a))/q \rceil, \text{ and 0 otherwise.}$$

Handcrafted $\overline{\mathbf{d}}_i$ + Smooth $\mathbf{y}_i$ It is also possible to predict a smooth discrete map from the initial discrete map $\mathbf{y}_{i,j}^{[\text{onehot}]}$. The indicator in the classification map is softened by applying on $\overline{\mathbf{d}}$ a Gaussian kernel, as to predict distance within a coarser range. The smooth $\mathbf{y}_i$ are defined by:

$$\mathbf{y}_{i,j}^{[\text{smo1}]} = e^{-\gamma||\log(d_{i,j})-\overline{\mathbf{d}}||^2} \tag{4.5}$$

$$\mathbf{y}_{i,j}^{[\text{smo2}]} = \frac{e^{-\gamma||\log(d_{i,j})-\overline{\mathbf{d}}||^2}}{\sum_{p=0}^{\text{K}-1} e^{-\gamma||\log(d_{i,j})-\overline{d}_p||^2}} \tag{4.6}$$

where γ is a hyperparameter which can be regarded as the scale of the discrete distribution (the smaller γ, the flatter the label distribution in $\mathbf{y}_{i,j}$). Specifically, Yang et al. [244] use Eq 4.5 as the unnormalized soft target labels, while SORN [51] applies the normalized version in Eq 4.6. Moreover, Cao et al. [28] introduce a $\text{K} \times \text{K}$ symmetric "information gain" matrix H in their loss function with elements $\text{H}(k, \mathbf{p}) = e^{-\gamma||k-\mathbf{p}||^2}$, where $\mathbf{p} = [0, ..., \text{K}-1]$ and k is the discrete ground truth index as defined in Eq 4.3. In this case:

$$\mathbf{y}_{i,j}^{[\text{smo3}]} = e^{-\gamma||k-\mathbf{p}||^2} = e^{-\gamma \cdot q^{-2}||\log(d_{i,j})-(\overline{\mathbf{d}}-0.5q)||^2} \tag{4.7}$$

Since q is a constant, this strategy can be regarded as being equivalent to the $\mathbf{y}_{i,j}^{[\text{smo1}]}$ in Eq 4.5.

4.3.3 Discretization: Adaptive

In the absence of the handcrafted depth table or the classification map, one may also implicitly train both of them using a regression loss as in Adabins [15]. Thus the goal of the DNN is changed from fitting the handcrafted classification maps to fitting the continuous ground truth depth while still following the principle of building depth tables and classification maps.

In Adabins [15], the depth table is implicitly trained along with the classification map using a non-linear block $g_{\boldsymbol{\omega}_2}$ with $\boldsymbol{\omega}_2$ the parameters of g, which is a mini ViT [15, 53]. In this case, $g_{\boldsymbol{\omega}_2}$ is set on top of the backbone $f_{\boldsymbol{\omega}_1}$, and it will output $\hat{\overline{\mathbf{d}}}_i^{[\text{ada}]}$ and $\hat{\mathbf{y}}_i^{[\text{ada}]}$ given $f_{\boldsymbol{\omega}_1}(\mathbf{x}_i)$:

$$\hat{\overline{\mathbf{d}}}_i^{[\text{ada}]} = \left\{ a + (b-a)\left(\sum_{s=0}^{p} \hat{\overline{d}}_{i,s}^{[\text{ada}]}\right) \right\}_{p=0}^{\text{K}-1}, \hat{\mathbf{y}}_{i,j}^{[\text{ada}]} = \frac{e^{\mathbf{l}_{i,j}}}{\sum_{p=0}^{\text{K}-1} e^{l_{i,j,p}}} \tag{4.8}$$
$$\text{with } \{\hat{\overline{d}}_i^{[\text{ada}]}\}_p^{\text{K}}, \mathbf{l}_i = g_{\boldsymbol{\omega}_2}(f_{\boldsymbol{\omega}_1}(\mathbf{x}_i))$$

where, $\hat{\mathbf{y}}_{i,j}^{[\text{ada}]}$ is the product of a Softmax function, and $\hat{\overline{\mathbf{d}}}_i^{[\text{ada}]}$ is a cumulative summation output followed by a normalization operation which is included in $g_{\boldsymbol{\omega}_2}$. Since $\hat{\overline{\mathbf{d}}}_i^{[\text{ada}]}$ is a product of $g_{\boldsymbol{\omega}_2}$ taking $f_{\boldsymbol{\omega}_1}(\mathbf{x}_i)$, for each $\mathbf{x}_i$, not only a unique classification map but also a unique depth table will be provided.

4.3.4 Loss function

Based on the previous discretization strategies, we introduce here the loss function design. Models should fit their output to the designed $\mathbf{y}$ or $\mathbf{d}$. For brevity, we define first the total loss $\mathrm{L}_{\text{total}} = \sum_i \sum_{j=0}^{\mathrm{N}-1} \mathrm{L}_{i,j}$ where $\mathrm{L}_{i,j}$ is the loss for pixel j, on the i-th data. We just define $\mathrm{L}_{i,j}$ in the following sections for simplicity.

Cross entropy (CE) loss CE loss is a straightforward solution given a one-hot classification map:

$$\mathrm{L}_{i,j}^{[\mathrm{CE}]}(\boldsymbol{\omega}_1) = -(\mathbf{y}_{i,j}^{[\mathrm{onehot}]} \log \hat{\mathbf{y}}_{i,j}) \tag{4.9}$$

Ordinal regression loss It is essentially an implicit ordinal selection plus a multiple binary cross entropy (BCE) loss. Instead of directly using $\hat{\mathbf{y}}_{i,j}$, it requires to do an ordinal selection on the logit map $f_{\boldsymbol{\omega}_1}(\mathbf{x}_i)$ with $c = 2\mathrm{K}$ to $c = \mathrm{K}$ as the predicted classification map, then to apply a Multiple-BCE loss on it:

$$\begin{aligned} \mathrm{L}_{i,j}^{[\mathrm{ordi}]}(\boldsymbol{\omega}_1) &= -\left[\mathbf{y}_{i,j}^{[\mathrm{ordi}]} \log \hat{\mathbf{y}}_{i,j}^{[\mathrm{ordi}]} + (1 - \mathbf{y}_{i,j}^{[\mathrm{ordi}]}) \log(1 - \hat{\mathbf{y}}_{i,j}^{[\mathrm{ordi}]})\right] \\ \text{with} \quad \hat{\mathbf{y}}_{i,j}^{[\mathrm{ordi}]} &= \frac{e^{[f_{\boldsymbol{\omega}_1}(\mathbf{x}_i)]_{2p+1}}}{e^{[f_{\boldsymbol{\omega}_1}(\mathbf{x}_i)]_{2p+1}} + e^{[f_{\boldsymbol{\omega}_1}(\mathbf{x}_i)]_{2p}}} \end{aligned} \tag{4.10}$$

where $2p+1$ and $2p$ represent the indices of the coefficient.

Weighted CE loss This loss is applied when the target vector is a soft discrete distribution. The CE loss turns out to be equal to the following:

$$\mathrm{L}_{i,j}^{[\mathrm{WCE}]}(\boldsymbol{\omega}_1) = -(\mathbf{y}_{i,j}^{[\mathrm{smo2}]} \log \hat{\mathbf{y}}_{i,j}) \tag{4.11}$$

and it has the same form as the Kullback-Leibler divergence loss.

Multiple BCE loss This loss is another solution when the target is a soft discrete distribution. Yang et al. [244] apply BCE loss in every class. The loss function is similar to Eq. 4.10:

$$\mathrm{L}_{i,j}^{[\mathrm{MBCE}]}(\boldsymbol{\omega}_1) = -\left[\mathbf{y}_{i,j}^{[\mathrm{smo1}]} \log \delta([f_{\boldsymbol{\omega}_1}(\mathbf{x}_i)]_j) + (1 - \mathbf{y}_{i,j}^{[\mathrm{smo1}]}) \log(1 - \delta([f_{\boldsymbol{\omega}_1}(\mathbf{x}_i)]_j)\right] \tag{4.12}$$

where $\delta(\cdot)$ represents the Sigmoid function.

Regression loss (Smooth L1 loss) DS-SIDENet [200] applies CAR with a smooth L1 loss [31] to fit the one-hot classification map target:

$$\begin{aligned} \mathrm{L}_{i,j}^{[\mathrm{smoL1}]}(\boldsymbol{\omega}_1) &= \begin{cases} 0.5(\hat{d}_{i,j} - k)^2 & \text{if } |\hat{d}_{i,j} - k| < 1 \\ |\hat{d}_{i,j} - k| - 0.5 & \text{otherwise} \end{cases} \\ \text{with } \hat{d}_{i,j} &= \sum_{p=0}^{\mathrm{K}-1} \hat{y}_{i,j,p} \cdot (p+1) \end{aligned} \tag{4.13}$$

where k is defined in Eq. 4.3.

Regression loss (Scale-Invariant loss) This loss is applied along with post-processing to produce the continuous depth. Adabins [15] uses the per-image adaptive depth table $\hat{\mathbf{d}}_i^{[\mathrm{ada}]}$ defined in Eq. 4.8 instead of the fixed $\overline{\mathbf{d}}$, then applies a Scale-Invariant loss [57]:

$$\begin{aligned} \mathrm{L}_i^{[\mathrm{SI}]}(\boldsymbol{\omega}_1, \boldsymbol{\omega}_2) &= \omega \sqrt{\frac{1}{\mathrm{N}} \sum_{j=0}^{\mathrm{N}} h_{i,j}^2 - \frac{\lambda}{\mathrm{N}^2} (\sum_{j=0}^{\mathrm{N}} h_{i,j})^2} \\ \text{with } h_{i,j} &= \log \hat{d}_{i,j}^{[\mathrm{ada}]} - \log d_{i,j}; \quad \hat{d}_{i,j}^{[\mathrm{ada}]} = \sum_{p=0}^{\mathrm{K}-1} \hat{y}_{i,j,p}^{[\mathrm{ada}]} \cdot \hat{\overline{d}}_{i,p}^{[\mathrm{ada}]} \end{aligned} \tag{4.14}$$

where ω and λ are hyper-parameters.

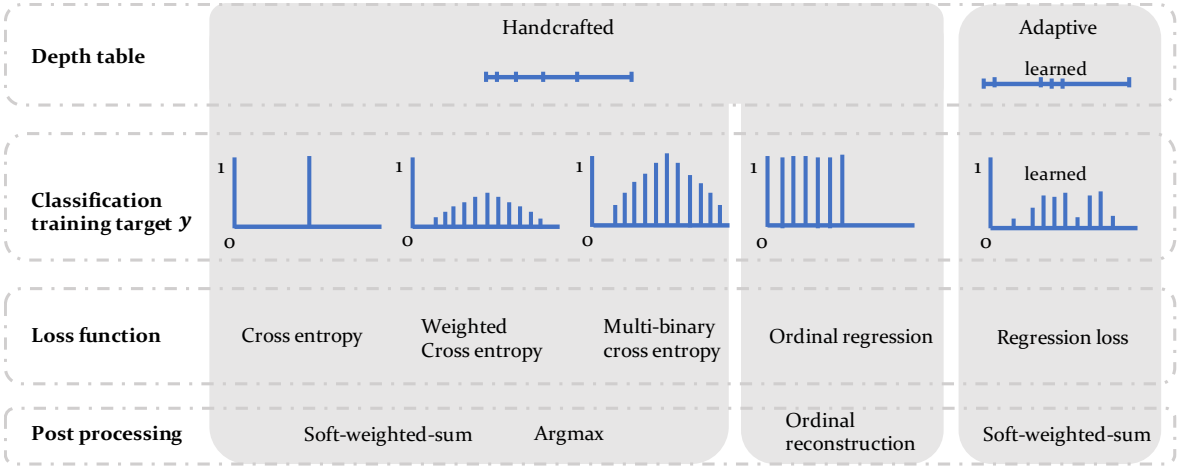


Figure 4.1: Visual summary of the three key components applied in CAR.

4.3.5 Post-processing

Post-processing aims to restore the discrete predicted labels to continuous depth values. In the following equations, we use the power function in base e. See Sec. 4.3.1.

Ordinal sum For DORN [68], the continuous depth is restored from the sum of the output Sigmoid labels which are higher than or equal to 0.5:

$$\hat{d}_{i,j} = \exp\left\{\log(a) + q \cdot \left[\sum_{p=0}^{K-1} \mathbb{1}\{\hat{y}_{i,j,p}^{[\text{ordi}]} \geq 0.5\} + 0.5\right]\right\} \tag{4.15}$$

in which q and $\hat{y}_{i,j,p}^{[\text{ordi}]}$ are defined in Eq. 4.2 and Eq. 4.10 respectively.

Soft weighted sum This solution can be applied to both handcrafted or learned depth tables. It sums the Hadamard product between the depth table and the classification map:

$$\hat{d}_{i,j} = \exp\left\{\sum_{p=0}^{K-1} \overline{d}_p \cdot \hat{y}_{i,j,p}\right\} \tag{4.16}$$

Note that essentially Eq. 4.14 also follows this pattern.

Argmax The authors of SORN [51] claim that Argmax outperforms Soft weighted sum in their case:

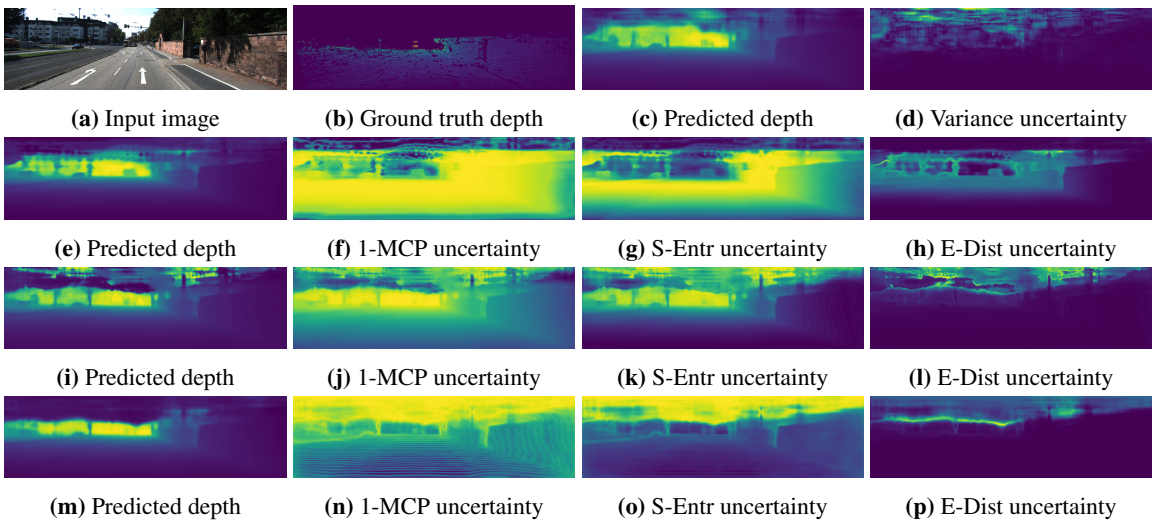
$$\hat{d}_{i,j} = \exp\left\{\log(a) + q \cdot \left[\underset{p}{\text{argmax}}(\{\hat{y}_{i,j,p}\}_p^K) + 0.5\right]\right\} \tag{4.17}$$

where q is defined in Eq. 4.2.

4.3.6 Uncertainty estimation of CAR MDE

In this section, we will discuss the previous works on uncertainty estimation for CAR MDE, the difficulty of this problem and our proposed approaches on estimating CAR uncertainty.

The ground truth uncertainty or the Oracle should be the model's prediction error. The previous works on MDE uncertainty estimation [191, 252] mainly use the principle of learning the prediction error [113]. Meanwhile, the **Variance** among the point estimations given by MC-Dropout [70] and Deep Ensembles [123] can also be applied for this task. Unlike the previous works, the likelihoods of the predicted class (the quantified depth value) given the input data provided by CARs can offer



(a) Input image **(b)** Ground truth depth **(c)** Predicted depth **(d)** Variance uncertainty

(e) Predicted depth **(f)** 1-MCP uncertainty **(g)** S-Entr uncertainty **(h)** E-Dist uncertainty

(i) Predicted depth **(j)** 1-MCP uncertainty **(k)** S-Entr uncertainty **(l)** E-Dist uncertainty

(m) Predicted depth **(n)** 1-MCP uncertainty **(o)** S-Entr uncertainty **(p)** E-Dist uncertainty

Figure 4.2: Illustrations of predicted depth and uncertainty for the selected strategies applied on KITTI dataset. For both depth and uncertainty, the brighter the pixel is, the higher the depth/uncertainty value is. The figures are arranged as follows:
Input image and ground truth depth: (a) (b). For the different strategies: **Deep Ensembles** [123]: (c) (d); **Adabins based** [15]: (e) - (h); **Dorn based** [68]: (i) - (l); **Yang et al. based** [244]: (m) - (p). All these strategies are based on FCN-ResNet101 [87, 151] backbone.

another possibility to estimate the uncertainty mentioned in the previous works but rarely discussed. Yang et al. [244] suggest to use **Shannon Entropy** (**S-Entr**) [211] among the output Softmax classification map:

$$\text{S-Entr} = -\sum_{p=0}^{K-1} \hat{y}_{i,j,p} \log \hat{y}_{i,j,p} \tag{4.18}$$

Moreover, they showed cases where the depth is well predicted, yet the entropy is high, leading to an under-confident uncertainty score. The output of other strategies such as **1-Maximum Class Probability** (**1-MCP**) can also be regarded empirically as an uncertainty estimation [11]:

$$\text{1-MCP} = 1 - \max_{p}(\{\hat{y}_{i,j,p}\}_p) \tag{4.19}$$

These are typical solutions used in classification tasks, and we argue that they will be suitable in the case of using Argmax in post-processing for CAR problems. Widely used soft weighted sum (see Table 4.1) makes the property of CAR unique: not only the classification map but also the depth table should be considered in the final result.

Following these remarks, we propose a new solution for CAR uncertainty here. We first note that the previously mentioned methods lack consideration of the depth table and further its relationship to the classification map. Hence, we define as CAR uncertainty metric: the **Expectation of Distance** (**E-Dist**) between the quantified depth values (either handcrafted logarithm depth table $\overline{\mathbf{d}}$ (Eq. 4.2) or the learned one $\hat{\overline{\mathbf{d}}}_i^{[ada]}$ (Eq. 4.8)) and the final predicted depth $\hat{\mathbf{d}}_i$ (Eq. 4.15, 4.16, 4.17) :

$$\text{E-Dist} = \sum_{p=0}^{K-1} \hat{y}_{i,j,p} \cdot (e^{\overline{d}_p} - \hat{d}_{i,j})^2 \quad \text{or} \quad \text{E-Dist} = \sum_{p=0}^{K-1} \hat{y}_{i,j,p} \cdot (\hat{\overline{d}}_{i,p}^{[ada]} - \hat{d}_{i,j})^2 \tag{4.20}$$

In the recent work [94], the authors share a similar idea of using Eq. 4.20 as the uncertainty estimation for CAR-based models, where the conditional probability mass function is used to model the CAR problem and the Eq. 4.20 is regarded as the error variance that can provide information about the spread of predicted probability mass.

Additionally, to our knowledge, no previous works discuss the uncertainty of the ordinal regression model [68]. According to its CAR strategy, only values greater than or equal to 0.5 in its classification

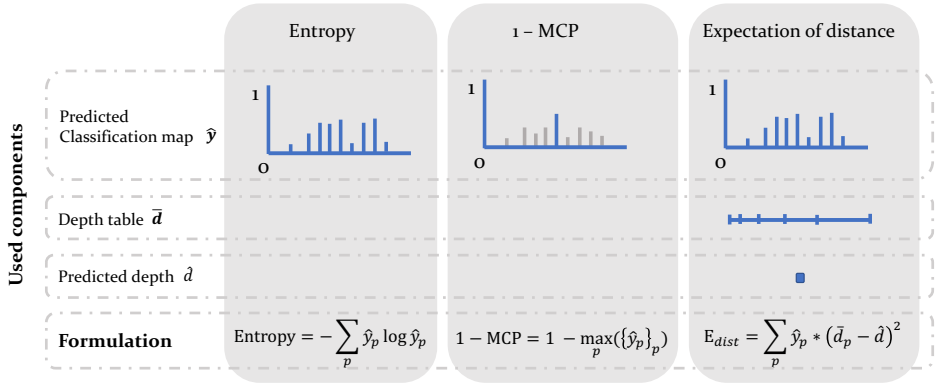


Figure 4.3: Visual summary of the uncertainty quantification methods using CAR.

map will be considered in the final depth calculation. Thus we argue that the uncertainty comes from this part. The modified E-Dist for ordinal regression is as below: we propose to discretize the depth prediction $\hat{\mathbf{d}}_i$ using Eq. 4.4, that we denote as $\mathbf{y}'^{[\text{ordi}]}_{i,j}$. Then we calculate the distance between $\mathbf{y}'^{[\text{ordi}]}_{i,j}$ and $\hat{\mathbf{y}}^{[\text{ordi}]}_{i,j}$ (defined in Eq.4.10) weighted by the depth table and only consider the part with $\hat{\mathbf{y}}^{[\text{ordi}]}_{i,j} \geq 0.5$:

$$\sum_{p=0}^{\text{K}-1} e^{\bar{d}_p} \cdot (y'^{[\text{ordi}]}_{i,j,p} - \hat{y}^{[\text{ordi}]}_{i,j,p})^2 \cdot \mathbb{1}\{\hat{y}^{[\text{ordi}]}_{i,j,p} \geq 0.5\} \tag{4.21}$$

The visual summary is shown as Fig. 4.3. Our E-dist involved all three components and formalized the uncertainty quantification as a depth variance weighted by the predicted classification map. Note that in contemporaneous works, we found Hu et al. [94] to be consistent with our idea. They used the variance from the estimated conditional probability mass function to infer the variance of the response error (see Eq.11-12 in [94]), which has the same form in the end as our proposed E-dist.

4.4 Experiments

The previous works collected in Table 4.1 lack a full comparison, and the network structures and hyperparameters they use are different, as we summarized in Table 4.3. In this section, we fill in the missing comparisons of the previous works. Meanwhile, our experiments provide an extensive analysis of CAR MDE uncertainty estimation. While it is not trivial to propose a model-agnostic approach, the ensuing discussion establishes some important guidelines for performing this task on CAR models.

4.4.1 Experiment settings

All the experiments are based on Eigen-split [57] KITTI dataset [227]. We followed the original settings in the corresponding papers for CAR strategies and applied them on a regression-based and a classification-based backbone, respectively. In addition, we added experiments with K = 80 to the methods with originally different choices for K for better comparison. Fig 4.4 illustrates the experiment pipeline.

4.4.1.1 Backbones

Regression-based backbone We choose BTS-DenseNet161 [100, 130] as the backbone. We add a classifier head as in FCN [151] for a smoother output of a multiple-channel map on the top. We find that this can get better results than just adjusting the number of output channels. We use the same BTS training settings for all the methods.

CAR MDE solutions	Encoder Backbone	Choice of K	Param. smooth	Uncertainty eval.?	Comp. with other CAR?	Comp. with Reg?
DORN [68]	ResNet-101 [87]	80	-	✗	✓	✓
Cao et al. [28]	ResNet-101 [87]	50	65	✗	✗	✓
Li et al. [136]	ResNet-152 [87]	50 (150 [139])	-	✗	✗	✓
SORN [51]	Xception [37]	120	1	✗	✗	✓
Yang et al. [244]	ResNet-50 [87]	128	15	✓	✓	✓
DS-SIDE [200]	Self-made	80	-	✗	✓	✓
Adabins [15]	EfficientNet [222]	256	-	✗	✓	✓

Table 4.3: Summary on KITTI experiment settings for typical CAR strategies. **Param. smooth**: Coefficient used for label smoothing. **Uncertainty eval.**: Whether this work evaluates the uncertainty and compared with the other works. **Comp. with other CAR**: Whether this work compared their proposed CAR strategy with the other CAR methods. **Comp. with Reg.**: Whether this work compared the CAR with the regression version of its model.

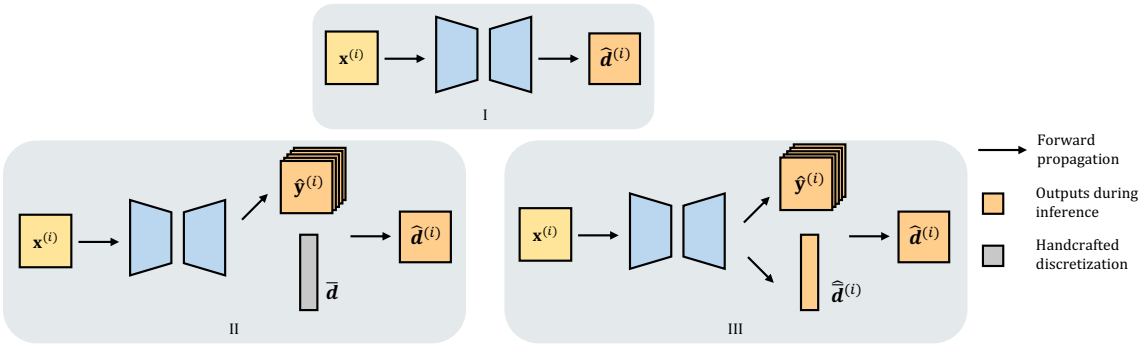


Figure 4.4: Experiment pipeline and three types of architectures for monocular depth estimation task. *I*. original regression version; *II*. modeling using handcrafted discretization; *III*. modeling using adaptive discretization using a mini ViT module [15, 53] on the top.

Classification-based backbone We choose FCN-ResNet101 [87, 151] as the backbone. FCN is originally designed for semantic segmentation, thus, it is suitable for CAR methods. For the one-channel regression version (org), followed BTS, we apply a Sigmoid on the top and multiply the output by b.

4.4.1.2 Evaluation matrices

Uncertainty quality evaluation We use the area under sparsification error curve (AUSE), as in [191, 244, 252]. 1% of pixels are removed each time and we calculate RMSE and AbsRel for the rest. The pixels are removed according to their RMSE and AbsRel from high to low as the Oracle curves and removed based on their predicted uncertainty order as the predictive curves. The areas between the predictive curves and the Oracle curves are denoted as AUSE-RMSE and AUSE-AbsRel. The uncertainty estimation methods we used are introduced in Sec. 4.3.6. We will compare the CAR MDE uncertainty with widely used MC-Dropout [70] (with 8 forward passes) and Deep Ensembles [123] (with 3 models).

Training time consumption We use one NVIDIA Titan RTX to count the time consumption to train 500 images for all CAR methods with K = 80 as well as the original regression method and the Deep Ensembles [123] using the same training settings.

Backbones	BTS								FCN								K
Metrics	δ1↑	δ2↑	δ3↑	Abs Rel↓	Sq Rel↓	RMSE↓	RMSE log↓	log10↓	δ1↑	δ2↑	δ3↑	Abs Rel↓	Sq Rel↓	RMSE↓	RMSE log↓	log10↓	
DORN [68]	**0.952**	**0.992**	**0.998**	0.069	**0.267**	**2.802**	**0.103**	**0.029**	**0.940**	0.990	**0.998**	0.076	**0.292**	**2.962**	**0.113**	0.033	80
Cao et al. [28]	0.945	**0.992**	**0.998**	0.077	0.292	2.988	0.111	0.034	0.928	0.989	**0.998**	0.084	0.344	3.223	0.122	0.036	50
Li et al. [136]	0.950	0.990	**0.998**	0.070	0.287	2.928	0.106	0.030	**0.940**	0.988	0.997	0.075	0.314	3.190	0.116	0.033	150 [139]
SORN [51]	0.947	**0.992**	**0.998**	0.071	0.290	2.929	0.107	0.031	0.863	0.976	0.995	0.119	0.563	3.938	0.163	0.051	120
Yang et al. [244]	0.951	0.991	**0.998**	**0.065**	0.276	2.897	**0.103**	**0.029**	**0.940**	0.989	0.997	**0.072**	0.302	3.096	**0.113**	**0.032**	128
DS-SIDE [200]	0.950	0.991	**0.998**	0.071	0.275	2.886	0.106	0.032	0.931	0.990	**0.998**	0.079	0.331	3.353	0.119	0.035	80
Adabins [15]	0.935	0.990	**0.998**	0.078	0.347	3.143	0.114	0.033	0.937	**0.991**	**0.998**	0.079	0.331	3.027	**0.113**	0.033	256
DORN [68]	0.952	0.992	**0.998**	0.069	**0.267**	**2.802**	**0.103**	**0.029**	**0.940**	**0.990**	**0.998**	**0.076**	**0.292**	**2.962**	0.113	0.033	80
Cao et al. [28]	**0.953**	0.991	**0.998**	**0.066**	0.268	2.857	**0.103**	**0.029**	0.934	0.989	0.997	**0.076**	0.319	3.124	0.117	0.033	80
Li et al. [136]	0.949	0.990	0.997	0.087	0.305	2.982	0.116	0.037	0.933	0.988	0.997	0.096	0.350	3.157	0.126	0.040	80
SORN [51]	0.949	**0.993**	**0.998**	0.072	0.283	2.902	0.106	0.031	0.863	0.976	0.995	0.122	0.573	3.950	0.165	0.052	80
Yang et al. [244]	0.948	0.991	**0.998**	0.070	0.284	2.973	0.107	0.031	**0.940**	**0.990**	0.997	**0.076**	0.308	3.067	0.115	0.034	80
DS-SIDE [200]	0.950	0.991	**0.998**	0.071	0.275	2.886	0.106	0.032	0.931	**0.990**	**0.998**	0.079	0.331	3.353	0.119	0.035	80
Adabins [15]	0.933	0.989	**0.998**	0.079	0.357	3.203	0.116	0.033	0.937	**0.990**	**0.998**	**0.076**	0.318	3.062	**0.112**	**0.032**	80
Org	0.955	0.993	0.998	0.060	0.249	2.798	0.096	0.027	0.944	0.992	0.998	0.069	0.275	2.938	0.107	0.030	1
MC-Dropout [70]	0.941	0.992	0.998	0.083	0.308	2.910	0.114	0.035	0.918	0.984	0.996	0.085	0.369	3.157	0.125	0.036	1
Deep Ensembles [123]	0.957	0.993	0.999	0.059	0.233	2.688	0.093	0.026	0.946	0.992	0.998	0.068	0.269	2.923	0.106	0.030	1

Table 4.4: Depth accuracy evaluations. Org: the original BTS [130] model and the regression version applied on FCN [151] model.

4.4.2 Performance and discussions

Table 4.4 and Table 4.5 provide all depth and uncertainty results. The best/second-best values are highlighted in dark/light blue. We only highlight the CAR-based results. The results from regression-based models are settled as the reference. Figure 4.2 shows some illustrations for predicted depth as well as the predicted uncertainty given by different uncertainty estimation strategies.

4.4.2.1 Depth

We find that all CAR MDE methods are portable, but training directly with the settings of the original backbones degrades performance. We discover that the Adabins, DS-SIDE, and SORN [15, 51, 200] based models are more sensitive to the selected backbone than the other ones. Despite the influence of the backbones, we also consider that the training settings for the original Adabins are more different from the ones of BTS. This difference may cause Adabins to produce worse performance after porting. We report that the biggest difference is the batch size, where the original Adabins uses 4 times the batch size (16 images) than the original BTS (4 images). DORN-based model [68] achieves the best result among CAR DNNs, which confirms the effectiveness of ordinal constraints.

4.4.2.2 Uncertainty

Our proposed E-Dist shows good and robust performance in most cases given a CAR MDE method. As we can see in Figure 4.2, E-Dist empirically concentrates more on the aleatoric uncertainty (which frequently appears on the edges of the prediction) than S-Entr and 1-MCP. We also argue that S-Entr can provide a higher uncertainty to the sky (upper part in the prediction maps), which is a good property, but it is hard to analyze the uncertainty quality for this part.

Among the CAR strategies, we found that the uncertainty quality is related to the sharpness of the labeling during discretization and also to the loss function. Li et al. [136], Yang et al. [244] and Cao et al. [28] based DNNs perform better for the uncertainty. Li et al. [136] model has one-hot encoded labels in the classification map which leads to the sharpest label distribution. Yang et al. [244] model has $\gamma = 15$ in Eq. 4.5 and we can also have $\gamma \cdot q^{-2} = 65$ in Eq. 4.7 for Cao et al. [28] model. This

Backbones	BTS						FCN						K
Metrics	AUSE RMSE↓			AUSE AbsRel↓			AUSE RMSE↓			AUSE AbsRel↓			
Methods	1-MCP	S-Entr	E-Dist	1-MCP	S-Entr	E-Dist	1-MCP	S-Entr	E-Dist	1-MCP	S-Entr	E-Dist	
Cao et al. [28]	0.542	0.770	0.133	0.382	0.424	0.411	0.532	0.701	0.127	0.354	0.375	0.375	50
Li et al. [136]	0.174	0.153	0.187	0.276	0.262	0.409	0.137	0.138	0.132	0.241	0.235	0.259	150
SORN [51]	1.371	1.394	0.157	0.939	0.982	0.427	1.244	1.283	0.170	0.754	0.755	0.451	120
Yang et al. [244]	0.141	0.145	**0.094**	0.232	**0.219**	0.256	0.142	0.161	**0.111**	**0.225**	0.226	0.247	128
DS-SIDE [200]	0.698	0.806	0.293	0.525	0.544	0.397	1.212	1.331	0.995	0.630	0.722	0.484	80
Adabins [15]	0.855	0.827	0.179	0.536	0.527	0.377	0.984	0.945	0.191	0.608	0.589	0.398	256
DORN [68]	0.188	0.158	0.128	0.530	0.430	0.303	0.202	0.165	0.135	0.593	0.445	0.283	80
Cao et al. [28]	0.371	0.476	0.119	0.323	0.329	0.356	0.349	0.393	0.117	0.284	0.265	0.308	80
Li et al. [136]	0.206	0.178	0.181	0.355	0.348	0.449	0.170	0.163	0.124	0.318	0.310	0.333	80
SORN [51]	1.367	1.390	0.157	0.900	0.941	0.444	1.228	1.275	0.175	0.725	0.737	0.473	80
Yang et al. [244]	0.194	0.179	**0.099**	0.273	**0.259**	0.271	0.156	0.169	**0.104**	0.258	**0.252**	0.274	80
DS-SIDE [200]	0.698	0.806	0.293	0.525	0.544	0.397	1.212	1.331	0.995	0.630	0.722	0.484	80
Adabins [15]	0.823	0.683	0.181	0.499	0.450	0.360	0.775	0.710	0.234	0.502	0.478	0.391	80
DORN [68]	0.188	0.158	0.128	0.530	0.430	0.303	0.202	0.165	0.135	0.593	0.445	0.283	80
MC-Dropout [70]	0.460 (**Variance**)			0.501 (**Variance**)			0.322 (**Variance**)			0.456 (**Variance**)			1
Deep Ensembles [123]	0.165 (**Variance**)			0.261 (**Variance**)			0.184 (**Variance**)			0.290 (**Variance**)			1

Table 4.5: Depth uncertainty evaluations. MC-Dropout [70] and Deep Ensembles [123] will only provide the uncertainty with one method.

big coefficient can sharpen the label distribution. Conversely, in SORN [51], the γ in Eq. 4.6 is much smaller, which results in the evener distributed labels, and we consider this is the main cause of its worse performance. Yang et al. [244] based model outperforms the others, which indicates that the Multi-BCE loss is more suitable for uncertainty estimation, which is similar to the one-versus-all strategy [65].

4.4.2.3 Choices of K

According to two sets of results separated by K, we observed that the performance of uncertainty quantification is positively related to the number of K, while we argue that the depth estimation performance depends on the methods. Most depth estimation performance increases with K, but the opposite for SORN [51]. We think the reason is that the label smoothing strategy of SORN makes the label too smooth. When K becomes large, the label weight around the correct label will be much larger than other methods. In addition, Argmax strategy is used during post-processing, which makes the final depth value easier to estimate around the correct label. On the contrary, the performance of uncertainty estimation is almost always positively related to K. In comparison, the uncertainty estimate is more robust to K, and in the context of monocular depth estimation, we argue that post-hoc methods are less affected by model training and three key components.

In the ablation studies of the previous works, the authors also show that the performance of depth estimation is not always positively correlated with the value of K. However, due to the lack of uncertainty experiments in the previous works, we cannot know the impact of K on uncertainty estimation from experiments in previous work. Our work fills this gap to a certain extent.

4.4.2.4 Time efficiency

According to Table. 4.6, we argue that the CAR strategy will slightly slow down the training, especially for the ones requiring label smoothing in discretization [28, 68, 244]. However, Deep Ensembles [123] with only three models still require the most training time.

Time consumption (ms)									
CAR Solutions	DORN [68]	Cao et al. [28]	Li et al. [136]	SORN [51]	Yang et al. [244]	DS-SIDE [200]	Adabins [15]	Org	Deep Ensembles [123]
BTS	610.96	509.18	431.32	444.66	613.98	430.14	421.06	378.98	1136.94
FCN	735.38	614.52	538.92	556.84	722.80	540.70	588.26	517.66	1552.98

Table 4.6: Time consumption on Forward+Backward passes for one image using 1 NVIDIA Titan RTX.

4.5 Conclusion

In this work, we choose monocular depth estimation as the carrier to summarize the classification approaches for regression in detail, along with three key components, and conduct experiments for evaluating both depth and uncertainty quality. At the same time, we propose the expectation of distance (E-dist), an uncertainty quantification method that is not based on direct learning of prediction errors. On almost all evaluated CAR methods, E-dist can obtain better uncertainty quantification quality than entropy.

This work is a guide that gives us a clearer and more detailed understanding of solving regression problems from the perspective of classification using discretization techniques, especially for uncertainty quantification. In the next chapter, we propose a general auxiliary network solution. Moreover, we pioneered exploring the usage of discretization on epistemic uncertainty estimation in regression problems. By combining discretization with Dirichlet posterior estimation, we show empirically better epistemic uncertainty quantification results in regression.

Chapter 5

Discretization-Induced Dirichlet Posterior for Robust Uncertainty Quantification on Regression

Contents

5.1 Introduction

As mentioned in the previous chapters, from auxiliary uncertainty estimator SLURP [252] to classification approach for regression CAR [253], we discussed the design of auxiliary networks for uncertainty quantification and the usage of discretization to transfer the regression to classification. These guide works lead to this chapter, in which we will introduce a more robust Auxiliary Uncertainty Estimator (AuxUE) design for uncertainty quantification.

A robust AuxUE is required in this case to provide stable aleatoric uncertainty estimates when facing In-Distribution (ID) inputs and robust epistemic uncertainty estimates when encountering OOD inputs. This can help to make effective decisions under anomalies and uncertainty [84], such as in autonomous driving [7]. Based on these requirements, the prerequisite for a robust AuxUE thus to be

disentangling the two types of uncertainty. This can help to estimate the epistemic uncertainty and to find a more robust aleatoric uncertainty estimation solution.

For vision regression tasks, the basic AuxUE addresses only aleatoric uncertainty estimation [252]. In recent works, Upadhyay et al. [229] and Qu et al. [197] aim to improve the generalization ability of the basic AuxUEs. In DEUP [107], the authors propose to add a density estimator based on normalizing flows [201] in the AuxUE, yet challenging to apply on pixel-wise vision tasks. In the current context, both the robustness analysis and modeling of epistemic uncertainty are underexplored for vision regression problems.

To further explore robust aleatoric and epistemic uncertainty estimation in vision regression tasks, in this work, we propose a novel uncertainty quantification solution based on AuxUEs. For estimating aleatoric uncertainty, we follow the approach of previous works such as [113, 180, 229, 252] and model the heteroscedastic noise using different distribution assumptions. For epistemic uncertainty quantification, we use a discretization approach to the continuous prediction errors of the main task. This helps to mitigate the numerical impact of the training targets, which may be distributed in a long-tailed manner. With the discretized prediction errors, we propose parameterizing Dirichlet posterior [34, 108, 209] for estimating epistemic uncertainty without relying on OOD data during the training process.

Contributions

In this work, we aim to propose a solution based on AuxUEs for more robust uncertainty quantification. To achieve this:

1. We propose a generalized AuxUE solution for aleatoric and epistemic uncertainty estimation.
2. We propose Discretization-Induced Dirichlet pOsterior (DIDO), a new epistemic uncertainty estimation strategy for regression, which, to the best of our knowledge, is the only existing work employing this distribution for regression.
3. We demonstrate that assuming the noise that affects the main task predictions to follow Laplace distribution can help AuxUE achieve a more robust aleatoric uncertainty estimation.
4. We propose a new evaluation strategy for the OOD analysis of pixel-wise regression tasks based on systematically non-annotated patterns.

We show the robustness and scalability of the proposed generalized AuxUE and DIDO on the age estimation, super-resolution, and monocular depth estimation tasks.

5.2 Related works

Auxiliary uncertainty estimation solutions

Similar to the Side learning for uncertainty estimation section in Chapter 3, we will briefly go through the Auxiliary uncertainty estimation strategies here and introduce them from two categories: unsupervised and supervised. For the former, to achieve uncertainty estimates, MC-Dropout [70] and Dropout layer [217] injection [167] sample the network by forward propagations, and [92] proposed to use the gradients from the back-propagation. Supervised approaches apply AuxUEs to obtain the uncertainty. For classification tasks, ConfidNet [41] and KLoS [40] learn the true class probability and evidence for the DNNs, respectively. Shen et al. [212] apply evidential classification [108, 209] to their AuxUE. ObsNet [13] uses adversarial noise to provide more abundant training targets in the semantic segmentation task for their AuxUE. For regression tasks, DEUP [107] uses the density estimator [201] and the model variance estimator to construct the AuxUE. SLURP [252] learns the prediction error of the main task DNN, and BayesCap [229] constructs the output as generalized Gaussian distribution parameters. Only the latter two are applied to pixel-wise tasks.

Evidential deep learning and Dirichlet networks

Evidential deep learning [228] (EDL) is a modern application of the Dempster-Shafer Theory [47] to estimate epistemic uncertainty with single forward propagation. In classification tasks, EDL is usually formed as parameterizing a prior [157, 158] or a posterior [33, 34, 108, 209] Dirichlet distribution. In regression problems, EDL estimates the parameters of the conjugate prior of Gaussian distribution [5, 33, 156]. Oh and Shin [182] use multi-task learning to alleviate main task performance degradation due to applying such techniques, yet using AuxUE will not affect main task performance. Therefore, we apply EDL to our AuxUE. Moreover, we are the first to apply the Dirichlet network to the regression tasks by discretizing the main task prediction errors.

Robustness of uncertainty estimation

A robust uncertainty estimator should show stable performance when encountering images perturbed to varying degrees [90, 112, 169]. Similar studies are applied to evaluate the robustness of uncertainty estimates [66, 248]. Meanwhile, it should provide a higher uncertainty when facing OOD data, such as in classification tasks [91, 140, 212]. In image-level regression, we can use the definition of OOD from image classification [223] in, for example, age estimation task. But for pixel-wise regression tasks, the notion of OOD data is ill-defined. Typical OOD analysis estimates uncertainty on a different dataset than the training dataset [33, 229]. Yet, image patterns that have never been assigned the ground truth values in the training set can also be regarded as OOD. For instance, the sky patterns in the real-world outdoor KITTI [75, 227] dataset have no depth ground truth, and they will be OOD patterns in the monocular depth estimation task. In this work, we also provide a new evaluation strategy for OOD patterns based on outdoor depth estimation to compensate for this experimental shortfall.

5.3 Generalized auxiliary uncertainty estimator and DIDO approach

5.3.1 Preliminaries

In this section, we will first provide the notations and the problem settings. We define a training dataset $\mathscr{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_i^{\mathrm{N}}$ where N is the number of images. We consider that $\mathbf{x}, \mathbf{y}$ are drawn from a joint distribution $\mathrm{P}(\mathbf{x}, \mathbf{y})$. A pipeline for the main task and auxiliary uncertainty estimation is shown in Figure 5.1. We define a main task DNN $f_{\boldsymbol{\omega}}$ with trainable parameters $\boldsymbol{\omega}$ as shown in the blue area in Figure 5.1. Similar to [19], we view $f_{\boldsymbol{\omega}}$ as a probabilistic model $\mathrm{P}(y|\mathbf{x}, \boldsymbol{\omega})$ which follows a Gaussian distribution $\mathcal{N}(y|\mu, \sigma^2)$ [17]. The variable σ^2 represents the variance of the noise in the DNN's prediction, and the variable μ is the prediction $\hat{y} = f_{\boldsymbol{\omega}}(\mathbf{x})$ in this case. The noise is considered here to be homoscedastic as all data have the same noise. The parameter $\boldsymbol{\omega}$ is optimized by maximizing the log-likelihood: $\hat{\boldsymbol{\omega}} = \mathrm{argmax}_{\boldsymbol{\omega}} \log(\mathrm{P}(\mathscr{D}|\boldsymbol{\omega}))$ which is often performed by minimizing Negative Log Likelihood (NLL) loss in practice. With the above-mentioned Gaussian assumption on $\hat{\mathbf{y}}$, the NLL loss optimizes with the same objective as the Mean Square Error (MSE) loss [17], thus only the main prediction goal $\mathbf{y}$ is considered, and the uncertainty modeling is absent in the main task model training objective.

AuxUE aims to obtain this missing uncertainty estimation without modifying $\hat{\boldsymbol{\omega}}$. We consider two DNNs σ_{Θ_1} and σ_{Θ_2} in our generalized AuxUE with parameters Θ_1 and Θ_2, i.e., the two DNNs in the orange area of the Figure 5.1. σ_{Θ_1} is for estimating aleatoric uncertainty $\mathbf{u}_{\text{alea}}$, and σ_{Θ_2} is for estimating epistemic uncertainty $\mathbf{u}_{\text{epis}}$. The backbone of σ_{Θ_1} and σ_{Θ_2} are based on the basic AuxUEs such as ConfidNet [41], BayesCap [229] and SLURP [252] depending on the tasks. The input of AuxUE can be the input, output, or intermediate features of $f_{\hat{\boldsymbol{\omega}}}$ and it depends on the design of the basic AuxUEs, which is not the focus of this paper. It depends on the accessibility of the information we can get from the main task network. In practice, if we have only a black box model, we will only use the input and the output of this model. If we have the gray box model, we can get access to the

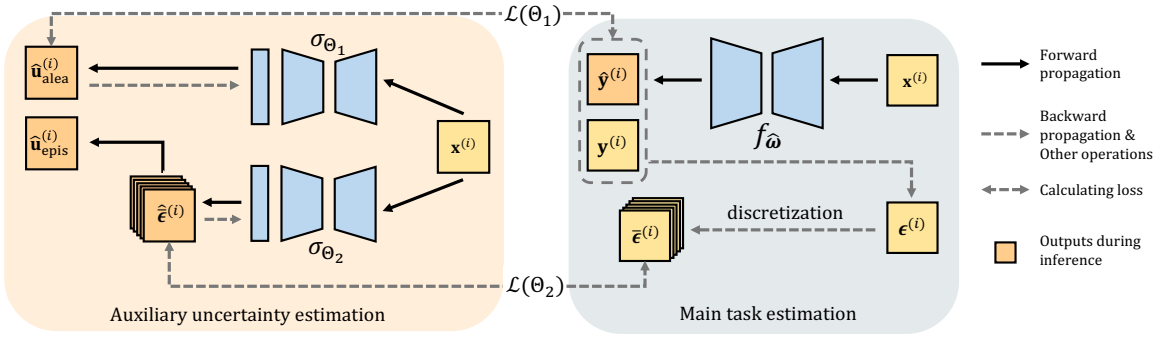


Figure 5.1: Pipeline of our proposed AuxUE solution. A generalized AuxUE is considered with two DNNs σ_{Θ_1} and σ_{Θ_2} for estimating aleatoric and epistemic uncertainty, respectively. The input of AuxUE can be the input, output, or intermediate features of $f_{\hat{\boldsymbol{\omega}}}$, we here simplify it to the image $\mathbf{x}^{(i)}$ for brevity.

intermediate features of the main task model. For brevity, we simplify the input of AuxUE to the image $\mathbf{x}$. We detail the inputs for different experiments in Section 5.4.

5.3.2 Aleatoric uncertainty estimation on AuxUE

Based on the preliminaries of the settings, we now start with the first AuxUE σ_{Θ_1}, which addresses $\mathbf{u}_{\text{alea}}$ estimation problem as in SLURP [252] and BayesCap [229].

We consider the data-dependent noise [16, 80, 180] follows $\mathcal{N}(0, \boldsymbol{\sigma}^2)$. Then we use the DNN σ_{Θ_1} to estimate the heteroscedastic aleatoric uncertainty $\mathbf{u}_{\text{alea}}$ [113, 180]. $\widehat{\Theta}_1$ and the loss function $\mathrm{L}(\Theta_1)$ are given by:

$$\widehat{\Theta}_1 = \underset{\Theta_1}{\operatorname{argmax}} \mathrm{P}(\mathscr{D}|\hat{\boldsymbol{\omega}}, \Theta_1) = \underset{\Theta_1}{\operatorname{argmax}} \sum_{i=1}^{\mathrm{N}} \log(\mathrm{P}(y^{(i)}|\mathbf{x}^{(i)}, \hat{\boldsymbol{\omega}}, \Theta_1)) \tag{5.1}$$

$$\mathscr{L}(\Theta_1) = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \left[\frac{1}{2} \log(2\pi\sigma_{\Theta_1}(\mathbf{x}^{(i)})) + \frac{(y^{(i)} - f_{\hat{\boldsymbol{\omega}}}(\mathbf{x}^{(i)}))^2}{2\sigma_{\Theta_1}(\mathbf{x}^{(i)})} \right] \tag{5.2}$$

The top of the σ_{Θ_1} is an exponential or Softplus function to maintain the output non-negative. The aleatoric uncertainty estimation will be: $\hat{u}^{(i)}_{\text{alea}} = \sigma_{\Theta_1}(\mathbf{x}^{(i)})$. Minimizing $\mathscr{L}(\Theta_1)$ is also equivalent to making σ_{Θ_1} correctly predict the main task errors on the training set. The errors set is denoted as $\boldsymbol{\epsilon} = \{\epsilon^{(i)}\}_{i=1}^{\mathrm{N}} = \{(y^{(i)} - f_{\hat{\boldsymbol{\omega}}}(\mathbf{x}^{(i)}))^2\}_{i=1}^{\mathrm{N}}$.

Given the fact that distribution assumption on the noise affecting $\hat{\mathbf{y}}$ can be different than Gaussian, e.g., Laplacian [161] and Generalized Gaussian distribution [175, 229] also been considered in this work, the corresponding loss functions are listed as follows. For Laplacian distribution, the loss function we use is Eq. 5.3.

$$\mathscr{L}(\Theta_1) = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \log(2\sigma_{\Theta_1}(\mathbf{x}^{(i)})) + \frac{|y^{(i)} - f_{\hat{\boldsymbol{\omega}}}(\mathbf{x}^{(i)})|}{\sigma_{\Theta_1}(\mathbf{x}^{(i)})} \tag{5.3}$$

For Generalized Gaussian distribution, the loss function is as Eq. 5.4.

$$\mathscr{L}(\Theta_1) = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \left(\frac{|y^{(i)} - f_{\hat{\boldsymbol{\omega}}}(\mathbf{x}^{(i)})|}{\hat{\alpha}^{(i)}} \right)^{\hat{\beta}^{(i)}} - \log \frac{\hat{\beta}^{(i)}}{\hat{\alpha}^{(i)}} + \log\Gamma(\frac{1}{\hat{\beta}^{(i)}}) \tag{5.4}$$

with $\sigma_{\Theta_1}(\mathbf{x}^{(i)}) = (\hat{\alpha}^{(i)}, \hat{\beta}^{(i)})$, which means σ_{Θ_1} will output two other components (except for $\tilde{y}^{(i)}$ defined in [229] which stands for $f_{\hat{\boldsymbol{\omega}}}(\mathbf{x}^{(i)})$ in our case) for Generalized Gaussian distribution.

The objective remains unchanged: employing AuxUE to estimate and predict the component associated with aleatoric uncertainty using various distribution assumptions. When input data is perturbed in various ways and under different types of noise, the actual distribution of noise becomes difficult

to be accurately identified. If we rely on a single distribution assumption, the choice of distribution assumption and the loss function can impact the reliability of estimates for aleatoric uncertainty. In our experiments, we compare the effects of using different distribution assumptions and loss functions on the robustness of these estimates.

5.3.3 Epistemic uncertainty estimation on AuxUE

Modeling AuxUEs as formalized in Eq. 5.1 helps to estimate aleatoric uncertainty for $f_{\hat{\omega}}$. However, taking this uncertainty prediction as an indicator for epistemic uncertainty is not methodologically grounded, although some recent works have explored this direction. Upadhyay et al. [229] uses Generalized Gaussian distribution to improve their AuxUE performance on sparse data samples with high prediction errors, and Qu et al. [197] modify the training procedure using meta-learning. Both aim to facilitate for AuxUEs the optimization of learning heteroscedastic noise since the prediction errors are very small on most samples when the main task model is trained, which results in a long-tailed distribution. These methods have the potential to help AuxUE generalize better on rarely seen data and even attach higher uncertainty on unseen ones. Still, we cannot separate epistemic uncertainty from aleatoric uncertainty, and modeling for epistemic uncertainty is needed.

Evidential learning is considered an efficient uncertainty estimation approach [5, 209, 228], which can capture epistemic uncertainty with a single pass. It explicitly parameterizes the prior of the assumed distribution for the main task training target. We thus take it as an alternative to implement on AuxUE. The loss function, in this case, will be Eq. 5.6:

$$\begin{aligned}\mathscr{L}_1(\Theta_1) &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\log(\frac{\pi}{\nu^{(i)}}) - \alpha^{(i)}\log(\Omega^{(i)}) \\ &\quad + (\alpha^{(i)}+\frac{1}{2})\log((y^{(i)}-f_{\hat{\omega}}(\mathbf{x}^{(i)}))^2\nu^{(i)}+\Omega^{(i)}) + \log(\frac{\Gamma(\alpha^{(i)})}{\Gamma(\alpha^{(i)}+\frac{1}{2})}) \\ \mathscr{L}_2(\Theta_1) &= \frac{1}{N}\sum_{i=1}^{N}|y^{(i)}-f_{\hat{\omega}}(\mathbf{x}^{(i)})|\cdot(2\nu^{(i)}+\alpha^{(i)})\end{aligned} \tag{5.5}$$

$$\mathscr{L}(\Theta_1) = \mathscr{L}_1(\Theta_1) + \lambda_{\text{NIG}}\cdot\mathscr{L}_2(\Theta_1) \tag{5.6}$$

where $\Omega^{(i)} = 2\beta^{(i)}(1+\nu^{(i)})$. $\sigma_{\Theta_1}(\mathbf{x}^{(i)}) = (\hat{\alpha}^{(i)}, \hat{\beta}^{(i)}, \hat{\nu}^{(i)})$, which means σ_{Θ_1} will output three other components (except for $\gamma^{(i)}$ defined in [5] which stands for $f_{\hat{\omega}}(\mathbf{x}^{(i)})$ in our case) for Generalized Gaussian distribution.

In regression tasks, the parameters of the conjugate prior (Normal Inverse Gamma (NIG) distribution [5]) of Gaussian distribution are estimated by the DNN. The training will make the model fall back onto a NIG prior for the rare samples by attaching lower evidence for the samples with higher prediction errors using a regularization term $\mathscr{L}_2$ in Eq. 5.5. Yet, long-tailed prediction errors make AuxUE more inclined to give high evidence for most data points, which results in reducing the ability to estimate epistemic uncertainty. Our experiments in Sec. 5.4.5.2 also confirmed this tendency.

Previous works consider the *numerical value* of the prediction errors for both types of uncertainty. In our solution, we disentangle the estimation of aleatoric and epistemic uncertainty and use discretization to decrease the numerical bias caused by long-tailed distributed prediction errors. Specifically, as discussed in Sec. 5.3.2, σ_{Θ_1} aims to provide aleatoric uncertainty estimation considering the *numerical value* of the prediction errors. For epistemic uncertainty, σ_{Θ_2} will consider the *value-free categories* of the prediction errors. Specifically, we propose Discretization-Induced Dirichlet pOsterior (DIDO). The discretization will be conducted on the prediction errors. We then apply evidential learning and estimate the Dirichlet posterior given the discrete errors. We will describe the details in the following Sec. 5.3.3.1 and 5.3.3.3.

5.3.3.1 Discretization on prediction errors

To prevent numerical bias caused by imbalanced data, specifically in our case where we aim to estimate the prediction errors, we utilize a balanced discretization approach. Discretization is widely applied in classification approaches for regression, such as monocular depth estimation and age estimation [15, 27, 28, 68, 253]. The popular discretization methods can be generally divided into handcrafted [28, 68] and adaptive [15]. The latter requires a module (e.g., a mini-ViT [15, 53]) to extract global features, greatly increasing the computational cost. Thus, we discretize the prediction errors in a handcrafted way uniformly, which we detail in the next paragraph.

For pixel-wise cases, discretization is applied to the per-image prediction errors using quantiles. For other cases, e.g., image-level tasks and 1D signal estimation, discretization is applied to the per-dataset prediction errors using quantiles, and the solution is presented as follows.

We divide the set of errors $\boldsymbol{\epsilon}$, denoted in Sec. 5.3.2, into K subsets, where the kth subset is represented by the subscript k. To do this, we sort the errors in ascending order and create a new set, denoted by $\boldsymbol{\epsilon}'$, with the same elements as $\boldsymbol{\epsilon}$. Then we divide $\boldsymbol{\epsilon}'$ into K subsets of equal size, represented by $\{\boldsymbol{\epsilon}_k\}_{k=1}^{\mathrm{K}}$. Each error value $\epsilon^{(i)}$ is then replaced by the index of its corresponding subset $k \in [1, \mathrm{K}]$, and transformed into a one-hot vector, denoted by $\overline{\boldsymbol{\epsilon}}^{(i)}$, as the final training target. Specifically, the one-hot vector is defined as:

$$\overline{\boldsymbol{\epsilon}}^{(i)} = [\overline{\epsilon}_1^{(i)} \dots \overline{\epsilon}_k^{(i)} \dots \overline{\epsilon}_{\mathrm{K}}^{(i)}]^{\mathrm{T}} \in \mathbb{R}^{\mathrm{K}} \tag{5.7}$$

This process creates a new dataset, denoted by $\overline{\mathscr{D}} = \{\mathbf{x}^{(i)}, \overline{\boldsymbol{\epsilon}}^{(i)}\}_i^{\mathrm{N}}$, consisting of discretized prediction errors represented as one-hot vectors, which serves for training epistemic uncertainty estimator σ_{Θ_2}.

5.3.3.2 Modeling epistemic uncertainty using $\boldsymbol{\epsilon}$ in auxiliary uncertainty estimation

In a Bayesian framework, given an input $\mathbf{x}$, the predictive uncertainty of a DNN is modeled by $\mathrm{P}(y|\mathbf{x}, \mathscr{D})$. Since we have a trained main task DNN, and as proposed in [157], we assume a point-estimate of $\boldsymbol{\omega}$ (denoted as $\hat{\boldsymbol{\omega}}$), then we have:

$$\mathrm{P}(\boldsymbol{\omega}|\mathscr{D}) = \delta(\boldsymbol{\omega} - \hat{\boldsymbol{\omega}}) \rightarrow \mathrm{P}(y|\mathbf{x}, \mathscr{D}) \approx \mathrm{P}(y|\mathbf{x}, \hat{\boldsymbol{\omega}}) \tag{5.8}$$

with δ being the Dirac function.

We then follow the Gaussian assumption, i.e., the prediction is drawn from $\mathscr{N}(y|\mu, \sigma^2)$ and according to the modeling in evidential regression [5], we denote $\boldsymbol{\alpha}$ as the parameters of prior distributions of (μ, σ^2). Following the same work, we first have:

$$\mathrm{P}(\mu, \sigma^2|\mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}}) = \mathrm{P}(\mu|\sigma^2, \mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}})\mathrm{P}(\sigma^2|\mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}}) \tag{5.9}$$

According to Eq. 5.8, we regard the μ depends only on $\mathbf{x}$ and the main task model $\hat{\boldsymbol{\omega}}$:

$$\begin{aligned} \mathrm{P}(\mu, \sigma^2|\mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}}) &= \mathrm{P}(\mu|\mathbf{x}, \hat{\boldsymbol{\omega}})\mathrm{P}(\sigma^2|\mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}}) \\ &= \delta(\mu - f_{\hat{\boldsymbol{\omega}}}(\mathbf{x}))\mathrm{P}(\sigma^2|\mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}}) \end{aligned} \tag{5.10}$$

We introduce $\boldsymbol{\alpha}$ and re-write $\mathrm{P}(y|\mathbf{x}, \hat{\boldsymbol{\omega}})$ in Eq. 5.8 as:

$$\begin{aligned} \mathrm{P}(y|\mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}}) &= \iint \mathrm{P}(y|\mu, \sigma^2)\mathrm{P}(\mu, \sigma^2|\mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}}) d\mu d\sigma^2 \\ &\overset{(a)}{=} \iint \mathrm{P}(y|\mu, \sigma^2)\mathrm{P}(\mu|\sigma^2, \mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}})\mathrm{P}(\sigma^2|\mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}}) d\mu d\sigma^2 \\ &= \iint \mathrm{P}(y, \mu|\sigma^2, \mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}})\mathrm{P}(\sigma^2|\mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}}) d\mu d\sigma^2 \\ &\overset{(b)}{=} \int \delta(\mu - f_{\hat{\boldsymbol{\omega}}}(\mathbf{x})) d\mu \int \mathrm{P}(y|\sigma^2, \mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}})\mathrm{P}(\sigma^2|\mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}}) d\sigma^2 \\ &= \int \mathrm{P}(y|\mathbf{x}, \sigma^2)\mathrm{P}(\sigma^2|\mathbf{x}, \boldsymbol{\alpha}, \hat{\boldsymbol{\omega}}) d\sigma^2 \end{aligned} \tag{5.11}$$

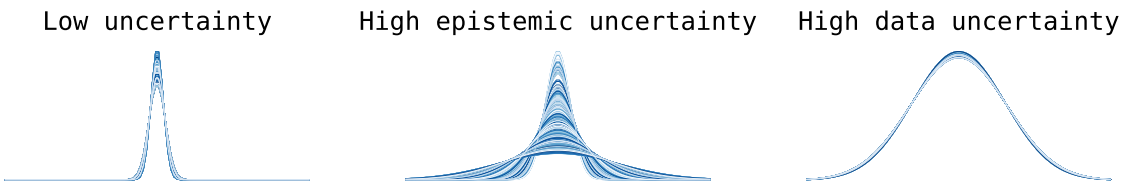


Figure 5.2: Visualizations on desired behaviors of regression results in auxiliary uncertainty estimation scenario. Different types of uncertainty result in different distributions of the variance under Gaussian assumption, which support the link between the posterior over y and ϵ.

where the equality (a) and (b) in Eq. 5.11 are given by Eq. 5.9 and Eq. 5.10, respectively.

In summary, we have

$$\begin{aligned} \mathrm{P}(y|\mathbf{x},\mathscr{D}) &= \iint \mathrm{P}(y|\mathbf{x},\sigma^2)\mathrm{P}(\sigma^2|\boldsymbol{\omega})\mathrm{P}(\boldsymbol{\omega}|\mathscr{D})d\sigma^2 d\boldsymbol{\omega} \\ &= \int \mathrm{P}(y|\mathbf{x},\sigma^2)\mathrm{P}(\sigma^2|\mathscr{D})d\sigma^2 \\ &\overset{(a)}{\approx} \int \mathrm{P}(y|\mathbf{x},\sigma^2)\mathrm{P}(\sigma^2|\mathbf{x},\boldsymbol{\alpha},\hat{\boldsymbol{\omega}})d\sigma^2 \end{aligned} \tag{5.12}$$

where the approximation (a) in Eq. 5.12 is given by Eq. 5.11.

In this case, we first consider ϵ to be drawn from a continuous distribution parameterized by σ^2. Furthermore, we argue that $\mathrm{P}(\sigma^2|\mathscr{D})$ describes the epistemic uncertainty when we have a trained and fixed main task model and the variational approach can be applied [108, 157]: $\mathrm{P}(\sigma^2|\mathbf{x},\boldsymbol{\alpha},\hat{\boldsymbol{\omega}}) \approx \mathrm{P}(\sigma^2|\mathscr{D})$. It shows the special case of the approximation for the posterior over y, where the mean is fixed and only variances differ. Similar to [156], we can illustrate this by using the ensembles of the regression results as in Figure 5.2. The discrepancy in variances determines the epistemic uncertainty of the final prediction. After discretization, we can transform the approximation to $\mathrm{P}(\boldsymbol{\pi}|\mathbf{x},\boldsymbol{\alpha},\hat{\boldsymbol{\omega}}) \approx \mathrm{P}(\boldsymbol{\pi}|\overline{\mathscr{D}})$, with $\overline{\mathscr{D}}$ defined as in Section 5.3.3.1, $\boldsymbol{\pi}$ the parameters of a discrete distribution and $\boldsymbol{\alpha}$ redefined as the prior distribution parameters of this discrete distribution. In the next section, we omit $\hat{\boldsymbol{\omega}}$ and $\mathbf{x}$ for the sake of brevity.

5.3.3.3 Dirichlet posterior for epistemic uncertainty

According to the previous discussions on the epistemic uncertainty modeling and error discretization, we model Dirichlet posterior [33, 108, 209] on the discrete errors $\bar{\boldsymbol{\epsilon}}$ to achieve epistemic uncertainty on the main task.

Intuitively, we consider each one-hot prediction error $\bar{\boldsymbol{\epsilon}}^{(i)}$ to be drawn from a categorical distribution, and $\boldsymbol{\pi}^{(i)} = (\pi_1^{(i)},\ldots,\pi_{\mathrm{K}}^{(i)})$ denotes the random variable over this distribution, where $\sum_{k=1}^{\mathrm{K}} \pi_k^{(i)} = 1$ and $\pi_k^{(i)} \in [0,1]$ for $k \in \{1,...,\mathrm{K}\}$. The conjugate prior of categorical distribution is a Dirichlet distribution:

$$q(\boldsymbol{\pi}^{(i)}|\boldsymbol{\alpha}^{(i)}) = \frac{\Gamma(\mathrm{S}^{(i)})}{\prod_{k=1}^{\mathrm{K}}\Gamma(\alpha_k^{(i)})}\prod_{k=1}^{\mathrm{K}}{\pi_k^{(i)}}^{\alpha_k^{(i)}-1} \tag{5.13}$$

where $\Gamma(\cdot)$ is the Gamma function, $\boldsymbol{\alpha}^{(i)}$ are positive concentration parameters of the Dirichlet distribution and $\mathrm{S}^{(i)} = \sum_{k=1}^{\mathrm{K}} \alpha_k^{(i)}$ the Dirichlet strength.

To get access to the epistemic uncertainty, the categorical posterior $\mathrm{P}(\boldsymbol{\pi}|\overline{\mathscr{D}})$ is needed, yet it is untractable. Monte-Carlo sampling [70] or ensembles [123] can be used to approximate $\mathrm{P}(\boldsymbol{\pi}|\overline{\mathscr{D}})$. However, more computational cost is required. Instead, we use a variational way to learn a Dirichlet distribution in Eq. 5.13 to approximate $\mathrm{P}(\boldsymbol{\pi}|\overline{\mathscr{D}})$ as proposed in [108].

In this case, we let σ_{Θ_2} output the concentration parameters $\boldsymbol{\alpha}$ of $q(\boldsymbol{\pi}|\boldsymbol{\alpha})$, and $\boldsymbol{\alpha}$ will be updated according to the observed inputs. It can also be viewed as collecting the evidence $\boldsymbol{e}$ as a measure for supporting the classification decisions for each class [209], which is equivalent to estimating the Dirichlet posterior. Since the numbers of data points are identical for each class in $\overline{\mathscr{D}}$, and no $\boldsymbol{e}^{(i)}$ output before training, we can set the initial $\boldsymbol{\alpha}$ as $\mathbf{1}$ so that the Dirichlet concentration parameters can be formed as in [34, 209]:

$$\boldsymbol{\alpha}^{(i)} = \boldsymbol{e}^{(i)} + \mathbf{1} = \sigma_{\Theta_2}(\mathbf{x}^{(i)}) + \mathbf{1} \tag{5.14}$$

where $\boldsymbol{e}^{(i)}$ is given by an exponential function on the top of σ_{Θ_2}.

Then we minimize the Kullback-Leibler (KL) divergence between the variational distribution $q(\boldsymbol{\pi}|\mathbf{x}, \Theta_2)$ and the true posterior $\mathrm{P}(\boldsymbol{\pi}|\overline{\mathscr{D}})$ to achieve $\widehat{\Theta}_2$:

$$\begin{aligned}
\widehat{\Theta}_2 &= \underset{\Theta_2}{\operatorname{argmin}} \mathrm{KL}[q(\boldsymbol{\pi}|\mathbf{x}, \Theta_2)||\mathrm{P}(\boldsymbol{\pi}|\overline{\mathscr{D}})] \\
&= \underset{\Theta_2}{\operatorname{argmin}} \int q(\boldsymbol{\pi}|\mathbf{x}, \Theta_2) \log \frac{q(\boldsymbol{\pi}|\mathbf{x}, \Theta_2)}{\mathrm{P}(\boldsymbol{\pi})\mathrm{P}(\overline{\mathscr{D}}|\boldsymbol{\pi})} \\
&= \underset{\Theta_2}{\operatorname{argmin}} -\mathbb{E}_{q(\boldsymbol{\pi}|\mathbf{x}, \Theta_2)}[\log \mathrm{P}(\overline{\mathscr{D}}|\boldsymbol{\pi})] + \mathrm{KL}[q(\boldsymbol{\pi}|\mathbf{x}, \Theta_2)||\mathrm{P}(\boldsymbol{\pi})]
\end{aligned}$$

The loss function will be equivalent to minimizing the negative evidence lower bound [109], considering the prior distribution $\mathrm{P}(\boldsymbol{\pi})$ as $\mathrm{Dir}(\mathbf{1})$:

$$\mathscr{L}(\Theta_2) = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \sum_{k=1}^{\mathrm{K}} \left[\overline{\epsilon}_k^{(i)} (\psi(\mathrm{S}^{(i)}) - \psi(\alpha_k^{(i)})) \right] + \lambda \mathrm{KL}\big(\mathrm{Dir}(\boldsymbol{\alpha}^{(i)})||\mathrm{Dir}(\mathbf{1})\big) \tag{5.15}$$

where ψ is the digamma function, λ is a positive hyperparameter for the regularization term and $\overline{\epsilon}$ is given by Eq. 5.7.

For measuring epistemic uncertainty, we consider using the spread in the Dirichlet distribution [34, 212], which is shown in [212] to outperform other metrics, such as differential entropy. Specifically, the epistemic uncertainty is inversely proportional to the Dirichlet strength:

$$\hat{u}_{\text{epis}}^{(i)} = \sigma_{\widehat{\Theta}_2}(\mathbf{x}^{(i)}) = \frac{\mathrm{K}}{\mathrm{S}^{(i)}} \tag{5.16}$$

σ_{Θ_2} gives more evidence for common patterns, and conversely, less evidence is attached to rare patterns. The class corresponding to the maximum output value from σ_{Θ_2} can also represent the aleatoric uncertainty. However, this is a rough estimate due to quantization error. We employed the expected entropy of Dirichlet outputs as the aleatoric uncertainty estimates and will provide the corresponding results in Section 5.4.5. In this case, we take only σ_{Θ_1} output as the aleatoric uncertainty.

Summary

In conclusion, under the auxiliary uncertainty quantification structure, we propose a generalized AuxUE with two components, namely σ_{Θ_1} and σ_{Θ_2}, to quantify the uncertainty of the prediction given by the main task model. Based on different distribution assumptions on heteroscedastic noise in training data introduced in Sec. 5.3.2, we can train σ_{Θ_1} to estimate aleatoric uncertainty. Meanwhile, as described in Sec. 5.3.3, applying the proposed DIDO on σ_{Θ_2} and measuring the spread of Dirichlet distribution can help to estimate the epistemic uncertainty. Overall, we integrate the optimization for both uncertainty estimators, and the final loss for training the generalized AuxUE is:

$$\mathscr{L}_{\text{AuxUE}} = \mathscr{L}(\Theta_1) + \mathscr{L}(\Theta_2) \tag{5.17}$$

For $\mathscr{L}(\Theta_1)$, in addition to the Gaussian NLL, we will test other NLL loss functions according to different distribution assumptions in the experiment. Furthermore, there are two hyper-parameters, namely the regularization weight λ and the number of classes K. We will talk about the choosing of them in the ablation study in Sec. 5.4.6.

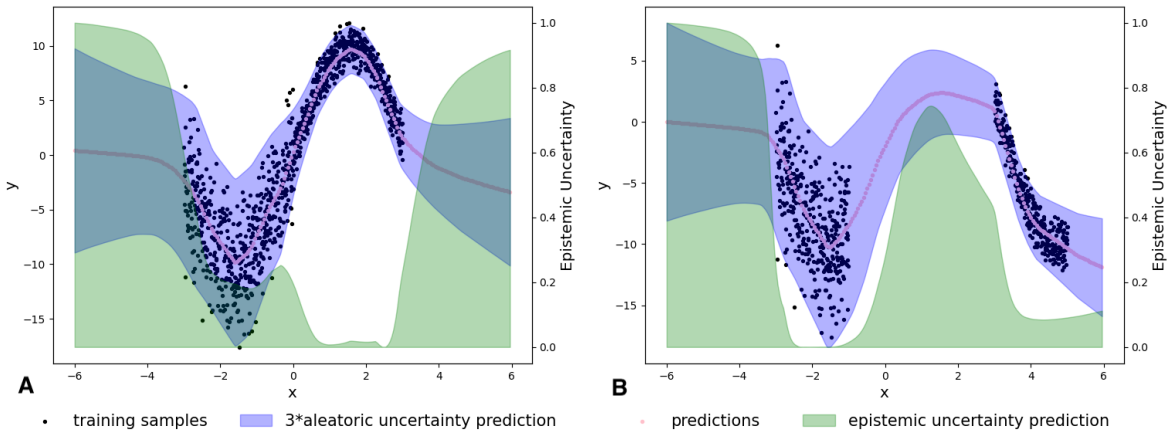


Figure 5.3: Results on 1D toy examples. Aleatoric and epistemic uncertainty estimations given by our proposed AuxUE are presented respectively as the prediction interval and the uncertainty degree (0-1).

5.4 Experiments

In this section, we first show the feasibility of the proposed generalized AuxUE on toy examples. Then, we demonstrate the effectiveness and scalability of epistemic uncertainty estimation using the proposed DIDO on a tabular data example, the age estimation, and the monocular depth estimation (MDE) tasks. We also investigate the robustness of aleatoric uncertainty estimation on super-resolution (SR) and MDE tasks. In the result tables, shar.enc. and sep.enc. denote respectively shared-parameters for the encoders and separate encoders of σ_{Θ_1} and σ_{Θ_2} in the generalized AuxUE.

In the result tables, the top two performing methods are highlighted in colors. All the results are averaged by three runs. The shar.enc. and sep.enc. denote respectively shared-parameters for the encoders and separate encoders of σ_{Θ_1} and σ_{Θ_2} in the generalized AuxUE. For epistemic uncertainty, we compare our proposed method with the solutions based on modified main DNN: LDU [66], Evidential learning (Evi.) [108] and Deep Ensembles (DEns.) [123], as well as training-free methods: Gradient-based uncertainty (Grad.) [92], Variance based on Inject-Dropout (Inject.) [167].

5.4.1 Toy examples: Simple 1D regression

We generate two toy datasets to illustrate uncertainty estimates given by our proposed AuxUE, as shown in Figure 5.3.

Dataset The Figure 5.3 (A) was generated as follows: $y = 10\sin(x) + \epsilon$, with ϵ:

$$\begin{cases} \epsilon \sim \mathcal{N}(0,3) & x \in [-3,0] \\ \epsilon \sim \mathcal{N}(0,1) & x \in [0,3] \\ 0 & \text{otherwise} \end{cases}$$

The Figure 5.3 (B) was generated as follows: $y = 10\sin(x) + \epsilon$, with ϵ:

$$\begin{cases} \epsilon \sim \mathcal{N}(0,3) & x \in [-3,-1] \\ \epsilon \sim \mathcal{N}(0,1) & x \in [3,5] \\ 0 & \text{otherwise} \end{cases}$$

Models Our main task model consists of an MLP with four hidden layers with 300 hidden units per layer and ReLU non-linearities. We use a generalized AuxUE method similar to ConfidNet [41].

Hyperparameters	Main task	AuxUE
learning rate	0.001	0.005
# epochs	200	100
batch size	64	64
λ	-	0.01
K	-	5

Table 5.1: Hyperparameters for 1D signal toy examples.

Hyperparameters	Main task	AuxUE
learning rate	0.001	0.001
# epochs	150	20
batch size	64	64
λ	-	0.0001
K	-	5

Table 5.2: Hyperparameters for tabular data example.

In particular, the input of the AuxUE is the features from the output of the penultimate layer of the main task model. Thus, in this case, the generalized AuxUE does not need the encoders, as shown in the general process in Figure 5.1. The architecture of this AuxUE is as follows. σ_{Θ_1} is composed of one fully connected layer (FCL) with an exponential activation function on the top. σ_{Θ_2} is composed of an MLP with a cosine similarity layer and a hidden layer with 300 hidden units per layer, and an exponential activation function on the top.

The reason for using the cosine similarity layer is to decrease the impact of the numerical value. This operation is similar to the fully connected layer operation but simply divides the output by the product of the norm of the layer's inputs (trainable parameters of the linear and input features).

Training The hyperparameters are listed in Table 5.1. As a reminder, λ and K are the hyperparameters specifically for AuxUE (σ_{Θ_2}), which stand for the weight for the regularization term in the loss function, and respectively for the number of the class we set for discretization.

Discussion In both examples, a tight aleatoric uncertainty estimation is provided on training data areas. For epistemic uncertainty, in Figure 5.3 (A), DIDO provides small uncertainty until reaching the unknown inputs $x \notin [-3,3]$. In Figure 5.3 (B), we report the 'in-between' uncertainty estimates [60]. On the in-between part $x \in [-1,3]$, DIDO can provide higher epistemic uncertainty than in training set regions $x \in [-3,-1]$ and $x \in [3,5]$. In summary, the generalized AuxUE provides reliable uncertainty estimates in regions where training data is either present or absent.

5.4.2 Toy examples: Tabular data

To show the scalability of the proposed DIDO, we conduct the experiment on a tabular dataset.

Dataset We use the red wine quality dataset [44] for the OOD detection task. We randomly separate the dataset in training, validation, and test sets with 72%, 8%, and 20% as the proportions of the whole dataset for each set. We generate the OOD data using the ID test set. We first replicate two test sets as OOD sets, one of which we set all the features in the table to be negative, and the other, we randomly shuffle the values of the features, i.e., we shuffle the values in the columns.

	MSE ↓	AUROC ↑	AUPR ↑
DEns.	**0.646**	0.548	0.250
DIDO	**0.646**	**0.936**	**0.863**

Table 5.3: Main task and OOD detection performance on tabular data example.

Metrics	Coral	CE	CE +SWS	LDU	Evi.	DEns.
MAE ↓	3.47 ± 0.05	3.60 ± 0.02	3.39	3.41	3.70 ± 0.19	**3.31**
RMSE ↓	4.71 ± 0.06	5.03 ± 0.03	4.52 ± 0.03	4.50	4.72 ± 0.23	**4.40**

Table 5.4: Main task performance for ResNet34 model based on different methods on age estimation task. The evaluation is based on AFAD [179] test set.

Models and training Our main task model consists of an MLP with four hidden layers with 16, 32, and 16 hidden units in the respective layer and ReLU non-linearities. We use a generalized AuxUE method similar to ConfidNet [41].

In particular, we find it better to provide the tabular data to the AuxUE directly. We use one hidden layer with 16 hidden units followed by ReLU as the feature extractor for σ_{Θ_1} and σ_{Θ_2} uncertainty estimators. For the uncertainty estimators, we use the same ones as in the 1D signal data. The hyperparameters are listed in Table 5.2.

Results We trained three models to build Deep Ensembles (DEns.) [123]. The epistemic uncertainty estimates are obtained using the variance of DNNs' point estimates. We evaluate the OOD detection performance using AUROC and AUPR as the metrics. The results are shown in Table 5.3. The proposed DIDO outperforms the DEns. on OOD detection task using one extra DNN apart from the main task model.

5.4.3 Age estimation and OOD detection

Epistemic uncertainty estimation for age estimation is similar to one for classification problems but has rarely been discussed in previous works. In this section, we will check OOD detection performance using different approaches based on the age estimator.

Models and training We use (unmodified) official ResNet34 [87] checkpoints from Coral [27] as the main task models. We observe that the age estimation result can outperform the one achieved by Coral by applying soft-weighted-sum (SWS) [253] on the top of the models trained using cross-entropy loss. The goal of SWS is a post-processing operation to transfer the discrete Softmax outputs to the continuous age estimates. For this reason, we use the main task models trained by cross-entropy loss.

Our AuxUE is applied in a ConfidNet [41] style since it is more suitable for image-level tasks. Similarly to the toy example settings, we take the pre-logits (512 features) from the main task model as the inputs of our AuxUE. For σ_{Θ_1}, we use an MLP with one hidden layer with 512 hidden units and an FCL with an exponential function on the top. For σ_{Θ_2}, we use an MLP with a cosine similarity layer and one hidden layer with 512 hidden units per layer and ReLU non-linearities, followed by an FCL with an exponential function on the top.

We choose AFAD [179] dataset as the training set. To train the AuxUE DNN, we use the hyperparameters shown in Table 5.5. We use the same optimizer and batch size as for the main task training, while we use 25 epochs, which is much less than training the main task.

Hyperparameters	Main task	AuxUE
learning rate	0.0005	0.001
# epochs	200	25
batch size	256	256
λ	-	0.01
K	-	8

Table 5.5: Hyperparameters for age estimation.

		AuxUE		Modified main DNN			Training-free	
OOD Dataset	Metrics	Ours σ_{Θ_1}	Ours σ_{Θ_2} DIDO	LDU	Evi.	DEns.	Grad. (inv.)	Inject.
CIFAR10	AUROC ↑	96.0	**100**	95.2	50.0	99.2	**100**	94.5
	AUPR ↑	91.7	**100**	88.3	23.4	95.1	**100**	87.3
SVHN	AUROC ↑	98.3	**100**	94.8	50.0	99.2	**100**	94.0
	AUPR ↑	98.1	**100**	93.2	44.3	97.8	**100**	92.5
MNIST	AUROC ↑	97.8	**100**	97.6	50.0	99.6	**100**	98.8
	AUPR ↑	93.9	**100**	93.8	23.4	97.2	**100**	96.9
Fashion MNIST	AUROC ↑	97.7	**100**	95.6	50.0	99.1	**100**	97.7
	AUPR ↑	94.0	**100**	89.3	23.4	93.8	**100**	94.2
Oxford Pets	AUROC ↑	**82.9**	55.9	31.5	50.1	56.1	50.7	48.6
	AUPR ↑	**53.3**	23.9	12.5	18.5	21.3	19.6	20.3
Fake Data	AUROC ↑	67.0	**80.8**	70.0	50.0	33.2	45.9	45.1
	AUPR ↑	59.7	**70.2**	58.8	49.5	37.8	46.3	44.6

Table 5.6: OOD detection results on Age estimation task. ID data is from Asian Face Age Dataset (AFAD) [179].

Evaluation settings There is no benchmark for OOD detection on age estimation, the straightforward OOD inputs are irrelevant compared to the training images [223]. Thus, we use CIFAR10 [120], SVHN [177], MNIST [126], FashionMNIST [242], Oxford-Pets [186] and Noise image generated by Pytorch [187] (FakeData) as the OOD datasets. We employ the area under the receiver operating characteristic (***AUROC***) and the precision-recall curve (***AUPR***) (higher is better for both) to evaluate OOD detection performance.

Results OOD detection results are shown in Table 5.6. DIDO performs the best on most datasets. The training-free methods also perform well, but we observe that the Grandient-based solution needs inversed uncertainty (inv.) to provide better performance. On the Pets dataset, DIDO performs worse than DEns. and aleatoric uncertainty estimation head σ_{Θ_1}. We argue that images of pets provide features closer to facial information, resulting in higher evidence estimates given by DIDO. While σ_{Θ_1} performs better in this case, which can jointly make AuxUE a better uncertainty estimator. Overall, we consider that using generalized AuxUE with DIDO is an alternative that can better detect OOD inputs than ensembling-based solutions. Ensembling-based solutions still have advantages, such as better main task accuracy, as shown in Table 5.4.

Additional: Main task performance For the age estimation task, we list the main task results in Table 5.4 given by the original Coral, the original cross entropy (CE)-based models and the CE-based models using soft-weighted-sum (SWS). We can see that SWS really improves the main task

performance. Furthermore, by adjusting the original model to output the parameters of Gaussian distribution [113, 180] and training three models like this from scratch, we can achieve the results given by Deep Ensembles (DEns.) [123]. We also implement LDU [66] and Evidential learning (Evi.) [108] based on the ResNet34 backbone. The overall difference among different techniques is not huge, while the adjustments still reduce a bit the age estimation performance. We argue that the adjusted DNNs might achieve comparable performance to the unchanged ones, but more tuning and hyperparameter searching should be required.

5.4.4 Super-resolution task

In the SR task, the noise in the reconstructed image will be irreducible given the noisy low-resolution input, and we consider this uncertainty to be aleatoric. Moreover, we argue that the definition of epistemic uncertainty is rather vague in this task. Therefore, in this section, we use AuxUE to estimate the aleatoric uncertainty based on different distribution assumptions. We choose SRGan [128] as the main task model and BayesCap [229] as the AuxUE and follow the same training and evaluation settings as in [229]. The goal is to analyze the fundamental performance and robustness of aleatoric uncertainty estimation under different distribution assumptions. We choose simple Gaussian (Sgau) [180], Laplacian (Lap), Generalized Gaussian (Ggau) [229] and Normal-Inverse-Gamma (NIG) [5] distributions on BayesCap [229]. We modify the loss functions and the head of the Bayescap to output the desired parameters of the distributions.

Evaluation metrics We follow [229] to use the Uncertainty Calibration Error (***UCE***, lower is better) metric [125]. It measures the difference between the predicted uncertainty and the prediction error. Specifically, the prediction error and estimated uncertainty are assigned into bins, and the absolute difference between the mean prediction error and mean estimated uncertainty in each bin is calculated. UCE is the sum of the results from all bins.

Models and training In the following, our goal is to find a more robust design based on different distribution assumptions. Meanwhile, we find that the definition of epistemic uncertainty is vague for the super-resolution task. Thus we only have σ_{Θ_1} in our generalized AuxUE, and we take directly BayesCap as σ_{Θ_1} with some minor modifications for different distribution assumptions. We follow [229] to use the output of the main task model SRGan [128] as the input of the AuxUE.

We only modify the prediction heads on Bayescap. Original BayesCap [229] uses multiple Residual blocks [87] followed by three heads which output the three parameters for the Generalized Gaussian distribution, including one as the refined main task prediction. Each head contains a set of convolutional layers + PReLU activation functions. As we apply different distribution assumptions, we use the different numbers of the same heads to construct the variants of BayesCap. Specifically, we use two heads for two Gaussian distribution parameters, two heads for two Laplace distribution parameters, and four heads for four parameters in NIG distribution. We follow the same training settings (batch size, learning rate, weight for the additional identity mapping loss, and the number of epochs) as in the original paper [229].

Datasets We use ImageNet [48] as the training set for both SRGan and BayesCap models. For uncertainty evaluation, we use Set5 [14], Set14 [258], and BSDS100 [162] as the testing sets. Moreover, we generate Set5-C, Set14-C, and BSDS100-C using the code of ImageNet-C [90] to have different corruptions on the images. We apply the following eighteen perturbations with five severities: Gaussian noise, shot noise, impulse noise, iso noise, defocus blur, glass blur, motion blur, zoom blur, frost, fog, snow, dark, brightness, contrast, pixelated, elastic, color quantization, and JPEG. Only low-resolution images (inputs) are polluted by noise, while the corresponding high-resolution ground truth images are clean. Castillo et al. [29] applied the noise to the input images during training, while we apply them during inference for robust uncertainty estimation evaluation.

Metrics	Set5	Set14	BSDS100
PSNR ↑	29.40	26.02	25.16
SSIM ↑	0.8472	0.7397	0.6688

Table 5.7: Main task performance for SRGan model on super-resolution task.

Methods	AbsRel ↓	log10 ↓	RMSE ↓	SqRel ↓	RMSElog ↓	δ1 ↑	δ2 ↑	δ3 ↑
Org	**0.056**	**0.025**	2.430	**0.201**	**0.089**	**0.963**	0.994	**0.999**
SinglePU	0.065	0.029	2.606	0.234	0.100	0.952	0.993	0.998
LDU	0.059	0.026	**2.394**	0.203	0.091	0.960	0.994	**0.999**
DEns.	0.060	0.026	2.435	0.202	0.092	0.961	**0.995**	**0.999**

Table 5.8: Main task performance for original and modified BTS models on monocular depth estimation. The evaluation is based on KITTI [75] Eigen-split [57] validation set.

Results In the super-resolution task, we take the main task SRGan [128] model used in BayesCap [229]. Thus we have the same main task performance as they showed in the paper. We list the results in Table 5.7 as a reminder. The evaluation is based on Set5 [14], Set14 [258], BSDS100 [162] dataset. For uncertainty quantification performance, as shown in Table 5.9, the Laplacian assumption on the data-dependent noise performs better than all the other assumptions, including the Generalized Gaussian distribution proposed in BayesCap. When the noise severity increases, using the Laplacian assumption can provide more robust uncertainty than the others.

5.4.5 Monocular depth estimation task

For the MDE task, we will evaluate both aleatoric and epistemic uncertainty estimation performance based on the AuxUE SLURP [252]. Our generalized AuxUE is also constructed using SLURP as the backbone. We use BTS [130] as the main task model and KITTI [75, 227] Eigen-split [57] training set for training both BTS and AuxUE models.

5.4.5.1 Aleatoric uncertainty estimation

In this section, we will analyze the performance of the same distribution assumptions as in Sec. 5.4.4.

Models and training We use SLURP [252] as the backbone in this experiment. We modify the prediction heads to achieve the uncertainty estimates.

For σ_{Θ_1}, we do not modify the model when the distribution assumption only contains one parameter (except for the main task prediction term). For the distribution assumptions with more than one parameter output, we add one more convolutional layer with ReLU on the top for a fair comparison.

For σ_{Θ_2}, similarly to the one in age estimation, we replace the original head (a single convolutional layer) with the cosine similarity layer followed by two convolutional layers with ReLU activation functions. In the cases where we share the encoders to make the general AuxUE lighter, based on the original SLURP, we doubled the number of features fed into the prediction head. We split them into two sets, feeding them into two prediction heads. The two prediction heads are consistent in the structures mentioned before. We follow [252] to use the depth output and the encoder features of the main task model BTS [130] as the input of the AuxUE.

The hyperparameters used during training are listed in Table 5.11. The learning rate decrement is consistent with the main task BTS [130] model.

Super Resolution (Metric: UCE ↓)

Dataset	S	Original (Ggau)	+ Sgau	+ NIG Uncer 1	+NIG Uncer 2	Ours σ_{Θ_1} (+ Lap)
Set5	0	0.0088	0.0083	**0.0018**	0.0025	0.0019
	1	0.0186	0.0180	**0.0156**	0.0171	0.0157
	2	0.0253	0.0243	**0.0226**	0.0244	0.0227
	3	0.0363	0.0341	0.0333	0.0351	**0.0332**
	4	0.0434	0.0394	0.0392	0.0415	**0.0389**
	5	0.0525	0.0462	0.0464	0.0056	**0.0040**
Set14	0	0.0137	0.0092	**0.0040**	0.0056	**0.0040**
	1	0.0221	0.0195	0.0176	0.0198	**0.0174**
	2	0.0281	0.0255	0.0241	0.0265	**0.0240**
	3	0.0350	0.0318	0.0310	0.0334	**0.0308**
	4	0.0408	0.0368	0.0364	0.0391	**0.0362**
	5	0.0509	0.0465	0.0465	0.0494	**0.0461**
BSDS 100	0	0.0124	0.0071	0.0036	0.0048	**0.0033**
	1	0.0204	0.0174	0.0162	0.0180	**0.0160**
	2	0.0271	0.0237	0.0229	0.0249	**0.0227**
	3	0.0332	0.0288	0.0286	0.0305	**0.0358**
	4	0.0425	0.0363	0.0363	0.0385	**0.0358**
	5	0.0539	0.0459	0.0460	0.0482	**0.0453**

Table 5.9: Aleatoric uncertainty estimation results on Super-Resolution task. Datasets with an S (severity) greater than 1 are the -C variants of the corresponding clean datasets.

Evaluation metrics We first build Sparsification curves (SC) [23]: we achieve predictive SC by computing the prediction error of the remaining pixels after removing a certain partition of pixels (5% in our experiment) each time according to the highest uncertainty estimations. We can also obtain an Oracle SC by removing the pixels according to the highest prediction errors. Then, we have the same metrics used in [191] to measure: 1. the difference between predictive SC and the Oracle SC: Area Under the Sparsification Error (***AUSE***, lower is better). 2. the difference between predictive SC and a random SC (no-modeling uncertainty): Area Under the Random Gain (***AURG***, higher is better). We choose absolute relative error (REL) and root mean square error (RMSE) as the prediction error metrics.

Datasets We generate KITTI-C from KITTI Eigen-split validation set [57] using the same steps as in Sec. 5.4.4 and take it along with the original KITTI for evaluation.

Results In monocular depth estimation, we list in Table 5.8 the results for the methods using modified main task BTS [130] models, namely SinglePU [113], Deep Ensembles (DEns.) [123], LDU [66], as well as the original model, which is used for AuxUEs and the training-free methods. Note that we use the evaluation code based on AdaBins [15], which corrected the error made in the BTS evaluation code, and the result will be slightly better than the one claimed in the original BTS. The evaluation is based on KITTI [75] Eigen-split [57] validation set. As we can see, modifying the model and training in [113] way will affect the main task performance even after doing Deep Ensembles, LDU can provide competitive results to the original, yet only on several metrics. Overall, the AuxUE is necessary to be applied for uncertainty estimation without changing and affecting the main task.

For aleatoric uncertainty estimation, as shown in Table 5.10, the results follow the SR task. The Laplacian assumption is more robust when the severity increases, while the Gaussian one works bet-

Monocular Depth Estimation

S	Metrics	Original	+ Ggau	+ Sgau	+ NIG Uncer1	+ NIG Uncer2	Ours (DIDO) sep. enc. σ_{Θ_2}	Ours (+ Lap) shar. enc. σ_{Θ_1}	Ours (+ Lap) sep. enc. σ_{Θ_1}
0	AUSE-REL ↓	0.013	0.014	0.013	**0.012**	0.013	0.050	0.013	0.013
	AUSE-RMSE ↓	0.204	0.258	**0.202**	0.208	0.205	3.277	0.205	0.203
	AURG-REL ↑	0.023	0.023	0.023	**0.024**	0.023	-0.016	0.023	0.023
	AURG-RMSE ↑	1.869	1.815	**1.871**	1.865	1.868	-1.453	1.869	1.870
1	AUSE-REL ↓	0.019	0.021	0.019	**0.018**	0.020	0.064	**0.018**	0.019
	AUSE-RMSE ↓	0.340	0.482	**0.332**	0.335	0.350	4.085	**0.332**	0.336
	AURG-REL ↑	0.031	0.029	0.031	**0.032**	0.030	-0.014	**0.032**	0.031
	AURG-RMSE ↑	2.357	2.215	**2.365**	2.362	2.347	-1.388	**2.365**	2.361
2	AUSE-REL ↓	0.024	0.026	0.023	**0.022**	0.025	0.077	**0.022**	0.023
	AUSE-RMSE ↓	0.483	0.707	**0.463**	0.479	0.505	4.788	0.464	0.468
	AURG-REL ↑	0.038	0.035	**0.039**	**0.039**	0.037	-0.016	**0.039**	0.038
	AURG-RMSE ↑	2.759	2.535	**2.779**	2.763	2.737	-1.546	2.777	2.774
3	AUSE-REL ↓	0.033	0.036	**0.031**	**0.031**	0.035	0.099	**0.031**	**0.031**
	AUSE-RMSE ↓	0.795	1.176	0.737	0.806	0.846	5.743	0.749	**0.730**
	AURG-REL ↑	0.047	0.044	**0.049**	**0.049**	0.045	-0.019	**0.049**	**0.049**
	AURG-RMSE ↑	3.243	2.862	3.301	3.232	3.192	-1.705	3.289	**3.308**
4	AUSE-REL ↓	0.056	0.057	0.050	0.053	0.059	0.125	0.051	**0.049**
	AUSE-RMSE ↓	1.517	2.380	1.364	1.582	1.607	5.743	1.430	**1.268**
	AURG-REL ↑	0.051	0.051	0.058	0.054	0.049	-0.019	0.056	**0.059**
	AURG-RMSE ↑	3.680	2.817	3.834	3.615	3.590	-1.705	3.767	**3.929**
5	AUSE-REL ↓	0.071	0.082	0.064	0.069	0.071	0.140	0.066	**0.059**
	AUSE-RMSE ↓	2.202	3.878	2.043	2.414	2.307	7.354	2.157	**1.760**
	AURG-REL ↑	0.056	0.045	0.063	0.057	0.055	-0.014	0.061	**0.067**
	AURG-RMSE ↑	4.054	2.377	4.213	3.842	3.949	-1.098	4.098	**4.496**

Table 5.10: Aleatoric uncertainty estimation results on Monocular Depth Estimation task. Datasets used in MDE are KITTI (S=0) and KITTI–C (S>0). The top two are respectively bolded and underlined. All the results are averaged by three models.

ter when the noise severity is smaller. We also check the proposed generalized AuxUE with a shared encoder. It shows that the epistemic uncertainty estimation branch affects the robustness of aleatoric uncertainty estimation in this case, especially when encountering stronger noise. Furthermore, using the expected entropy of Dirichlet outputs, i.e., outputs of σ_{Θ_2} can achieve aleatoric uncertainty estimates theoretically. We also provide the partial results from the other distribution assumptions to make a comparison. From the last column, we can see that the Dirichlet outputs cannot provide comparable results on quantitative metrics since the discretization affects the original numerical values of the prediction errors on the ID training data.

In the next experiment, we will evaluate epistemic uncertainty estimation quality using DIDO. The experiments are conducted from two perspectives. Firstly, we verify if DIDO can better help the model identify the dataset change. Secondly, we check if DIDO can detect well the patterns that are rarely seen during training. The comparison is composed of the proposed DIDO and the solutions based on modified main task models: LDU [66], Evidential regression (Evi.Reg.) [5] and Deep Ensembles (DEns.) [123] and training-free uncertainty estimation: Gradient-based uncertainty (Grad.) [92],

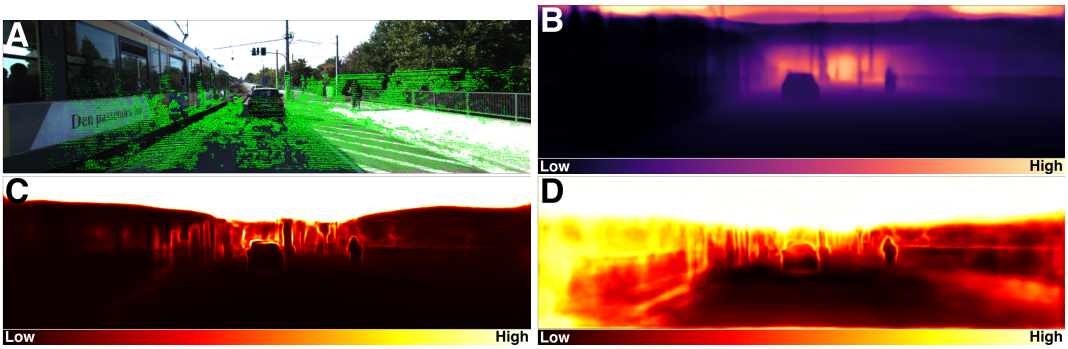


Figure 5.4: Illustrations of uncertainty estimations for MDE task. A: input image, green points represent pixels with depth groundtruth; B: depth prediction; C and D: aleatoric and epistemic uncertainty estimations. The areas lacking depth groundtruth, e.g., sky and tramway, are assigned high uncertainty using DIDO.

Hyperparameters	Main task	AuxUE
start learning rate	1e-4	1e-4
end learning rate	1e-5	1e-5
# epochs	50	8
batch size	4	4
λ	-	0.01
K	-	32

Table 5.11: Hyperparameters for monocular depth estimation.

Variance based on Inject-Dropout (Inject.Var.) [167]. Furthermore, we also verify whether aleatoric uncertainty methods based on different distribution assumptions as in Sec. 5.4.4 and Sec. 5.4.5.1 can generalize to the OOD data, i.e., provide high uncertainty to the unseen patterns, even without explicitly modeling epistemic uncertainty.

5.4.5.2 Epistemic uncertainty estimation: robustness under dataset change

This experiment will discuss the predictive uncertainty performance encountering the dataset change. Supervised MDE is an ill-posed problem that heavily depends on the training dataset. In our case, the main task model is trained on the KITTI dataset, so the model will output meaningless results on the indoor data, which should trigger a high uncertainty estimation. The results are shown in Table 5.12.

Evaluation settings and datasets We take ***AUROC*** and ***AUPR*** (higher is better for both) as evaluation metrics. We take all the valid pixels from the KITTI validation set (ID) as the negative samples and the valid pixels from the NYU [176] validation set (OOD) as the positive samples.

Results Table 5.12 shows whether different uncertainty estimators can give correct indications facing the dataset change. Generalized Gaussian and Gradient-based methods can provide competitive results, while our method, especially DIDO, provides the best performance.

5.4.5.3 Epistemic uncertainty estimation: Robustness on unseen patterns during training

This experiment will focus on how uncertainty estimators behave on unseen patterns during training. The unseen patterns are drawn from the same dataset distribution as the patterns used in training, while the outputs of the main task model for such patterns may be reasonable. Still, they cannot be evaluated and thus are not reliable. The uncertainty estimators should assign high uncertainty to these predictions. Since this topic is rarely considered in MDE, we try to give a benchmark in this work.

Metrics	Auxiliary Uncertainty Estimator (SLURP)								Modified main task model					Training-free	
	Original	+ Ggau	+ Sgau	+ NIG Uncer1	+ NIG Uncer2	Ours σ_{Θ_1} sep. enc.	Ours σ_{Θ_2} sep. enc.	Ours σ_{Θ_2} shar. enc.	Single PU	LDU	Evi.Reg. Uncer1	Evi.Reg. Uncer2	DEns.	Grad. (inv.)	Inject. Var.
AUROC ↑	59.8	80.9	74.5	57.0	63.6	65.4	98.1	**98.4**	64.2	58.1	43.4	70.6	62.1	78.4	18.3
AUPR ↑	76.7	90.9	88.4	75.5	78.8	82.5	99.3	**99.4**	78.3	79.5	63.5	77.8	76.7	92.6	62.3

Table 5.12: Epistemic uncertainty estimation results encountering dataset change on Monocular depth estimation task. The evaluation dataset used here is NYU indoor depth dataset.

S	Metrics	Auxiliary Uncertainty Estimator (SLURP)								Modified main task model					Training-free	
		Original	+ Ggau	+ Sgau	+ NIG Uncer1	+ NIG Uncer2	Ours σ_{Θ_1} sep. enc.	Ours σ_{Θ_2} sep. enc.	Ours σ_{Θ_2} shar. enc.	Single PU	LDU	Evi.Reg. Uncer1	Evi.Reg. Uncer2	DEns.	Grad.	Inject. Var.
0	Sky-AUROC ↑	99.1	96.8	99.0	90.9	78.5	99.9	**100.0**	99.9	89.0	96.5	72.2	76.7	93.5	85.6	58.4
	Sky-AUPR ↑	94.6	80.3	91.6	57.6	39.2	99.7	**100.0**	99.0	62.0	93.8	48.8	42.6	70.0	76.3	28.1
	Sky-Overall ↓	0.643	0.277	0.934	0.983	0.999	0.961	0.015	0.018	0.005	0.278	0.986	0.986	0.005	**0.001**	0.800
1	Sky-AUROC ↑	98.4	96.0	99.0	90.4	76.2	99.8	**100.0**	99.9	86.9	96.3	65.2	69.7	92.8	76.9	58.5
	Sky-AUPR ↑	93.3	77.9	92.4	57.4	37.7	99.5	**99.9**	98.9	59.1	93.5	43.3	37.4	68.0	69.8	28.2
	Sky-Overall ↓	0.742	0.274	0.935	0.978	0.999	0.962	0.016	0.018	0.005	0.277	0.988	0.988	0.005	**0.002**	0.799
2	Sky-AUROC ↑	97.6	95.6	99.0	90.2	74.9	99.7	**99.9**	**99.9**	86.6	95.9	60.3	65.4	92.3	75.6	58.4
	Sky-AUPR↑	91.9	76.5	92.8	57.5	37.2	99.3	**99.8**	98.8	58.9	93.0	39.4	34.5	67.0	67.8	28.1
	Sky-Overall ↓	0.784	0.274	0.937	0.973	0.999	0.962	0.017	0.018	0.005	0.280	0.990	0.990	0.005	**0.002**	0.803
3	Sky-AUROC ↑	96.8	95.0	98.9	90.0	73.7	99.4	**99.9**	99.7	86.6	95.9	56.3	62.3	91.6	73.6	58.4
	Sky-AUPR ↑	90.5	75.1	92.9	58.2	36.9	98.9	**99.7**	98.1	59.5	92.8	36.6	32.8	65.7	64.5	28.2
	Sky-Overall ↓	0.815	0.277	0.938	0.965	0.999	0.960	0.018	0.020	0.005	0.283	0.991	0.992	0.005	**0.002**	0.809
4	Sky-AUROC ↑	94.9	93.2	98.5	89.8	72.9	99.0	**99.6**	99.5	87.2	96.1	51.2	58.8	91.8	71.3	58.4
	Sky-AUPR ↑	87.2	71.1	92.4	59.8	37.9	98.2	**99.1**	97.2	61.7	92.9	32.8	31.2	67.2	60.0	28.3
	Sky-Overall ↓	0.868	0.284	0.940	0.945	0.994	0.959	0.023	0.022	0.005	0.288	0.994	0.994	0.005	**0.002**	0.819
5	Sky-AUROC ↑	92.5	90.3	97.6	89.6	73.4	98.2	98.5	**99.0**	87.5	96.5	48.2	58.5	92.2	66.8	57.8
	Sky-AUPR ↑	83.8	66.6	91.4	63.6	42.1	96.8	**97.1**	96.1	64.6	93.7	31.6	32.8	70.4	53.8	28.2
	Sky-Overall ↓	0.902	0.299	0.943	0.909	0.982	0.959	0.035	0.026	0.005	0.295	0.995	0.996	0.005	**0.002**	0.839

Table 5.13: Epistemic uncertainty estimation results encountering unseen pattern on Monocular depth estimation task. The evaluation datasets used here are KITTI Seg-Depth (S=0) and KITTI Seg-Depth-C (S>0).

Evaluation settings and datasets We select sky areas in KITTI as OOD patterns. This setting is based on the following reasons. Due to the generalization ability of MDE DNNs, it is inappropriate to treat all pixels without ground truth as OOD. However, there is consistently no ground truth for the sky parts since LIDAR is used in depth acquisition. During training, sky patterns are masked and never seen by the DNNs (including the AuxUEs). Meanwhile, they are annotated in the KITTI semantic segmentation dataset [3] (200 images), thus can be used for evaluation.

Three metrics are applied for evaluating OOD detection performance as shown in Table 5.13. ***Sky-AUROC*** and ***Sky-AUPR*** (higher is better for both): we select 49 images that are not in the training set and have both depth and semantic segmentation annotations. For each image, we take the sky pixels as the positive class and the pixels with depth ground truth as the negative class. We use AUROC and AUPR to assess the uncertainty estimation performance. Note that this metric does not guarantee that the uncertainty of the sky is the largest in the whole uncertainty map. Thus, we have ***Sky-Overall*** (lower is better): all 200 images with semantic segmentation annotations are selected for evaluation. The ground truth uncertainties are set as $\mathbf{1}$ for the sky areas. Then we normalize the predicted uncertainty, take the sky areas $\hat{\mathbf{u}}_{\text{sky}}$ from the whole uncertainty map and measure: $mean((\mathbf{1}-\hat{\mathbf{u}}_{\text{sky}})^2)$. For simplicity, we denote KITTI Seg-Depth for both evaluation datasets, and we also generate KITTI Seg-Depth-C following Sec. 5.4.4.

Results Figure 5.4 shows a qualitative example of typical uncertainty maps computed on KITTI images. More visualizations are presented in Sec. 5.5. In Table 5.13, the Deep Ensembles and Gradient-based methods can better assign consistent and higher uncertainty to the sky areas, but they are unsatisfied in identifying the ID and OOD areas. Most distribution assumptions can help AuxUE achieve good AUROC and AUPR results, which shows that these AuxUEs all fit the ID data well. Yet, they can not assign consistent and higher uncertainty to the sky areas. Our DIDO can achieve a balanced performance on all the metrics, and at the same time, it maintains robust performance in the presence of noise.

5.4.6 Ablation study

The ablation studies are based on the monocular depth estimation task and arranged as follows:

1. **Hyperparameters:** We analyze the effect of the number of sets K defined in Sec. 5.3.3.1 for discretization and λ for the regularization term in Eq. 5.15.
2. **Necessity of using AuxUE:** We also apply DIDO on the main task model to check the impact on main task performance.
3. **Effectiveness of Dirichlet modeling:** We show the effectiveness of the Dirichlet modeling instead of using the normal Categorical modeling based on the discretized prediction errors. For the former, we apply classical cross-entropy on the Softmax outputs given by the AuxUE.

5.4.6.1 Hyperparameters

There are two main hyperparameters in our proposed DIDO: the number of the sets K in discretization and λ for the regularization term in the loss function as shown in Eq. 5.15. In this section, we analyze the effect of these two hyperparameters.

The evaluations are based on epistemic uncertainty estimation on unseen patterns and dataset change detection.

The effect of K is shown in Figure 5.5a and Figure 5.5c. We test $K = \{8, 16, 32, 64\}$ and λ is fixed to 0.01. The Sky-AUROC and Sky-AUPR performance decrease when we have bigger K, while on the Sky-Overall metric, bigger K provides better results. When the evaluation dataset is changed from KITTI to NYU [176], bigger K can also provide better AUROC and AUPR in identifying the change. We choose $K = 32$ to have a balanced performance.

The effect of λ is shown in Figure 5.5b and Figure 5.5d. We test $\lambda = \{1e-4, 1e-3, 1e-2, 1e-1\}$ and K is fixed to 32. We can see the Sky-AUROC and Sky-AUPR performance decrease when we use bigger λ during training, while on the Sky-Overall metric, it performs better when using bigger λ during training. For the dataset change experiment, we can see when $\lambda = 0.01$, the DIDO can provide the best AUROC and AUPR. We choose $\lambda = 0.01$ in the end for our model.

5.4.6.2 Necessity of using AuxUE

In this experiment, we apply DIDO on the main task BTS [130] model to see the impact on the main task performance and the uncertainty estimation performance. The comparison will only be conducted with the other modified main task models for fairness. In particular, we first adjust the BTS model to Single Predictive Uncertainty [113, 180] variant (BTS-SinglePU), i.e., we first add an aleatoric uncertainty estimation head parallel to and identical to the depth estimation head on top of the original model. Then we add the same head for DIDO applied on σ_{Θ_2}. Thus there are three prediction heads on the modified BTS model corresponding to depth prediction, aleatoric uncertainty estimation and epistemic uncertainty estimation. We denote this variant as BTS-DIDO. We will compare BTS-DIDO with the original BTS model (Org) and the BTS-SinglePU model to check the

impact on the main task. We also compare the BTS-DIDO with AuxUE + original BTS, BTS-DEns. and BTS-SinglePU models to check the uncertainty estimation performance.

To train the BTS-SinglePU models, we follow the original BTS settings for hyperparameters in training. We change the loss function to Gaussian NLL loss. For BTS-DIDO, we use the same hyperparameters as we used in AuxUEs for DIDO modeling, i.e., $K = 32$ and $\lambda = 0.01$. For the other hyperparameters, such as the batch size and learning rate, we follow the original BTS settings.

During training, we found that combining DIDO directly with the BTS will make the training unstable: the loss will explode after around fifteen epochs. As shown in Table 5.14, for the main task performance, the original BTS can outperform the others even for BTS-DEns. We argue that there are two reasons that might result in the performance reduction: BTS-DEns. component models (BTS-SinglePU models) are adjusted for the uncertainty output; SiLog loss [57], which is specifically applied to the MDE task, is replaced by the Gaussian negative log-likelihood loss. However, when the noise severity increases ($S > 3$), BTS-DIDO and BTS-DEns. can perform better than the others. In particular, BTS-DIDO shows a more robust performance given the inputs with heavy perturbations.

For the uncertainty estimation performance, as shown in Table 5.15, BTS-DIDO and AuxUE achieve similar performance on Sky-AUROC and Sky-AUPR metrics and on dataset change detection. While on Sky-Overall, AuxUE works slightly better than BTS-DIDO. For aleatoric uncertainty, since the main task performances for different models are different, the comparison can only be a reference. With a sacrifice on the main task performance, BTS-DIDO has the potential to achieve good uncertainty estimation performance.

We argue that it is necessary to use AuxUE to keep the main task performance when the input is relatively clean. Meanwhile, the good performance on BTS-DIDO under high severity perturbations makes it meaningful to work on stabilizing the training for DIDO-based models in the future.

5.4.6.3 Effectiveness of Dirichlet modeling

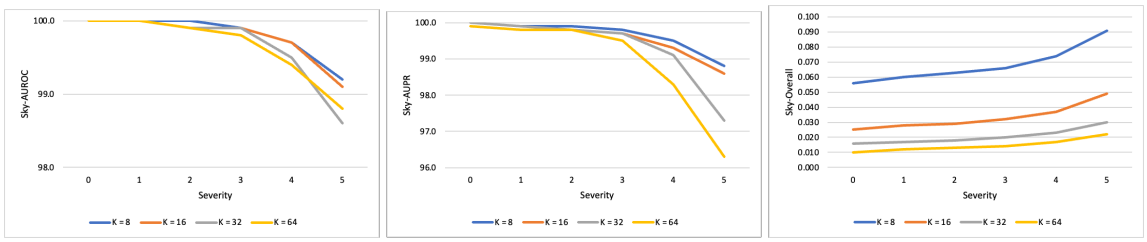
We show the effectiveness of the Dirichlet modeling instead of using the standard categorical modeling based on discretized prediction errors. For categorical modeling, we also choose $K = 32$ classes for discretization. We change the activation function on the top of σ_{Θ_2} from the ReLU function to the Softmax function, then apply classical cross-entropy on the Softmax outputs. For measuring uncertainty, we use the Shannon-Entropy [211] on the Softmax outputs. As shown in Figure 5.6, the Dirichlet modeling outperforms the Categorical modeling on all three metrics w.r.t. the OOD pattern detection. On the dataset change experiment, Categorial modeling provides 90.37 for AUROC and 96.83 for AUPR, which underperforms the results given by Dirichlet modeling. This study shows the effectiveness of DIDO and the use of evidential learning in the AuxUE.

S	Methods	Main task performance							
		AbsRel ↓	log10 ↓	RMSE ↓	SqRel ↓	RMSElog ↓	δ1 ↑	δ2 ↑	δ3 ↑
0	Org + AuxUE	**0.056**	**0.025**	**2.430**	**0.201**	**0.089**	**0.963**	0.994	**0.999**
	BTS-SinglePU	0.065	0.029	2.606	0.234	0.100	0.952	0.993	0.998
	BTS-DEns.	0.060	0.026	2.435	0.202	0.092	0.961	**0.995**	**0.999**
	BTS-DIDO	0.061	0.027	2.574	0.236	0.098	0.954	0.992	0.998
1	Org + AuxUE	**0.077**	**0.036**	**3.185**	**0.370**	**0.129**	**0.919**	**0.977**	**0.992**
	BTS-SinglePU	0.094	0.043	3.581	0.476	0.149	0.890	0.969	0.989
	BTS-DEns.	0.087	0.040	3.415	0.422	0.138	0.902	0.974	**0.992**
	BTS-DIDO	0.088	0.040	3.453	0.456	0.143	0.898	0.972	0.991
2	Org + AuxUE	**0.096**	**0.047**	**3.861**	**0.571**	**0.168**	**0.876**	**0.954**	0.979
	BTS-SinglePU	0.116	0.057	4.359	0.735	0.192	0.835	0.939	0.973
	BTS-DEns.	0.109	0.053	4.189	0.661	0.178	0.848	0.947	0.979
	BTS-DIDO	0.108	0.051	4.169	0.670	0.178	0.851	0.948	**0.980**
3	Org + AuxUE	**0.130**	0.069	**4.905**	**0.985**	0.237	**0.805**	0.908	0.949
	BTS-SinglePU	0.149	0.078	5.357	1.140	0.253	0.760	0.890	0.944
	BTS-DEns.	0.140	0.073	5.184	1.031	0.234	0.772	0.904	0.955
	BTS-DIDO	0.134	**0.067**	5.134	1.003	**0.228**	0.789	**0.912**	**0.961**
4	Org + AuxUE	0.195	0.117	6.591	1.888	0.370	0.680	0.808	0.874
	BTS-SinglePU	0.195	0.110	6.649	1.786	0.341	0.662	0.816	0.894
	BTS-DEns.	0.186	0.103	6.485	1.649	0.317	0.667	0.833	0.911
	BTS-DIDO	**0.170**	**0.089**	**6.292**	**1.485**	**0.293**	**0.711**	**0.862**	**0.930**
5	Org + AuxUE	0.265	0.172	8.259	2.932	0.508	0.555	0.696	0.783
	BTS-SinglePU	0.231	0.135	7.731	2.328	0.410	0.585	0.757	0.853
	BTS-DEns.	0.222	0.127	7.584	2.190	0.386	0.587	0.772	0.871
	BTS-DIDO	**0.211**	**0.116**	**7.484**	**2.065**	**0.367**	**0.621**	**0.799**	**0.890**

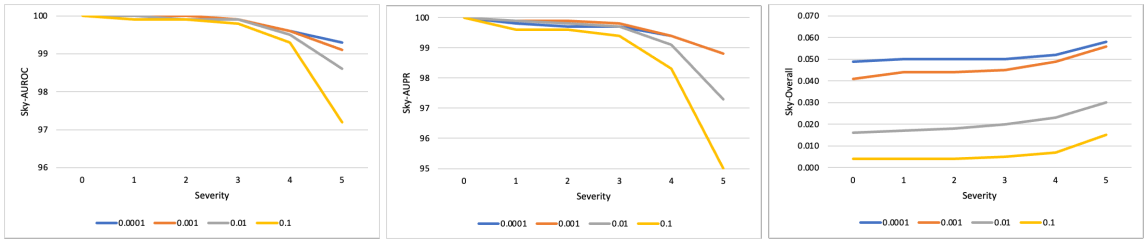
Table 5.14: Ablation study on the necessity of using AuxUE. Main task performance comparison on KITTI and KITTI-C.

		Aleatoric uncertainty estimation				Epsitemic uncertainty: Unseen pattern			Dataset change	
S	Methods	AUSE-REL ↓	AUSE-RMSE ↓	AURG-REL ↑	AURG-RMSE ↑	Sky-AUROC ↑	Sky-AUPR ↑	Sky-Overall ↓	AUROC ↑	AUPR ↑
0	Org + AuxUE	**0.013**	0.203	0.023	1.870	**100.0**	**100.0**	0.015	98.1	99.3
	BTS-SinglePU	0.016	0.222	0.026	1.978	89.0	62.0	**0.005**	64.2	78.3
	DEns.	0.014	**0.195**	0.024	1.866	93.5	70.0	**0.005**	62.1	76.7
	BTS-DIDO	**0.013**	0.207	**0.028**	**1.990**	**100.0**	**100.0**	0.017	**98.5**	**99.5**
1	Org + AuxUE	0.019	0.336	0.031	2.361	**100.0**	**99.9**	0.016		
	BTS-SinglePU	0.021	0.330	0.038	**2.657**	86.9	59.1	**0.005**		
	BTS-DEns.	0.019	**0.285**	0.036	2.573	92.8	68.0	**0.005**		
	BTS-DIDO	**0.017**	0.308	**0.041**	2.608	**100.0**	**99.9**	0.027		
2	Org + AuxUE	0.023	0.468	0.038	2.774	99.9	99.8	0.017		
	BTS-SinglePU	0.026	0.443	0.046	**3.150**	86.6	58.9	**0.005**		
	BTS-DEns.	0.022	**0.387**	0.044	3.078	92.3	67.0	**0.005**		
	BTS-DIDO	**0.021**	0.396	**0.050**	3.093	**100.0**	**99.9**	0.033		
3	Org + AuxUE	0.031	0.730	0.049	3.308	**99.9**	99.7	0.018		
	BTS-SinglePU	0.031	0.619	0.055	3.719	86.6	59.5	**0.005**		
	BTS-DEns.	0.027	0.526	0.054	3.685	91.6	65.7	**0.005**		
	BTS-DIDO	**0.023**	**0.500**	**0.062**	**3.749**	**99.9**	**99.8**	0.036		
4	Org + AuxUE	0.049	1.268	0.059	3.929	99.6	99.1	0.023		
	BTS-SinglePU	0.038	0.905	0.067	4.345	87.2	61.7	**0.005**		
	BTS-DEns.	0.032	0.734	0.067	4.401	91.8	67.2	**0.005**		
	BTS-DIDO	**0.029**	**0.680**	**0.074**	**4.446**	**99.8**	**99.7**	0.041		
5	Org + AuxUE	0.059	1.760	0.067	4.496	98.5	97.1	0.035		
	BTS-SinglePU	0.045	1.202	0.075	4.831	87.5	64.6	**0.005**		
	BTS-DEns.	**0.036**	**0.890**	0.080	**5.044**	92.2	70.4	**0.005**		
	BTS-DIDO	**0.036**	1.010	**0.084**	4.964	**99.5**	**99.3**	0.051		

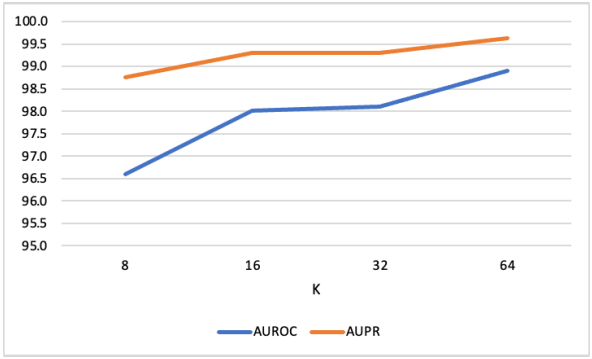
Table 5.15: Ablation study on the necessity of using AuxUE. Epistemic uncertainty estimation performance comparison on KITTI and KITTI-C. On clean KITTI, the extra columns stand for the dataset change experiment.



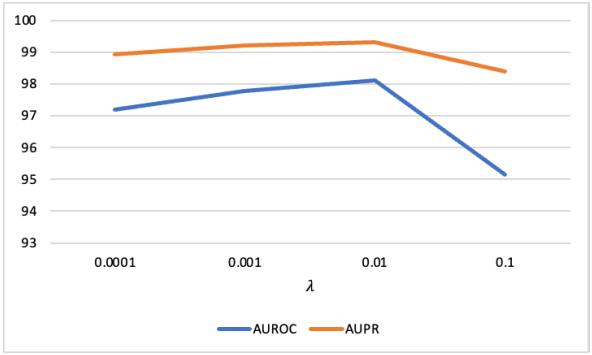
(a) Ablation study on K for DIDO on unseen patterns detection in KITTI dataset. The results are given by DIDO-based AuxUE with different numbers of classes (K) in discretization.



(b) Ablation study on λ for DIDO on unseen patterns detection in KITTI dataset. The results are given by DIDO-based AuxUE with (K = 32) trained by using different λ for the regularization term in loss $L(\Theta_2)$.



(c) Ablation study on K for DIDO on dataset change detection in monocular depth estimation. The evaluation is made by taking the KITTI outdoor dataset as the In-Distribution data and the NYU indoor dataset as the Out-of-Distribution data.



(d) Ablation study on λ for DIDO on dataset change detection in monocular depth estimation. The evaluation is made by taking the KITTI outdoor dataset as the In-Distribution data and the NYU indoor dataset as the Out-of-Distribution data.

Figure 5.5: Ablation study on hyperparameters for DIDO on monocular depth estimation.

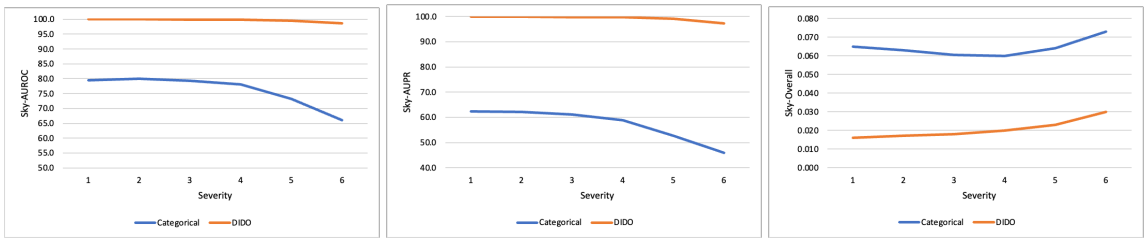


Figure 5.6: Ablation study on the effectiveness of Dirichlet modeling for DIDO on monocular depth estimation. K = 32 for both Categorical and Dirichlet modeling cases.

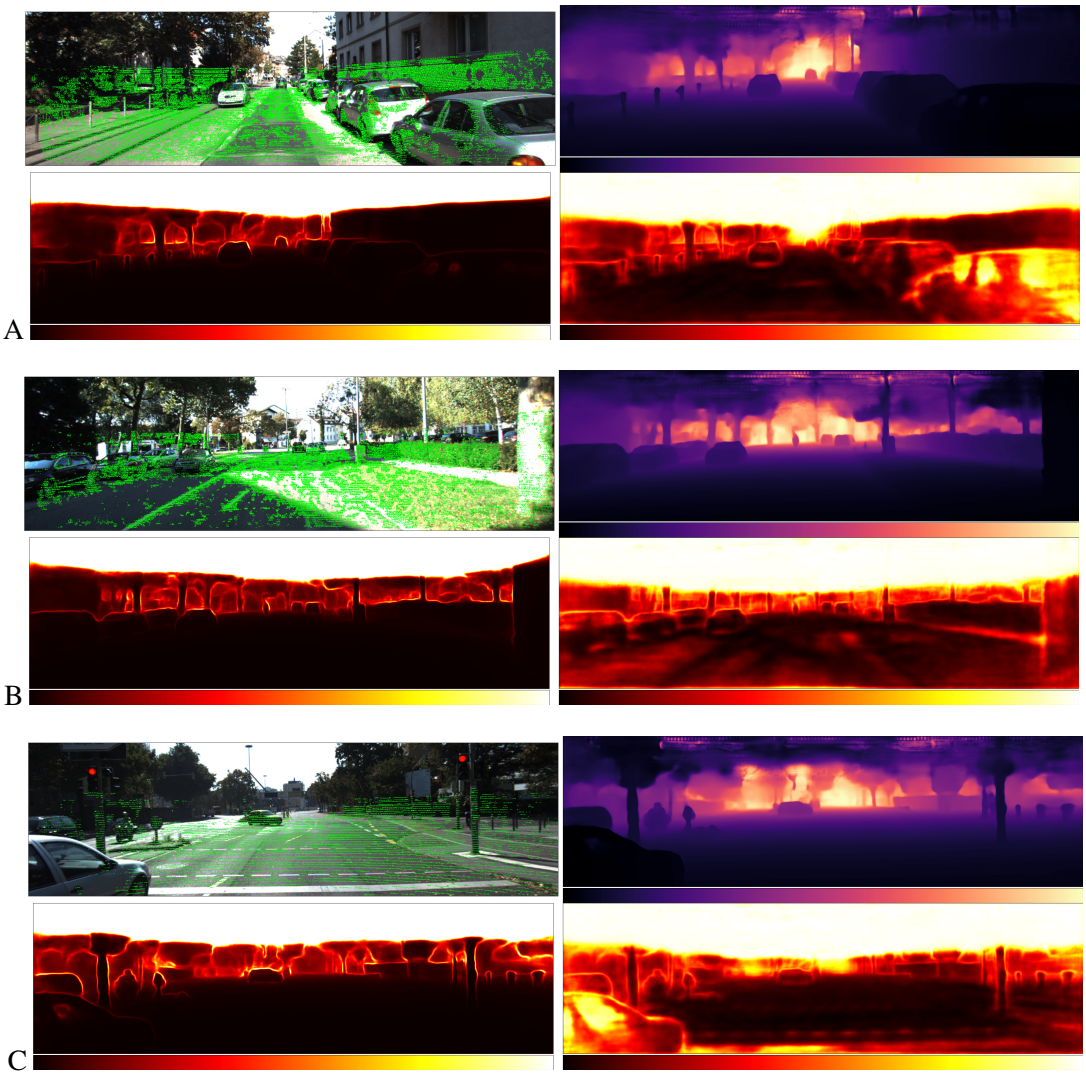
5.5 More visualizations

Figure 5.7 shows more visualizations on monocular depth estimation. For aleatoric uncertainty estimation maps, since some of the values on the unseen part (mostly the upper part of the map) are extremely high (>1e4), we **clip the values** to the maximum predicted value on the pixels with ground truth for better illustration. As uncertainty estimates show, our proposed DIDO can highlight the patterns rarely appearing throughout the whole dataset, e.g., the windshield of the car, the underside of the car, the barbed wire fence, and the upper part of the image, like the sky. However, only the sky part is a pattern that must have no groundtruth depth values and have semantic segmentation annotations. This is the reason we choose only the sky as the OOD pattern. Note that DIDO will not always highlight the areas without ground truth. For instance, we can see DIDO does not always assign higher uncertainty on the parts with no ground truth for the body of the car or the road since some of these patterns might have ground truth on the other images in the training set or share similar patterns which have ground truth values.

5.6 Conclusion

After introducing the previous chapters, in this chapter, we try to combine the ideas from the auxiliary network design SLURP and discretization-induced monocular depth estimation CAR MDE and improve the epistemic uncertainty estimation for regression tasks. Using a newly introduced generalized auxiliary network as a vehicle, we explore and design a method for estimating epistemic uncertainty for regression tasks based on discretization and Dirichlet posterior estimation. It is effective and scalable to both image and pixel-wise tasks and even can be applied to tabular data. We link regression tasks to the classification ones and want to unify the problem settings and provide a more scalable and effective uncertainty quantification solution.

The Dirichlet posterior estimation introduced in this chapter is a kind of approach that requires only a single forward propagate of the DNN to obtain uncertainty quantification. It is based on small modifications to the model training and was originally applied to classification tasks. In the next chapter, we will move to these single-pass uncertainty quantification approaches. Based on some simple modifications to the original main task model architecture and training, we hope to obtain more robust uncertainty quantification results and main task accuracy than using the original model alone, with a single forward propagate of the DNN. At the same time, we will try to solve the uncertainty quantification problems not only for regression tasks but also for classification ones.


A
B
C

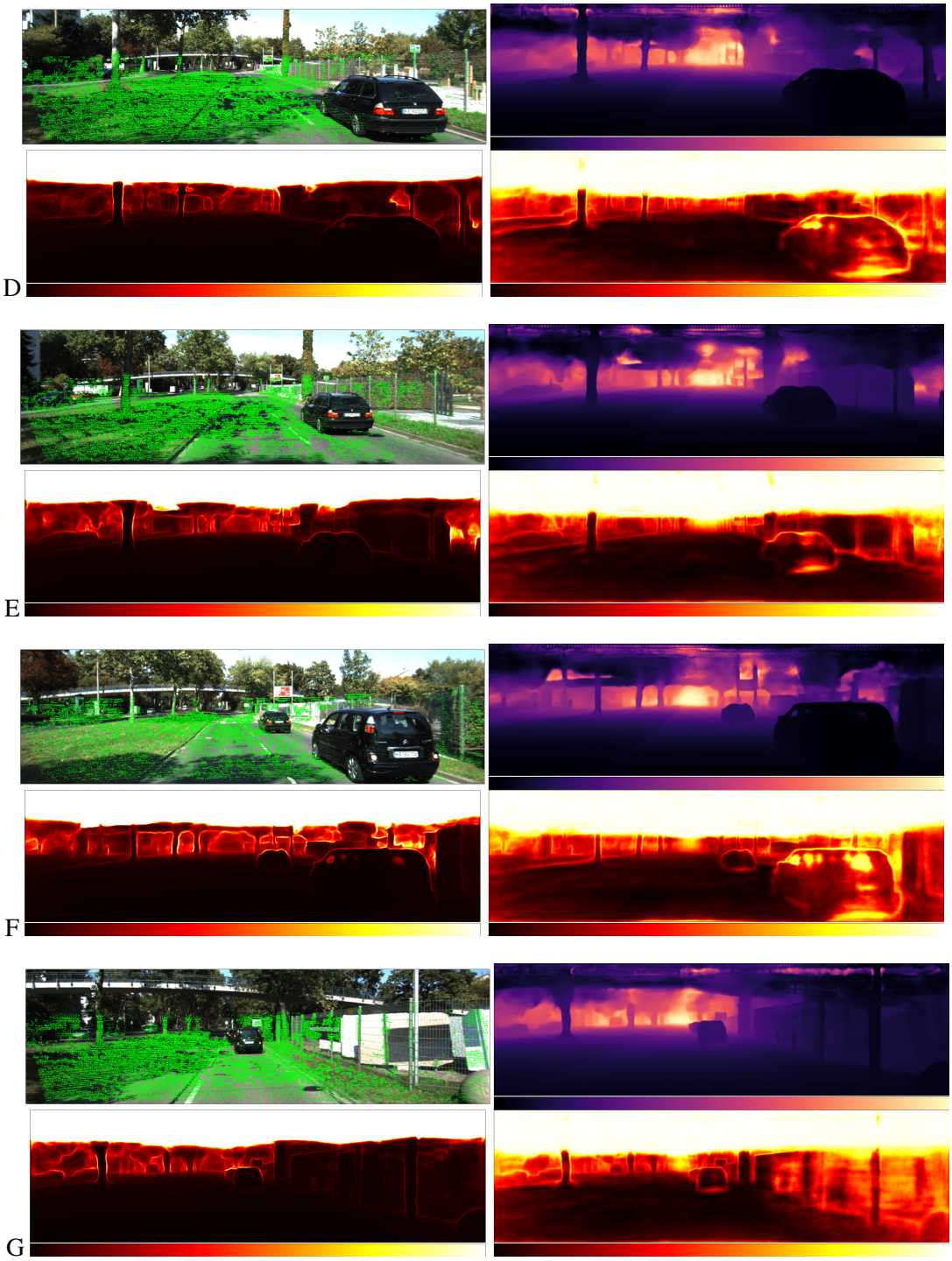


Figure 5.7: Visualizations on monocular depth estimations and corresponding uncertainty quantification results.

Part IV

Single-forward-propagation uncertainty quantification methods

Chapter 6

Latent discriminant discriminative uncertainty

Contents

6.1 Introduction

In uncertainty quantification solutions, the best-performing approaches are computationally expensive [123]. In the previous chapters, we introduced auxiliary networks to estimate uncertainty, which are lighter and able to achieve comparable and even better uncertainty quantification results without re-training or modifying the original model. Yet, it is always an alternative choice to modify the original DNN for better uncertainty quantification if the training code of the original model is open-source and the training consumption is acceptable. Since it only requires single forward propagation instead of double propagation using auxiliary DNNs.

In this chapter, we study a promising new line of methods, termed deterministic uncertainty methods (DUMs) [194], that has recently emerged for estimating uncertainty in a computationally efficient manner from a single forward pass [146, 171, 193, 230, 231]. 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 [190]. 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 the Lipschitz constraint directly.

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 [81] 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.

As introduced in the previous chapters, SLURP [252], as well as ConfidNet [41], have shown that we can train an auxiliary network to predict the uncertainty using a similar confidence-based loss. While they have 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

In this chapter, our contributions are as follows:

1. We introduce Latent Discriminant deterministic Uncertainty (LDU). LDU is an efficient and scalable DUM approach for uncertainty quantification. Moreover, we also provide a study of LDU's properties against feature collapse, which helps identify the OOD patterns or instances.

2. We evaluate LDU on depth estimation for both In-Distribution uncertainty and OOD detection to further encourage research in this area.

6.2 Related work

In this section, we focus on the related works from two perspectives: DUM-based uncertainty quantification algorithms applied to computer vision tasks and prototype learning on DNNs.

DUM-based uncertainty quantification algorithms

DUMs [2, 146, 171, 193, 230, 231, 239] are new strategies that allow quantifying epistemic uncertainty in the DNNs with a single forward pass. Different from the solutions which formalize the DNN output as a conjugate prior distribution of the likelihood distribution, such as the evidential DNNs [228], the DUMs here we mentioned mostly focus on learning useful and informative hidden representations of a model 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 [231] 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 slight 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 [146, 160, 231].

Yet, except for MIR [193], to the best of our knowledge, none of these techniques work on semantic segmentation. Some DUMs [193, 230] also work on regression tasks. DUE [230] is applied in a 1D regression task, and MIR [193] in monocular depth estimation. We argue that the scalable and effective DUM is still under-explored.

Prototype learning in DNNs

Prototype-based learning approaches have been introduced on traditional handcrafted features [145], and have been recently applied to DNNs as well, for more robust predictions [35, 81, 238, 246]. The center loss [238] 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) [246] with prototype loss also improves the intra-class compactness of the latent features. Chen et al. [35] try to bound the unknown classes by learning reciprocal points for better open set recognition. Similar to [199, 232], MemAE [81] learns a memory slot of the prototypes to strengthen the reconstruction error of anomalies in the reconstruction process. These prototype-based methods are well-suited for classification tasks but are rarely used in semantic segmentation and regression tasks. In our work, we will apply prototype learning on the penultimate layer to better regularize and learn the semantic information of the given features.

6.3 Latent Discriminant deterministic Uncertainty (LDU)

6.3.1 DUM preliminaries

Formally, we define $f_{\boldsymbol{\omega}}(\cdot)$ a DNN with trainable parameters $\boldsymbol{\omega}$, and an input sample $\mathbf{x}$ from a set of images $\mathscr{X}$. Our DNN $f_{\boldsymbol{\omega}}$ is composed of two main blocks: a feature extractor $h_{\boldsymbol{\omega}}$ and a head $g_{\boldsymbol{\omega}}$, such that $f_{\boldsymbol{\omega}}(\mathbf{x}) = (g_{\boldsymbol{\omega}} \circ h_{\boldsymbol{\omega}})(\mathbf{x})$. $h_{\boldsymbol{\omega}}(\mathbf{x})$ computes a latent representation from $\mathbf{x}$, while $g_{\boldsymbol{\omega}}$ is the final layer, that takes $h_{\boldsymbol{\omega}}(\mathbf{x})$ 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 $\mathscr{X}$:

$$\mathrm{L}_1 \|\mathbf{x}_1 - \mathbf{x}_2\| \leq \|h_{\boldsymbol{\omega}}(\mathbf{x}_1) - h_{\boldsymbol{\omega}}(\mathbf{x}_2)\| \leq \mathrm{L}_2 \|\mathbf{x}_1 - \mathbf{x}_2\| \tag{6.1}$$

where L_1 and L_2 are positive and bounded Lipschitz constants $0 < \mathrm{L}_1 < 1 < \mathrm{L}_2$. 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. [146] argue that for residual DNNs [87], we can ensure $f_{\boldsymbol{\omega}}$ to be bi-Lipschitz by forcing its residuals to be Lipschitz and choosing sub-unitary Lipschitz constants. A simple illustration of the difference between the DUMs and the baseline is shown in Figure 6.1.

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 [6, 10, 82, 170]. Wasserstein GAN [6] 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 [82]. An alternative solution from GAN optimization is gradient penalty [82] 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 [146, 171] and slower training [171]. Spectral Normalization [10, 170] 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. [193] 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 [193] 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.

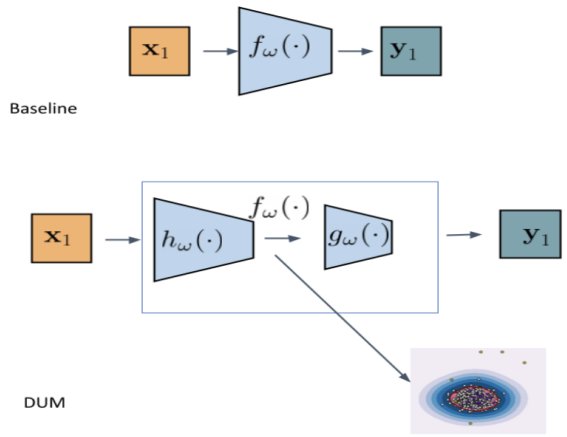


Figure 6.1: An overview of the DUMs.

6.3.2 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, 8]. 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 [153] that has been recently considered as a replacement for the last layer to produce better uncertainty estimates, in particular for OOD detection [153, 184]. 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 [78, 195, 216]. In contrast to all these approaches that use DM as a last layer for classification predictions, we employ it as a 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 the last layers where the weights are more specialized for classification decisions than for feature representation. We can easily integrate this layer into 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., $\mathbf{z} = h_{\boldsymbol{\omega}}(\mathbf{x})$, that is given as input to the DM layer. Given a set $\mathbf{p}_{\boldsymbol{\omega}}=\{\mathbf{p}_i\}_{i=1}^m$, of m vectors ($\mathbf{p}_i \in \mathbb{R}^n$) that are trainable, we define the DM layer as follows:

$$\mathrm{DM}_p(\mathbf{z}) = \Big[-\|\mathbf{z}-\mathbf{p}_1\|, \ldots, -\|\mathbf{z}-\mathbf{p}_m\|\Big]^\top \tag{6.2}$$

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

$$\mathrm{DM}_p(\mathbf{z}) = \Big[\mathrm{S}_c(\mathbf{z},\mathbf{p}_1), \ldots, \mathrm{S}_c(\mathbf{z},\mathbf{p}_m)\Big]^\top \tag{6.3}$$

The vectors $\mathbf{p}_i$ can be seen as a set of prototypes in the latent space that can help in better placing

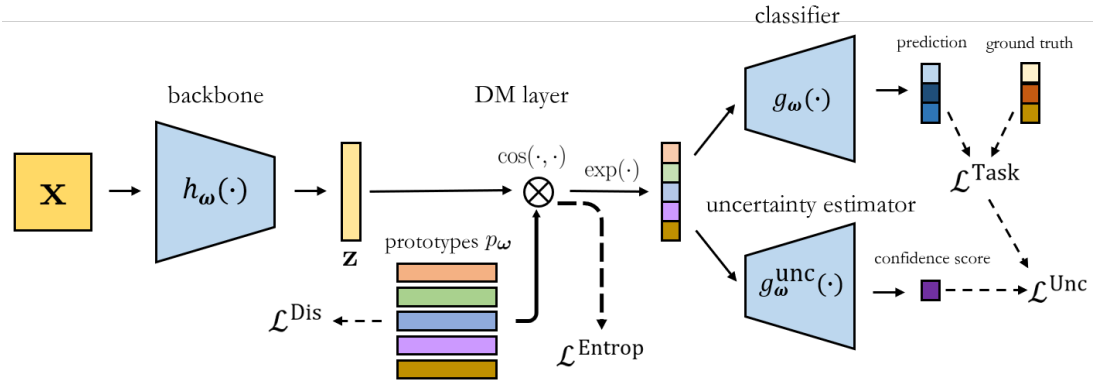


Figure 6.2: Overview of LDU. The DNN learns a discriminative latent space thanks to learnable prototypes $\mathbf{p}_{\omega}$. The DNN backbone computes a feature vector $\mathbf{z}$ for an input $\mathbf{x}$ and then the DM layer matches it with the prototypes. The computed similarities reflecting 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.

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 the last layer, where the prototypes represent canonical representations for samples belonging to a class [153, 216]. 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 an 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 6.2) can be written as:

$$f_{\omega}(\mathbf{x}) = \left[g_{\omega} \circ (\exp(-\mathrm{DM}_p(h_{\omega})))\right](\mathbf{x}) \tag{6.4}$$

We can see from Eq. (6.4) 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.

6.3.3 LDU optimization

Given a DNN f_{ω}, we usually optimize its parameters to minimize a loss $\mathscr{L}_{\text{Task}}$. 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 $\mathrm{DM}_p(h_{\omega})$ of the DNN to not rely only on a single prototype. Finally, we optimize an MLP g^{unc}_{ω} on the top of the latent representation $\mathrm{DM}_p(h_{\omega})$ 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:

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

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:

$$\mathscr{L}_{\text{Entrop}} = \sum_{i=1}^{n} \mathrm{Softmax}(\mathrm{DM}_p(h_{\omega}))_i \cdot \log(\mathrm{Softmax}(\mathrm{DM}_p(h_{\omega}))_i)$$

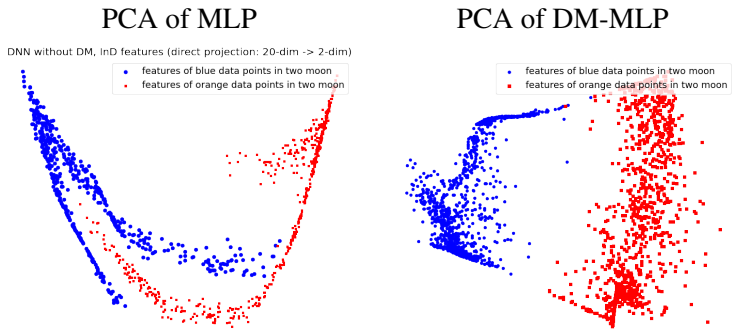


Figure 6.3: PCA 2D projection on the left of a standard MLP and on the right of a DM-MLP trained on the two moons dataset. Blue and red points indicate the features of data points of the two classes, respectively.

where the subscript index i corresponds to the i-th coefficient of a tensor. Different from per-class prototypes [35, 238, 246], 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 $g^{\text{unc}}_{\boldsymbol{\omega}}$ 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 $\mathscr{L}_{\text{Task}}(g_{\boldsymbol{\omega}}(\mathbf{x}), y)$, we train g^{unc}_{ω} by minimizing:

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

after normalizing $\mathscr{L}_{\text{Task}}(g_{\boldsymbol{\omega}}(\mathbf{x}), y)$ over the mini-batch such that its maximum value is equal to one and its minimum is equal to zero. BCE stands for 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 that can predict uncertainty, avoid feature collapse, and have the potential to improve the accuracy of the prediction. To summarize, the following loss function $\mathscr{L}_{\text{Total}}$ will be optimized to train a DNN containing a DM layer:

$$\mathscr{L}_{\text{Total}} = \mathscr{L}_{\text{Task}} + \lambda(\mathscr{L}_{\text{Entrop}} + \mathscr{L}_{\text{Dis}} + \mathscr{L}_{\text{Unc}}) \tag{6.5}$$

where λ is a hyper-parameter for the auxiliary losses.

6.3.4 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 6.3. 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, $h_{\boldsymbol{\omega}}$ 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 [154, 160] can allow it to learn to separate the two data distributions better, as illustrated in Figure 6.3.

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

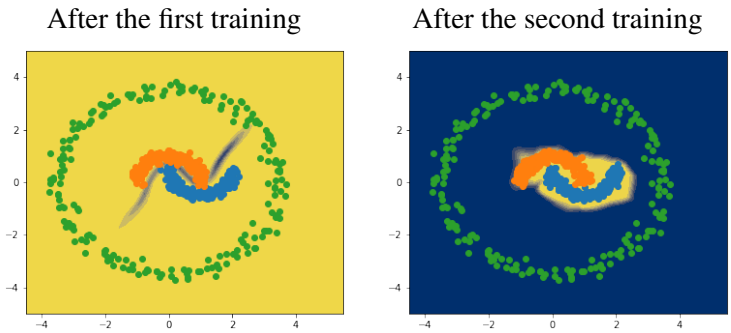


Figure 6.4: Illustration of confidence score results on the two moons dataset after the first training (on original data) on the left and with second training (on synthesized outliers) on the right. Orange and blue data points are sampled from two classes in two moons, and the green points are OOD data points. The yellow area indicates high confidence, and the blue area indicates uncertainty.

i.i.d. from a normal distribution $\mathcal{N}(0, \sigma^2 \mathrm{I}_d)$. I_d is the identity matrix of size d. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a Lipschitz function with Lipschitz constant K. The concentration theory ([21], p. 125) stipulates that : $\mathscr{P}(|f(\mathbf{X}) - \mathbb{E}(f(\mathbf{X}))| > t) \leq 2\exp(-\frac{t^2}{2\mathrm{K}^2\sigma^2})$ 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.

6.3.5 LDU and Epistemic/Aleatoric Uncertainty

We are interested in capturing two types of uncertainty with our DNN: aleatoric and epistemic uncertainty [50, 113]. 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 [171], and ensemble methods are usually superior [61, 159].

Optional training with synthesized outliers Due to limited training data and to the penalty enforced by $\mathscr{L}_{\text{Task}}$ being too small, the loss term $\mathscr{L}_{\text{Unc}}$ 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 [55, 157], and to introduce an optional step for training $g^{\text{unc}}_{\boldsymbol{\omega}}$ 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 $g^{\text{unc}}_{\boldsymbol{\omega}}$ over the synthesized outliers. Note that this optional stage would require for vision tasks an adequate OOD synthesizer [13, 55] which is beyond the scope of this paper, and that we applied it on the toy dataset. In Figure 6.4, we assess the uncertainty estimation performance of this model on the two moons dataset. The left image shows that after the first training stage, the uncertain area is between the two classes leading to a confidence score related to aleatoric uncertainty. The right one shows that, after the second training stage, the uncertain area is around the dataset leading to a confidence score related to epistemic uncertainty.

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

Aleatoric/Epistemic uncertainty For classification and semantic segmentation tasks, we estimate aleatoric uncertainty using maximum class probability (MCP) [91], and we can use the outputs of $g_{\boldsymbol{\omega}}^{\text{unc}}$ as the epistemic uncertainty estimates. However, for regression tasks, we use only the $g_{\boldsymbol{\omega}}^{\text{unc}}$ outputs as confidence scores. We argue that the outputs of $g_{\boldsymbol{\omega}}^{\text{unc}}$ represent aleatoric uncertainty estimates. However, thanks to the DM layer, 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. Therefore, although the output of $g_{\boldsymbol{\omega}}^{\text{unc}}$ can be interpreted as aleatoric uncertainty using the learning loss strategy [252] under the Bayesian framework and the Gaussian data-dependent noise assumption, we consider that the enhanced features can enable the aleatoric uncertainty estimate to generalize to rare inputs, and gives such inputs higher uncertainty in the output.

6.4 Experiments

One major interest of our technique is that it may be seamlessly applied to any computer vision task, be it classification or regression. We here propose to evaluate the quality of uncertainty quantification of different techniques on monocular depth estimation, the evaluation on the other tasks can be found in the LDU paper [66].

We use the Area Under the Sparsification Error: AUSE-RMSE and AUSE-Absrel similarly to [85, 191, 252] to evaluate the uncertainty quantification on monocular depth estimation. We also report the OOD detection results in Section 5.4.5.2 and Section 5.4.5.3 under the monocular depth estimation task setting.

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.

6.4.1 Monocular depth experiments

We set up our experiments on KITTI dataset [227] with Eigen split training and validation set [56] to evaluate and compare the predicted depth accuracy and uncertainty quality. We train BTS [130] with DenseNet161 [100], 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 [104, 113], 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). In detail, 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, and one for uncertainty estimation. For the Deep Ensembles [123], we train Single-PU models. We also trained an MC-Dropout [70], and we did eight forward passes during inference to achieve uncertainty estimates. Without the extra DNNs or training procedures, we also applied Infer-noise [167], which injects Gaussian noise layers to the trained BTS baseline model and propagates eight times to predict the uncertainty. For implementing LDU on BTS, we use 30 prototypes in the DM layer, and we provide all hyper-parameters in Table 6.1.

Results We note that in the monocular depth estimation setting and in agreement with previous works [52], 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 prediction accuracy, and still perform well for aleatoric uncertainty estimation. Table 6.2 lists the depth and uncertainty estimation results on the KITTI dataset. Using different settings of #**p** and λ, the proposed LDU is virtually aligned with the current state-of-the-art, while being significantly lighter computationally (see also Table. 6.3).

Hyper-parameter	**KITTI**
Architecture	BTS [130]
Backbone	DenseNet161 [100]
Initial learning rate	0.0001
Batch size	4
Number of train epochs	50
Weight decay	0.01
Random crop of training images	(352, 704)
Number of prototypes	30

Table 6.1: Hyper-parameter configuration used in the monocular depth estimation experiments.

Method	**Depth performance**								**Uncertainty performance**	
	δ1↑	δ2↑	δ3↑	AbsRel↓	SqRel↓	RMSE↓	RMSElog↓	log10↓	AUSE RMSE↓	AUSE Absrel↓
Baseline	0.955	**0.993**	0.998	**0.060**	0.249	2.798	0.096	0.027	-	-
Deep Ensembles [123]	**0.956**	**0.993**	**0.999**	**0.060**	**0.236**	**2.700**	**0.094**	**0.026**	**0.08**	**0.21**
MC-Dropout [70]	0.945	0.992	0.998	0.072	0.287	2.902	0.107	0.031	0.46	0.50
Single-PU [113]	0.949	0.991	0.998	0.064	0.263	2.796	0.101	0.029	**0.08**	**0.21**
Infer-noise [167]	0.955	**0.993**	0.998	**0.060**	0.249	2.798	0.096	0.027	0.33	0.48
LDU #$\mathbf{p}$ = 5, λ = 1.0	0.954	**0.993**	0.998	0.063	0.253	2.768	0.098	0.027	**0.08**	**0.21**
LDU #$\mathbf{p}$ = 15, λ = 0.1	0.954	**0.993**	0.998	0.062	0.249	2.769	0.098	0.027	0.10	0.28
LDU #$\mathbf{p}$ = 30, λ = 0.1	0.955	0.992	0.998	0.061	0.248	2.757	0.097	0.027	0.09	0.26

Table 6.2: Comparative results for **monocular depth estimation on KITTI eigen-split validation set**.

Visualization In Figure 6.5, 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 an OOD region after training on the entire dataset. In Chapter III.5, we provide the quantitative results for OOD detection from two aspects: OOD caused by the dataset change and unseen pattern during training.

Runtime consumption In Table 6.3, 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 an 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 6.4.1. 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. Furthermore, thanks to researchers in the field, a speedup of the cosine similarity operation is available [203], which could theoretically make the current inference speed even faster.

Method	Monocular depth		
	Runtime (ms)	Training time (ms)	#param.
Baseline	**45**	92.8	**47.00**
Deep Ensembles	133	287.8	141.03
MC-Dropout	370	**92.3**	**47.00**
Single-PU	**45**	95.6	47.01
LDU (ours)	**45**	104.0	**47.00**

Table 6.3: 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.

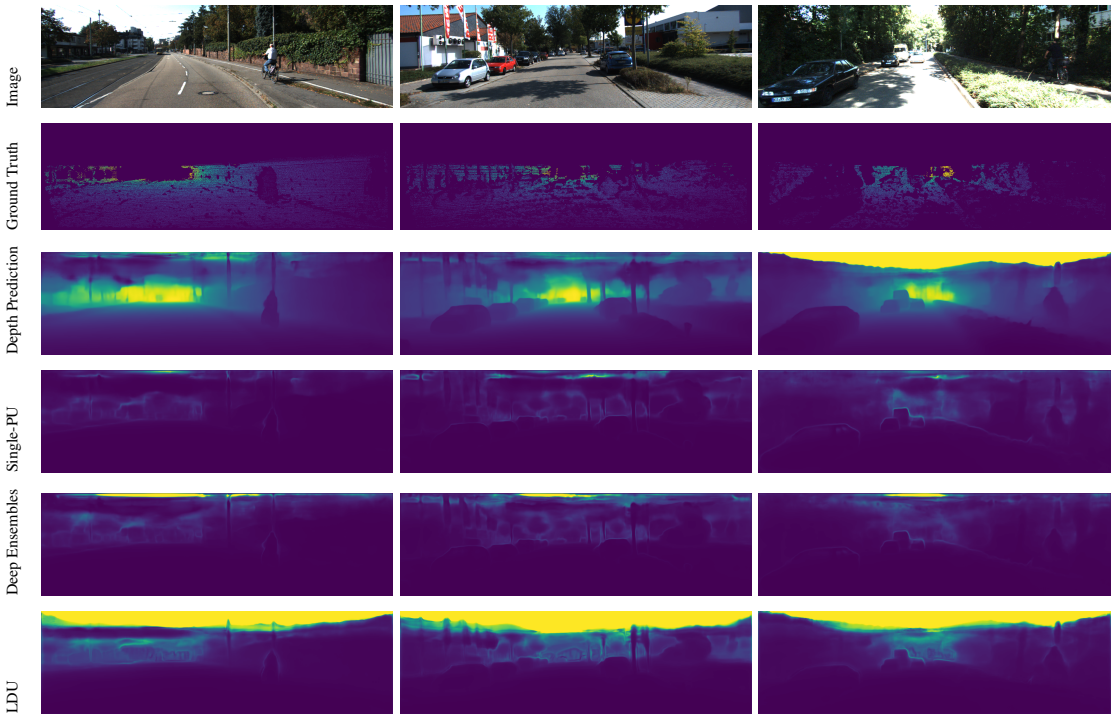


Figure 6.5: Illustration of different uncertainty maps (LDU, Deep Ensembles, Single-PU) on KITTI images for the monocular depth estimation task. For both depth and uncertainty maps, the brighter the color is, the bigger the value the pixel has.

6.5 Conclusions

This chapter introduced a novel deterministic uncertainty quantification method named Latent Discriminant deterministic Uncertainty (LDU). We show that instead of regularizing the whole network, adding a bi-Lipschitz constraint on the pre-logit layer can effectively avoid feature collapse and improve the OOD detection performance. LDU is a relatively more portable and scalable solution than the other DUMs.

Together with the previous chapters, we have already introduced two main kinds of uncertainty quantification solutions: post-hoc uncertainty quantification using auxiliary networks and the deterministic uncertainty method. In this way, in both cases of retaining and adjusting the original model architecture, we can flexibly choose methods to obtain uncertainty quantification of the main task and provide assistance for downstream tasks such as learning with imbalanced classes and noisy labels, active learning, etc.

Conclusion and future work

Conclusion

In this work, we propose a family of methods for uncertainty estimation of the outputs of deep neural networks. In particular, we focus on the solutions of using auxiliary neural networks to estimate the uncertainty of regression problems when the main task model is trained. The auxiliary model is a post-hoc solution capable of obtaining uncertainty estimates of the main task model's predictions without the need to modify and retrain the main task model. Additionally, we also propose solutions based on modifying the main task model to achieve more calibrated predictions on the in-distributional dataset and better-performed uncertainty estimates on out-of-distributional datasets.

We work on computer vision tasks, propose scalable uncertainty quantification methods, and apply them to both pixel-wise and image-level tasks. Based on the proposed algorithm, the uncertainty estimates can provide interpretability for the main task estimation given by the deep neural networks, and serve as an important reference for subsequent system or human decision-making.

In Part II, we introduce a fundamental uncertainty quantification solution based on the auxiliary network, dubbed SLURP, Side Learning Uncertainty for Regression Problems. Given a trained main task deep neural network, we consider that the prediction error of the main task model contains both aleatoric and epistemic uncertainty, and the majority is the aleatoric uncertainty since the main task model is already trained on the in-distribution dataset. The goal of the auxiliary network is to learn this prediction error in a more effective way, in order to provide reliable aleatoric and epistemic uncertainty and generalize well on the shifted datasets. For this goal, we first propose a novel auxiliary network framework. The main contribution is that we observe that using both the input and the output of the main task model can improve the uncertainty quantification performance. Furthermore, according to the observations of the prediction errors, our auxiliary network is designed based on the edge detection model, with a sequence of atrous convolutions and a coarse-to-fine fusion block on the top. Our experiments on optical flow and monocular depth estimation tasks show that the proposed auxiliary network is able to achieve comparable uncertainty quantification results to the deep ensembles.

In Part III, our focus switches to how to assign higher uncertainty to the unseen examples using auxiliary networks. Distributional uncertainty modeling is one of the most effective ways to achieve this goal, yet it is not trivial to apply it directly on the auxiliary networks. We observe that previous solutions with distributional uncertainty modeling for regression tasks are not suitable for the auxiliary network since the trained main task model provides unbalanced and small prediction errors on the in-distributional dataset, which will make it hard for the auxiliary network to learn the useful information from the prediction error. We propose to involve discretization and distributional uncertainty modeling in the auxiliary network framework.

A survey is first settled to summarize the use of discretization methods in regression problems, especially the monocular depth estimation task, in the previous literature. In the meantime, we propose a new uncertainty quantification method based on discretization-based regression tasks, named Expectation of Distance (E-Dist). E-Dist is a post-hoc uncertainty quantification method, without

adjusting the training procedures or the architecture of discretization-based models. The experiments on different discretization-based solutions show that E-Dist outperforms the traditional uncertainty quantification solutions, such as entropy and the maximum Softmax probability.

Based on the study on the discretization approaches, we pioneered the application of discretization in distributional uncertainty modeling for regression tasks, in order to reduce the effect of the small and unbalanced main task prediction error, which is served for the auxiliary network training. To estimate the distributional uncertainty, we use the Dirichlet distribution over the discretized prediction error to model the distributional uncertainty. The network will update the Dirichlet parameters and achieve the Dirichlet posterior after training. This technique is also called evidential learning in the previous literature. For aleatoric uncertainty, we keep the idea of learning loss provided by the fundamental SLURP. When the main task model parameters are fixed, it means that the model is not allowed to further learn or adapt to new data. The model's knowledge is static. Distributional uncertainty, which is associated with the inherent variability in the input data, may also be considered as a form of epistemic uncertainty because the model's fixed parameters cannot adapt to or account for this variability. It is a form of uncertainty that arises due to the model's inability to fully understand or model the complexity of the data distribution. In this case, a generalized auxiliary network framework is proposed, which contains two identical sub-auxiliary networks: one is for aleatoric uncertainty estimation, and another is for epistemic uncertainty estimation. In this case, not only the uncertainty in the in-distribution dataset can be measured, but the out-of-distribution examples can also be detected. Our experiments show this property on both image-level age estimation and pixel-wise monocular depth estimation.

Apart from the uncertainty quantification solutions based on the post-hoc methods and auxiliary network, we also work on modifying the main task model to achieve calibrated predictions and scalable out-of-distribution example detection. In Part IV, a deterministic uncertainty estimation solution is proposed, dubbed Latent Discriminant deterministic Uncertainty (LDU). The deterministic solution is to estimate uncertainty by single-pass. We suggest learning the prototypes for the features from the penultimate (pre-logit) layer. The prototypes are the features that are independent of the network inputs. This procedure is to maximize the cosine similarity between the features and the prototypes and use the cosine similarity values as the inputs of the last layer. These values are bi-lipschitz, which is proven to be more robust to the perturbations on the input feature space. The uncertainty estimation solution adjusts the learning loss strategy from SLURP to the main task model.

For the fair evaluation of the robustness of the main task prediction and the corresponding uncertainty quantification in the urban scenes, in Part I, we introduce the Multiple Uncertainty for Autonomous Driving dataset (MUAD). This synthetic dataset consists of images of urban street scenes under different weather conditions. The training and validation sets consist of daytime and nighttime street view images with clear weather. In the test sets, we added different weather conditions, including clear, rainy, foggy, and snowy weather during the day and night. Under different weather conditions, the objects in the test set with semantic information that was not present in the training and validation sets are also included. We provide benchmarks for the main task and the uncertainty quantification on this dataset based on the state-of-the-art backbones. We found that both supervised and unsupervised monocular depth estimation tasks are relatively robust to out-of-distribution objects. We also found that the angle of illumination, i.e., ordinary daytime, noon, or night, has little impact on the performance of the state-of-the-art monocular depth estimation models.

In summary, our main work involves scalable uncertainty quantification solutions and uncertainty estimation evaluation for regression tasks. Our focus is mainly on post-hoc methods to avoid modifying and retraining main task models. We propose the generalized auxiliary network framework, and in this framework, we model both aleatoric and epistemic uncertainty for the main task prediction. Table 6.4 provides a take-away on the proposed uncertainty quantification methods that can be deployed under different circumstances. Additionally, we also present a synthetic dataset that we hope will allow for a fair evaluation of the robustness of uncertainty estimates.

Proposed uncertainty quantification solutions	Freezing main task model	Additional training	# forward propagation
Simple auxiliary uncertainty estimator (**SLURP**) (Section 3, Part II)	✗	✓	2
Expectation of distance (**E-dist**) (Section 4, Part III)	✓	✗	1
Discretization-induced Dirichlet posterior (**DIDO**) (Section 5, Part III)	✗	✓	2
Latent Discriminant deterministic Uncertainty (**LDU**) (Section 6, Part IV)	✓	✗	1

Table 6.4: Features of proposed uncertainty quantification methods.

Perspectives for future works

Future work will focus on four directions: uncertainty quantification, applications of uncertainty estimates, model robustness and interpretability, and tasks other than computer vision. For uncertainty quantification, we could work on the post-hoc solutions, to reduce the cost of having high-quality uncertainty estimates. The focus will be on how to use less data to train a scalable auxiliary network and obtain better uncertainty estimation results. After three years of work on uncertainty quantification, it is meaningful to think about how uncertainty estimation can be explored in the most effective manner to help with the main task. Active learning is a well-known strategy that involves uncertainty estimation in its acquisition functions and annotation systems. Meanwhile, the visit of a machine learning, meta-learning, and optimization group at Leiden University also made me think about introducing the uncertainty in meta-learning. Uncertainty quantification, model robustness, and model interpretability are inseparable. Models are ultimately made available for human use, and the uncertainty estimates can be seen as part of interpretability. At the same time, when the main task is not robust, the robustness of its uncertainty quantification is crucial. Researching and improving the robustness of uncertainty estimation is a key guarantee for the safe deployment of the model. Finally, a good uncertainty quantification method should be more scalable, i.e., it can be applied not only to different tasks in computer vision but also to tasks outside of the computer vision field. For example, natural language processing, brain electrical signals, protein graph structure, time series information in stocks, etc. It is exciting to study uncertainty quantification in other fields. Applying the experience obtained in computer vision to other fields, such as using post-hoc methods to estimate uncertainty, etc., may bring interpretability to more fields and allow more trustworthy models to be deployed.

Appendix

A1 More experiments and benchmarks on MUAD dataset

A1.1 Full results on supervised monocular depth estimation

In Section 2.3, for the sake of brevity, we provided only partial results for depth and uncertainty metrics. We here provide full results from Table A5 to Table A11 for different uncertainty quantification solutions introduced in the main paper applied on supervised monocular depth estimation task. Overall, the Deep Ensembles [123] and SLURP [252] can provide better uncertainty estimations on the test sets without perturbations. When weather perturbations exist, MC-Dropout [70] and Deep Ensembles [123] perform better on uncertainty quantification. MC-Dropout can also provide better depth estimations than the other solutions under weather perturbations.

Methods	silog↓	AbsRel↓	log10↓	RMSE↓	SqRel↓	log_RMSE	d1↑	d2↑	d3↑	AUSE↓			AURG↑		
										AbsRel	RMSE	d1	AbsRel	RMSE	d1
Baseline	13.9767	0.1143	0.0444	3.3575	0.5571	0.1443	0.9219	0.9833	0.9933	-	-	-	-	-	-
Deep Ensembles [123]	13.6691	0.1110	0.0419	3.1994	0.6076	0.1400	0.9289	0.9843	0.9945	0.0604	0.2906	0.0431	0.0117	2.4618	0.0215
MC Dropout [70]	13.5602	0.1194	0.0447	3.2090	0.6897	0.1453	0.9193	0.9847	0.9941	0.0610	0.6339	0.0542	0.0161	2.0846	0.0193
Single-PU [113]	14.5896	0.1324	0.0484	3.2298	0.7738	0.1547	0.9054	0.9803	0.9933	0.0807	0.3131	0.0837	0.0042	2.4194	-0.0005
SLURP [252]	13.9767	0.1143	0.0444	3.3575	0.5571	0.1443	0.9219	0.9833	0.9933	0.0477	0.4672	0.0459	0.0252	2.3870	0.0237

Table A5: Supervised monocular depth results on **normal set**.

Methods	silog↓	AbsRel↓	log10↓	RMSE↓	SqRel↓	log_RMSE	d1↑	d2↑	d3↑	AUSE↓			AURG↑		
										AbsRel	RMSE	d1	AbsRel	RMSE	d1
Baseline	19.8427	0.1474	0.0757	5.0053	0.8301	0.2397	0.7861	0.9244	0.9613	-	-	-	-	-	-
Deep Ensembles [123]	22.7950	0.1564	0.0850	4.8919	0.8508	0.2759	0.7673	0.9010	0.9419	0.1047	0.7401	0.1823	-0.0103	3.1624	0.0023
MC Dropout [70]	21.6959	0.1505	0.0765	4.5799	0.7648	0.2459	0.7980	0.9199	0.9543	0.0980	1.0627	0.1473	-0.0074	2.5851	0.0182
Single-PU [113]	24.2069	0.1588	0.0849	4.8648	0.8522	0.2800	0.7727	0.8997	0.9417	0.1115	0.7892	0.1863	-0.0145	3.1099	-0.0025
SLURP [252]	19.8429	0.1474	0.0757	5.0053	0.8301	0.2397	0.7861	0.9244	0.9613	0.0898	1.1665	0.1789	-0.0040	2.8036	-0.0037

Table A6: Supervised monocular depth results on **low adv. without OOD set**.

Methods	silog↓	AbsRel↓	log10↓	RMSE↓	SqRel↓	log_RMSE	d1↑	d2↑	d3↑	AUSE↓			AURG↑		
										AbsRel	RMSE	d1	AbsRel	RMSE	d1
Baseline	27.2917	0.2072	0.1148	6.9890	1.5990	0.3603	0.6316	0.8275	0.9028	-	-	-	-	-	-
Deep Ensembles [123]	34.7624	0.2429	0.1478	7.4977	1.9794	0.4674	0.5657	0.7643	0.8507	0.1529	1.1824	0.3031	-0.0117	4.6140	-0.0044
MC Dropout [70]	30.5442	0.2073	0.1142	6.2782	1.3762	0.3652	0.6567	0.8292	0.8992	0.1277	1.3819	0.2169	-0.0055	3.5187	0.0393
Single-PU [113]	41.9847	0.2480	0.1588	7.6797	2.1362	0.5295	0.5708	0.7586	0.8435	0.1706	1.7402	0.3318	-0.0220	4.2634	-0.0322
SLURP [252]	27.2917	0.2072	0.1148	6.9890	1.5990	0.3603	0.6316	0.8275	0.9028	0.1281	1.7066	0.2740	-0.0100	3.7188	-0.0024

Table A7: Supervised monocular depth results on **high adv. without OOD set**.

Methods	silog↓	AbsRel↓	log10↓	RMSE↓	SqRel↓	log_RMSE	d1↑	d2↑	d3↑	AUSE↓ AbsRel	AUSE↓ RMSE	AUSE↓ d1	AURG↑ AbsRel	AURG↑ RMSE	AURG↑ d1
Baseline	12.4227	0.0895	0.0387	3.6461	0.4083	0.1257	0.9513	0.9909	0.9969	-	-	-	-	-	-
Deep Ensembles [123]	11.7212	0.0829	0.0351	3.4788	0.3867	0.1188	0.9553	0.9903	0.9967	0.0553	0.3363	0.0098	-0.0041	2.6248	0.0336
MC Dropout [70]	12.0129	0.0915	0.0389	3.4074	0.3888	0.1263	0.9475	0.9902	0.9969	0.0576	0.7856	0.0308	-0.0019	2.0452	0.0199
Single-PU [113]	12.4754	0.1052	0.0437	3.5463	0.4210	0.1344	0.9461	0.9895	0.9966	0.0788	0.3576	0.0308	-0.0189	2.5430	0.0212
SLURP [252]	12.4227	0.0895	0.0387	3.6461	0.4083	0.1257	0.9513	0.9909	0.9969	0.0328	0.5248	0.0100	0.0222	2.5207	0.0373

Table A8: Supervised monocular depth results on **normal test set with Overhead Sun**.

Methods	silog↓	AbsRel↓	log10↓	RMSE↓	SqRel↓	log_RMSE	d1↑	d2↑	d3↑	AUSE↓ AbsRel	AUSE↓ RMSE	AUSE↓ d1	AURG↑ AbsRel	AURG↑ RMSE	AURG↑ d1
Baseline	16.4332	0.1250	0.0525	3.6157	0.5875	0.1747	0.8956	0.9602	0.9783	-	-	-	-	-	-
Deep Ensembles [123]	16.3795	0.1142	0.0503	3.4465	0.4812	0.1724	0.9027	0.9600	0.9777	0.0739	0.4268	0.0563	-0.0016	2.4750	0.0296
MC Dropout [70]	16.1976	0.1277	0.0525	3.4437	0.5923	0.1744	0.8934	0.9620	0.9799	0.0720	0.7253	0.0649	0.0104	2.1331	0.0292
Single-PU [113]	17.1019	0.1319	0.0561	3.4628	0.5126	0.1833	0.8884	0.9580	0.9777	0.0948	0.4474	0.0872	-0.0135	2.4091	0.0103
SLURP [252]	16.4332	0.1250	0.0525	3.6157	0.5875	0.1747	0.8956	0.9602	0.9783	0.0681	0.7208	0.0852	0.0121	2.2899	0.0054

Table A9: Supervised monocular depth results on **OOD set**.

Methods	silog↓	AbsRel↓	log10↓	RMSE↓	SqRel↓	log_RMSE	d1↑	d2↑	d3↑	AUSE↓ AbsRel	AUSE↓ RMSE	AUSE↓ d1	AURG↑ AbsRel	AURG↑ RMSE	AURG↑ d1
Baseline	24.2098	2.6367	0.0980	4.7962	10.3942	0.3066	0.7134	0.8775	0.9280	-	-	-	-	-	-
Deep Ensembles [123]	25.9658	1.8097	0.1009	4.7072	5.1183	0.3237	0.7091	0.8652	0.9174	0.1292	0.6917	0.2091	0.1164	3.1474	0.0067
MC Dropout [70]	25.3372	3.9252	0.0924	4.3635	22.9193	0.2971	0.7437	0.8829	0.9287	0.2062	0.9267	0.1843	0.0598	2.6365	0.0125
Single-PU [113]	27.3008	4.3492	0.1009	4.7161	28.5999	0.3284	0.7140	0.8638	0.9174	0.4815	0.7444	0.2104	-0.0210	3.1238	0.0039
SLURP [252]	24.2098	2.6366	0.0980	4.7962	10.3930	0.3066	0.7134	0.8775	0.9280	0.2116	1.0715	0.2229	0.0682	2.8043	-0.0116

Table A10: Supervised monocular depth results on **low adv. with OOD set**.

Methods	silog↓	AbsRel↓	log10↓	RMSE↓	SqRel↓	log_RMSE	d1↑	d2↑	d3↑	AUSE↓ AbsRel	AUSE↓ RMSE	AUSE↓ d1	AURG↑ AbsRel	AURG↑ RMSE	AURG↑ d1
Baseline	32.1516	0.4588	0.1448	6.9160	10.0794	0.4422	0.5549	0.7727	0.8587	-	-	-	-	-	-
Deep Ensembles [123]	37.4423	0.3308	0.1672	7.4105	2.7108	0.5183	0.5209	0.7277	0.8179	0.1509	1.0724	0.2720	0.0347	4.8398	0.0285
MC Dropout [70]	34.0965	0.5448	0.1351	6.1764	14.0074	0.4229	0.6096	0.7933	0.8672	0.3137	1.2454	0.2394	0.0811	3.7196	0.0288
Single-PU [113]	42.7338	0.3513	0.1735	7.6272	5.0461	0.5606	0.5289	0.7224	0.8106	0.1556	1.3474	0.2768	0.0611	4.7969	0.0232
SLURP [252]	32.1516	0.4588	0.1448	6.9160	10.0794	0.4422	0.5549	0.7727	0.8587	0.1514	1.5640	0.2737	0.1437	3.9450	0.0134

Table A11: Supervised monocular depth results on **high adv. with OOD set**.

Bibliography

[1] Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *ICDT*. 102

[2] Alemi, A. A., Fischer, I., and Dillon, J. V. (2018). Uncertainty in the variational information bottleneck. In *UAI*. 100

[3] Alhaija, H., Mustikovela, S., Mescheder, L., Geiger, A., and Rother, C. (2018). Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision (IJCV)*. 23, 88

[4] Allen-Zhu, Z. and Li, Y. (2022). Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. In *The Eleventh International Conference on Learning Representations*. 11

[5] Amini, A., Schwarting, W., Soleimany, A., and Rus, D. (2020). Deep evidential regression. *NeurIPS*. 4, 6, 7, 8, 23, 73, 75, 76, 83, 86

[6] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *ICLR*. 101

[7] Arnez, F., Espinoza, H., Radermacher, A., and Terrier, F. (2020). A comparison of uncertainty estimation approaches in deep learning components for autonomous vehicle applications. *arXiv preprint arXiv:2006.15172*. 71

[8] Assent, I. (2012). Clustering high dimensional data. *KDD*. 102

[9] baranidesign (2020). Rain drop size and speed of a falling rain drop. https://www.baranidesign.com/faq-articles/2020/1/19/rain-drop-size-and-speed-of-a-falling-rain-drop. [Online; accessed 06-October-2022]. 29

[10] Behrmann, J., Grathwohl, W., Chen, R. T., Duvenaud, D., and Jacobsen, J.-H. (2019). Invertible residual networks. In *ICML*. 101

[11] Beluch, W. H., Genewein, T., Nürnberger, A., and Köhler, J. M. (2018). The power of ensembles for active learning in image classification. In *CVPR*. xii, 65

[12] Bengs, V., Hüllermeier, E., and Waegeman, W. (2022). Pitfalls of epistemic uncertainty quantification through loss minimisation. *Advances in Neural Information Processing Systems*, 35:29205–29216. 9

[13] Besnier, V., Bursuc, A., Picard, D., and Briot, A. (2021). Triggering failures: Out-of-distribution detection by learning from local adversarial attacks in semantic segmentation. In *ICCV*. 14, 24, 72, 105

[14] Bevilacqua, M., Roumy, A., Guillemot, C., and Alberi-Morel, M. L. (2012). Low-complexity single-image super-resolution based on nonnegative neighbor embedding. *BMVC*. 83, 84

[15] Bhat, S. F., Alhashim, I., and Wonka, P. (2021). Adabins: Depth estimation using adaptive bins. In *CVPR*. xx, 59, 60, 61, 62, 63, 65, 67, 68, 69, 70, 76, 85

[16] Bishop, C. and Quazaz, C. (1996). Regression with input-dependent noise: A bayesian treatment. *NeurIPS*. 74

[17] Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*. Springer. 5, 73

[18] Blum, H., Sarlin, P.-E., Nieto, J., Siegwart, R., and Cadena, C. (2019). Fishyscapes: A benchmark for safe semantic segmentation in autonomous driving. In *ICCVW*. 24, 25

[19] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In *ICML*. xix, 10, 11, 24, 73

[20] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*. xi

[21] Boucheron, S., Lugosi, G., and Massart, P. (2013). *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press. 105

[22] Brechtel, S., Gindele, T., and Dillmann, R. (2014). Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps. In *17th international IEEE conference on intelligent transportation systems (ITSC)*, pages 392–399. IEEE. xiii

[23] Bruhn, A. and Weickert, J. (2006). A confidence measure for variational optic flow methods. *Computational Imaging and Vision*, 31:283. 42, 85

[24] Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. (2012). A naturalistic open source movie for optical flow evaluation. In *ECCV*. xxiii, 23, 44, 47, 48, 49

[25] Cabon, Y., Murray, N., and Humenberger, M. (2020). Virtual KITTI 2. *arXiv:2001.10773*. 25, 32

[26] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2020). nuScenes: A multimodal dataset for autonomous driving. In *CVPR*. 24

[27] Cao, W., Mirjalili, V., and Raschka, S. (2020). Rank consistent ordinal regression for neural networks with application to age estimation. *Pattern Recognition Letters*, 140:325–331. 76, 81

[28] Cao, Y., Wu, Z., and Shen, C. (2017). Estimating depth from monocular images as classification using deep fully convolutional residual networks. *TCSVT*. 59, 60, 62, 67, 68, 69, 70, 76

[29] Castillo, A., Escobar, M., Pérez, J. C., Romero, A., Timofte, R., Van Gool, L., and Arbelaez, P. (2021). Generalized real-world super-resolution through adversarial robustness. In *ICCV*. 83

[30] Chan, R., Lis, K., Uhlemeyer, S., Blum, H., Honari, S., Siegwart, R., Salzmann, M., Fua, P., and Rottmann, M. (2021). Segmentmeifyoucan: A benchmark for anomaly segmentation. *arXiv preprint arXiv:2104.14812*. 24, 25

[31] Chang, J.-R. and Chen, Y.-S. (2018). Pyramid stereo matching network. In *CVPR*. 63

[32] Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al. (2019). Argoverse: 3d tracking and forecasting with rich maps. In *CVPR*. 24

[33] Charpentier, B., Borchert, O., Zügner, D., Geisler, S., and Günnemann, S. (2022). Natural Posterior Network: Deep Bayesian Predictive Uncertainty for Exponential Family Distributions. In *ICLR*. 4, 73, 77

[34] Charpentier, B., Zügner, D., and Günnemann, S. (2020). Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts. *NeurIPS*. 4, 72, 73, 78

[35] Chen, G., Qiao, L., Shi, Y., Peng, P., Li, J., Huang, T., Pu, S., and Tian, Y. (2020). Learning open set network with discriminative reciprocal points. In *ECCV*. 101, 104

[36] Chen, Y.-H., Chen, W.-Y., Chen, Y.-T., Tsai, B.-C., Frank Wang, Y.-C., and Sun, M. (2017). No more discrimination: Cross city adaptation of road scene segmenters. In *ICCV*. 29

[37] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *CVPR*. 67

[38] Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. (2014). Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613. 23

[39] Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR. 15

[40] Corbière, C., Lafon, M., Thome, N., Cord, M., and Pérez, P. (2021a). Beyond first-order uncertainty estimation with evidential models for open-world recognition. In *ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning*. 4, 15, 72

[41] Corbière, C., Thome, N., Bar-Hen, A., Cord, M., and Pérez, P. (2019). Addressing failure prediction by learning model confidence. In *NeurIPS*. xix, 4, 13, 14, 35, 36, 40, 44, 72, 73, 79, 81, 100

[42] Corbière, C., Thome, N., Saporta, A., Vu, T.-H., Cord, M., and Pérez, P. (2021b). Confidence estimation via auxiliary models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6043–6055. 13, 35, 37

[43] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *CVPR*. 23, 24, 29, 43, 49

[44] Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*. 80

[45] Dai, D., Sakaridis, C., Hecker, S., and Van Gool, L. (2020). Curriculum model adaptation with synthetic and real data for semantic foggy scene understanding. *IJCV*. 24, 25

[46] Dai, D. and Van Gool, L. (2018). Dark model adaptation: Semantic image segmentation from daytime to nighttime. In *ITSC*. 24, 25

[47] Dempster, A. P. (1968). A generalization of bayesian inference. *Journal of the Royal Statistical Society: Series B (Methodological)*, 30(2):205–232. 6, 73

[48] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *CVPR*. 22, 83

[49] Denoeux, T. (2000). A neural network classifier based on dempster-shafer theory. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 30(2):131–150. 6

[50] Der Kiureghian, A. and Ditlevsen, O. (2009). Aleatory or epistemic? does it matter? *Structural safety*. 105

[51] Diaz, R. and Marathe, A. (2019). Soft labels for ordinal regression. In *CVPR*. 59, 60, 62, 64, 67, 68, 69, 70

[52] Dijk, T. v. and Croon, G. d. (2019). How do neural networks see depth in single images? In *ICCV*. 31, 106

[53] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*. xx, 62, 67, 76

[54] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *ICCV*. xxiii, 23, 37, 44, 45, 47, 48, 49

[55] Du, X., Wang, Z., Cai, M., and Li, Y. (2022). Vos: Learning what you don't know by virtual outlier synthesis. In *ICLR*. 17, 105

[56] Eigen, D., Puhrsch, C., and Fergus, R. (2014a). Depth map prediction from a single image using a multi-scale deep network. In *NeurIPS*. xxiii, 30, 32, 106

[57] Eigen, D., Puhrsch, C., and Fergus, R. (2014b). Depth map prediction from a single image using a multi-scale deep network. *NeurIPS*. xxiv, 38, 42, 43, 47, 49, 63, 66, 84, 85, 90

[58] Esser, P., Chiu, J., Atighehchian, P., Granskog, J., and Germanidis, A. (2023). Structure and content-guided video synthesis with diffusion models. *arXiv preprint arXiv:2302.03011*. xi

[59] Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *IJCV*. 23

[60] Foong, A. Y., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. (2019). 'in-between'uncertainty in bayesian neural networks. *arXiv preprint arXiv:1906.11537*. 80

[61] Fort, S., Hu, H., and Lakshminarayanan, B. (2019). Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*. 11, 105

[62] Franchi, G., Belkhir, N., Ha, M. L., Hu, Y., Bursuc, A., Blanz, V., and Yao, A. (2021). Robust semantic segmentation with superpixel-mix. In *BMVC*. 31

[63] Franchi, G., Bursuc, A., Aldea, E., Dubuisson, S., and Bloch, I. (2020a). Encoding the latent posterior of bayesian neural networks for uncertainty quantification. *arXiv:2012.02818*. 24

[64] Franchi, G., Bursuc, A., Aldea, E., Dubuisson, S., and Bloch, I. (2020b). TRADI: Tracking deep neural network weight distributions. In *ECCV*. 12

[65] Franchi, G., Bursuc, A., Aldea, E., Dubuisson, S., and Bloch, I. (2022a). One versus all for deep neural network for uncertainty (ovnni) quantification. *IEEE Access*, 10:7300–7312. 12, 69

[66] Franchi, G., Yu, X., Bursuc, A., Aldea, E., Dubuisson, S., and Filliat, D. (2022b). Latent discriminant deterministic uncertainty. In *ECCV*. xv, 73, 79, 83, 85, 86, 106

[67] Franchi, G., Yu, X., Bursuc, A., Tena, A., Kazmierczak, R., Dubuisson, S., Aldea, E., and Filliat, D. (2022c). Muad: Multiple uncertainties for autonomous driving benchmark for multiple uncertainty types and tasks. In *33rd British Machine Vision Conference, BMVC*. xv, 21, 23

[68] Fu, H., Gong, M., Wang, C., Batmanghelich, K., and Tao, D. (2018). Deep ordinal regression network for monocular depth estimation. In *CVPR*. xx, 59, 60, 61, 62, 64, 65, 67, 68, 69, 70, 76

[69] Gaidon, A., Wang, Q., Cabon, Y., and Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349. xxiii, 24, 30, 32

[70] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*. xix, xxiv, 4, 7, 10, 24, 30, 31, 36, 40, 41, 44, 64, 67, 68, 69, 72, 77, 106, 107, 113, 114

[71] Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31. 12

[72] Gast, J. and Roth, S. (2018). Lightweight probabilistic deep networks. In *CVPR*. 6

[73] Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., et al. (2023). A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, pages 1–77. xi, 26

[74] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013a). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237. xxiii, 43, 44, 47, 48, 49

[75] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013b). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*. xxiv, 23, 73, 84, 85

[76] Geiger, A., Lenz, P., and Urtasun, R. (2012a). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. xxiii, 47, 48

[77] Geiger, A., Lenz, P., and Urtasun, R. (2012b). Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*. 24, 29, 32

[78] Gidaris, S. and Komodakis, N. (2018). Dynamic few-shot visual learning without forgetting. In *CVPR*. 102

[79] Godard, C., Mac Aodha, O., and Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. In *CVPR*. 30

[80] Goldberg, P., Williams, C., and Bishop, C. (1997). Regression with input-dependent noise: A gaussian process treatment. *NeurIPS*. 74

[81] Gong, D., Liu, L., Le, V., Saha, B., Mansour, M. R., Venkatesh, S., and Hengel, A. v. d. (2019). Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *ICCV*. 100, 101

[82] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of wasserstein gans. In *NeurIPS*. 101

[83] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *ICML*. 19, 20

[84] Guo, Z., Wan, Z., Zhang, Q., Zhao, X., Chen, F., Cho, J.-H., Zhang, Q., Kaplan, L. M., Jeong, D. H., and Jøsang, A. (2022). A survey on uncertainty reasoning and quantification for decision making: Belief theory meets deep learning. *arXiv preprint arXiv:2206.05675*. 71

[85] Gustafsson, F. K., Danelljan, M., and Schon, T. B. (2020). Evaluating scalable bayesian deep learning methods for robust computer vision. In *CVPR Workshops*. 106

[86] Gustafsson, F. K., Danelljan, M., and Schön, T. B. (2023). How reliable is your regression model's uncertainty under real-world distribution shifts? *arXiv preprint arXiv:2302.03679*. 17

[87] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*. xx, 19, 39, 65, 67, 81, 83, 101

[88] Hendrycks, D., Basart, S., Mazeika, M., Mostajabi, M., Steinhardt, J., and Song, D. (2019a). A benchmark for anomaly segmentation. *arXiv e-prints*, pages arXiv–1911. 12

[89] Hendrycks, D., Basart, S., Mazeika, M., Mostajabi, M., Steinhardt, J., and Song, D. (2019b). A benchmark for anomaly segmentation. *arXiv preprint arXiv:1911.11132*. 21, 24, 25, 31

[90] Hendrycks, D. and Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*. 22, 23, 73, 83

[91] Hendrycks, D. and Gimpel, K. (2017). A baseline for detecting misclassified and out-of-distribution examples in neural networks. *ICLR*. 73, 106

[92] Hornauer, J. and Belagiannis, V. (2022). Gradient-based uncertainty for monocular depth estimation. In *ECCV*. xix, 4, 16, 72, 79, 86

[93] Houston, J., Zuidhof, G., Bergamini, L., Ye, Y., Chen, L., Jain, A., Omari, S., Iglovikov, V., and Ondruska, P. (2020). One thousand and one hours: Self-driving motion prediction dataset. In *arXiv*. 24

[94] Hu, D., Peng, L., Chu, T., Zhang, X., Mao, Y., Bondell, H., and Gong, M. (2022). Uncertainty quantification in depth estimation via constrained ordinal regression. In *European Conference on Computer Vision*, pages 237–256. Springer. 65, 66

[95] Hu, S., Pezzotti, N., and Welling, M. (2020). A new perspective on uncertainty quantification of deep ensembles. *arXiv e-prints*, pages arXiv–2002. 36

[96] Hu, S., Pezzotti, N., and Welling, M. (2021). Learning to predict error for mri reconstruction. In *MICCAI*. 14

[97] Hu, X., Fu, C.-W., Zhu, L., and Heng, P.-A. (2019). Depth-attentional features for single-image rain removal. In *CVPR*. 23, 43, 49

[98] Hu, Y., Li, Y., and Song, R. (2017). Robust interpolation of correspondences for large displacement optical flow. In *CVPR*. 42

[99] Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. (2017a). Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*. 12

[100] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017b). Densely connected convolutional networks. In *CVPR*. 39, 43, 44, 45, 46, 66, 106, 107

[101] Huang, Y., Bai, B., Zhao, S., Bai, K., and Wang, F. (2022). Uncertainty-aware learning against label noise on imbalanced datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*. xiii

[102] Hubmann, C., Becker, M., Althoff, D., Lenz, D., and Stiller, C. (2017). Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles. In *2017 IEEE intelligent vehicles symposium (IV)*, pages 1671–1678. IEEE. xiii

[103] Hui, T.-W., Tang, X., and Loy, C. C. (2018). Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *CVPR*. 44

[104] Ilg, E., Cicek, O., Galesso, S., Klein, A., Makansi, O., Hutter, F., and Brox, T. (2018). Uncertainty estimates and multi-hypotheses networks for optical flow. In *ECCV*. 12, 23, 29, 40, 42, 45, 106

[105] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. (2017). Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*. 44

[106] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *CVPR*. 36

[107] Jain, M., Lahlou, S., Nekoei, H., Butoi, V., Bertin, P., Rector-Brooks, J., Korablyov, M., and Bengio, Y. (2021). Deup: Direct epistemic uncertainty prediction. *arXiv preprint arXiv:2102.08501*. 15, 35, 36, 38, 72

[108] Joo, T., Chung, U., and Seo, M.-G. (2020). Being bayesian about categorical probability. In *ICML*. 4, 14, 15, 72, 73, 77, 79, 83

[109] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37:183–233. 10, 78

[110] Jøsang, A. (2016). *Subjective Logic: A Formalism for Reasoning Under Uncertainty*. Springer Publishing Company, Incorporated, 1st edition. 6

[111] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589. xi

[112] Kamann, C. and Rother, C. (2021). Benchmarking the robustness of semantic segmentation models with respect to common corruptions. *IJCV*. 73

[113] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *NeurIPS*. 4, 5, 6, 14, 24, 29, 31, 36, 40, 41, 44, 60, 64, 72, 74, 83, 85, 89, 105, 106, 107, 113, 114

[114] Khan, S., Hayat, M., Zamir, S. W., Shen, J., and Shao, L. (2019). Striking the right balance with uncertainty. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 103–112. xiii

[115] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*. 11

[116] Kondermann, C., Mester, R., and Garbe, C. (2008). A statistical confidence measure for optical flows. In *ECCV*. 42

[117] Kondratyuk, D., Tan, M., Brown, M., and Gong, B. (2020). When ensembling smaller models is more efficient than single large models. *arXiv preprint arXiv:2005.00570*. 12

[118] Kong, L., Niu, Y., Xie, S., Hu, H., Ng, L. X., Cottereau, B. R., Zhao, D., Zhang, L., Wang, H., Ooi, W. T., et al. (2023). The robodepth challenge: Methods and advancements towards robust depth estimation. *arXiv preprint arXiv:2307.15061*. 23

[119] Kopetzki, A.-K., Charpentier, B., Zügner, D., Giri, S., and Günnemann, S. (2021). Evaluating robustness of predictive uncertainty estimation: Are dirichlet-based models reliable? In *International Conference on Machine Learning*, pages 5707–5718. PMLR. 9

[120] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. Technical report. 22, 82

[121] Kuleshov, V., Fenner, N., and Ermon, S. (2018). Accurate uncertainties for deep learning using calibrated regression. In *International conference on machine learning*, pages 2796–2804. PMLR. 19, 20

[122] Kybic, J. and Nieuwenhuis, C. (2011). Bootstrap optical flow confidence and uncertainty measure. *Computer Vision and Image Understanding*, 115(10):1449–1462. 42

[123] Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*. xii, xix, xx, xxiv, 4, 11, 24, 30, 31, 36, 40, 41, 44, 60, 64, 65, 67, 68, 69, 70, 77, 79, 81, 83, 85, 86, 99, 106, 107, 113, 114

[124] Laurent, O., Lafage, A., Tartaglione, E., Daniel, G., Martinez, J.-M., Bursuc, A., and Franchi, G. (2022). Packed-ensembles for efficient uncertainty estimation. *ICLR*. 12

[125] Laves, M.-H., Ihler, S., Kortmann, K.-P., and Ortmaier, T. (2020). Calibration of model uncertainty for dropout variational inference. *arXiv preprint arXiv:2006.11584*. 21, 83

[126] LeCun, Y. (1998). The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*. 21, 23, 82

[127] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444. xi

[128] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*. 83, 84

[129] Lee, J. and AlRegib, G. (2020). Gradients as a measure of uncertainty in neural networks. In *ICIP*, pages 2416–2420. IEEE. 15, 16

[130] Lee, J. H., Han, M.-K., Ko, D. W., and Suh, I. H. (2019). From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*. xxiv, 30, 43, 44, 47, 66, 68, 84, 85, 89, 106, 107

[131] Lee, K., Lee, K., Lee, H., and Shin, J. (2018). A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31. 17

[132] Lee, S., Capuano, V., Harvard, A., and Chung, S.-J. (2020). Fast uncertainty estimation for deep learning based optical flow. In *IROS*. 36

[133] Lee, S., Purushwalkam, S., Cogswell, M., Crandall, D., and Batra, D. (2015). Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*. 12

[134] Levi, G. and Hassner, T. (2015). Age and gender classification using convolutional neural networks. In *CVPR workshops*. 59

[135] Lewis, D. D. (1995). A sequential algorithm for training text classifiers: Corrigendum and additional data. In *Acm Sigir Forum*, volume 29, pages 13–19. ACM New York, NY, USA. xii, xix

[136] Li, B., Dai, Y., and He, M. (2018a). Monocular depth estimation with hierarchical fusion of dilated cnns and soft-weighted-sum inference. *Pattern Recognition*. 59, 60, 62, 67, 68, 69, 70

[137] Li, B., Han, Z., Li, H., Fu, H., and Zhang, C. (2022). Trustworthy long-tailed classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6979. xiii

[138] Li, K., Li, Y., You, S., and Barnes, N. (2017). Photo-realistic simulation of road scene for data-driven methods in bad weather. In *ICCVW*. 26

[139] Li, R., Xian, K., Shen, C., Cao, Z., Lu, H., and Hang, L. (2018b). Deep attention-based classification network for robust depth prediction. In *ACCV*. 60, 61, 62, 67, 68

[140] Liang, S., Li, Y., and Srikant, R. (2018). Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR*. 17, 73

[141] Liebel, L. and Körner, M. (2019). Multidepth: Single-image depth estimation via multi-task regression and classification. In *ITSC*. 60, 62

[142] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988. 38

[143] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *ECCV*. 23

[144] Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B., and Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88. xi

[145] Liu, C.-L. and Nakagawa, M. (2001). Evaluation of prototype learning algorithms for nearest-neighbor classifier in application to handwritten character recognition. *PR*. 101

[146] Liu, J. Z., Lin, Z., Padhy, S., Tran, D., Bedrax-Weiss, T., and Lakshminarayanan, B. (2020a). Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In *NeurIPS*. 99, 100, 101

[147] Liu, W., Wang, X., Owens, J., and Li, Y. (2020b). Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*. xii, 7, 17

[148] Liu, Y., Cheng, M.-M., Hu, X., Wang, K., and Bai, X. (2017). Richer convolutional features for edge detection. In *CVPR*. 39

[149] Lo, C.-C. and Vandewalle, P. (2021). Depth estimation from monocular images and sparse radar using deep ordinal regression network. In *ICIP*. 60, 62

[150] Lobacheva, E., Chirkova, N., Kodryan, M., and Vetrov, D. P. (2020). On power laws in deep ensembles. *Advances In Neural Information Processing Systems*, 33:2375–2385. 12

[151] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *CVPR*. xx, xxiv, 65, 66, 67, 68

[152] Mac Aodha, O., Humayun, A., Pollefeys, M., and Brostow, G. J. (2012). Learning a confidence measure for optical flow. *IEEE transactions on pattern analysis and machine intelligence*, 35(5):1107–1120. 21

[153] Macêdo, D., Ren, T. I., Zanchettin, C., Oliveira, A. L., and Ludermir, T. (2021). Entropic out-of-distribution detection. In *IJCNN*. 102, 103

[154] Macêdo, D., Zanchettin, C., and Ludermir, T. (2022). 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*. 104

[155] Magnor, M. A., Grau, O., Sorkine-Hornung, O., and Thebalt, C. (2015). *Digital Representations of the Real World*. CRC Press. 26

[156] Malinin, A., Chervontsev, S., Provilkov, I., and Gales, M. (2020a). Regression prior networks. *arXiv preprint arXiv:2006.11590*. 4, 7, 23, 73, 77

[157] Malinin, A. and Gales, M. (2018). Predictive uncertainty estimation via prior networks. *NeurIPS*. 4, 6, 13, 14, 73, 76, 77, 105

[158] Malinin, A. and Gales, M. (2019). Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness. *NeurIPS*. 73

[159] Malinin, A., Mlodozeniec, B., and Gales, M. (2020b). Ensemble distribution distillation. In *ICLR*. 12, 105

[160] Mandelbaum, A. and Weinshall, D. (2017). Distance-based confidence score for neural network classifiers. *arXiv preprint arXiv:1709.09844*. 100, 104

[161] Marks, R. J., Wise, G. L., Haldeman, D. G., and Whited, J. L. (1978). Detection in laplace noise. *IEEE Transactions on Aerospace and Electronic Systems*. 74

[162] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*. 83, 84

[163] Mateos, C. (2020). All synthetic data is not made equal. https://anyverse.ai/synthetic-data/synthetic-data-object-detection/. [Online; accessed 06-October-2022]. 29

[164] Meinert, N., Gawlikowski, J., and Lavin, A. (2023). The unreasonable effectiveness of deep evidential regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9134–9142. 8, 9

[165] Menze, M., Heipke, C., and Geiger, A. (2015). Joint 3d estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*. xxiii, 44, 48, 49

[166] Menze, M., Heipke, C., and Geiger, A. (2018). Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*. xxiii, 44, 48, 49

[167] Mi, L., Wang, H., Tian, Y., and Shavit, N. (2022). Training-free uncertainty estimation for dense regression: Sensitivity as a surrogate. In *AAAI*. xix, 4, 15, 16, 72, 79, 87, 106, 107

[168] Michael I., J. (2010). The exponential family: Conjugate priors. 7

[169] Michaelis, C., Mitzkus, B., Geirhos, R., Rusak, E., Bringmann, O., Ecker, A. S., Bethge, M., and Brendel, W. (2019). Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*. 23, 73

[170] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *ICLR*. 101

[171] Mukhoti, J., Kirsch, A., van Amersfoort, J., Torr, P. H., and Gal, Y. (2023). Deep deterministic uncertainty: A new simple baseline. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24384–24394. 24, 99, 100, 101, 105

[172] Mukhoti, J., Kulharia, V., Sanyal, A., Golodetz, S., Torr, P. H., and Dokania, P. K. (2020). Calibrating deep neural networks using focal loss. In *NeurIPS*. 24

[173] Muramoto, K., Shiina, T., Endoh, T., Konishi, H., Kitano, K., et al. (1989). Measurement of snowflake size and falling velocity by image processing. 29

[174] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press. xi

[175] Nadarajah, S. (2005). A generalized normal distribution. *Journal of Applied statistics*, 32(7):685–694. 74

[176] Nathan Silberman, Derek Hoiem, P. K. and Fergus, R. (2012). Indoor segmentation and support inference from RGBD images. In *ECCV*. 21, 23, 59, 87, 89

[177] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NeurIPS*. 82

[178] Neuhold, G., Ollmann, T., Rota Bulo, S., and Kontschieder, P. (2017). The mapillary vistas dataset for semantic understanding of street scenes. In *CVPR*. 24

[179] Niu, Z., Zhou, M., Wang, L., Gao, X., and Hua, G. (2016). Ordinal regression with multiple output cnn for age estimation. In *CVPR*. xxiv, 81, 82

[180] Nix, D. and Weigend, A. (1994). Estimating the mean and variance of the target probability distribution. In *ICNN*. 4, 5, 6, 29, 72, 74, 83, 89

[181] Oberdiek, P., Rottmann, M., and Gottschalk, H. (2018). Classification uncertainty of deep neural networks based on gradient information. In *Artificial Neural Networks in Pattern Recognition: 8th IAPR TC3 Workshop, ANNPR 2018, Siena, Italy, September 19–21, 2018, Proceedings 8*, pages 113–125. Springer. 15

[182] Oh, D. and Shin, B. (2022). Improving evidential deep learning via multi-task learning. In *AAAI*. 8, 9, 73

[183] OpenAI (2023). Gpt-4 technical report. *ArXiv*, abs/2303.08774. xi

[184] Padhy, S., Nado, Z., Ren, J., Liu, J., Snoek, J., and Lakshminarayanan, B. (2020). Revisiting one-vs-all classifiers for predictive uncertainty and out-of-distribution detection in neural networks. In *ICML Workshops*. 102

[185] Parisi, G. and Shankar, R. (1988). Statistical field theory. 7

[186] Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. V. (2012). Cats and dogs. In *CVPR*. 82

[187] 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., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 82

[188] Phan, M. H., Phung, S. L., and Bouzerdoum, A. (2021). Ordinal depth classification using region-based self-attention. In *ICPR*. 60, 62

[189] Pinggera, P., Ramos, S., Gehrig, S., Franke, U., Rother, C., and Mester, R. (2016). Lost and found: detecting small road hazards for self-driving vehicles. In *IROS*. 24, 25

[190] Pinto, F., Yang, H., Lim, S.-N., Torr, P., and Dokania, P. K. (2021). Mix-maxent: Improving accuracy and uncertainty estimates of deterministic neural networks. In *NeurIPS Workshops*. 99

[191] Poggi, M., Aleotti, F., Tosi, F., and Mattoccia, S. (2020). On the uncertainty of self-supervised monocular depth estimation. In *CVPR*. 6, 14, 17, 21, 23, 30, 42, 44, 60, 64, 67, 85, 106

[192] Poggio, T. and Girosi, F. (1989). A theory of networks for approximation and learning. Technical report, Massachusetts INST of TECH Cambridge Artificial Intelligence LAB. 6

[193] Postels, J., Blum, H., Strümpler, Y., Cadena, C., Siegwart, R., Van Gool, L., and Tombari, F. (2021a). The hidden uncertainty in a neural networks activations. In *ICML*. 99, 100, 101

[194] Postels, J., Segu, M., Sun, T., Van Gool, L., Yu, F., and Tombari, F. (2021b). On the practicality of deterministic epistemic uncertainty. *arXiv preprint arXiv:2107.00649*. 6, 99

[195] Qi, H., Brown, M., and Lowe, D. G. (2018). Low-shot learning with imprinted weights. In *CVPR*. 102

[196] Qu, H., Foo, L. G., Li, Y., and Liu, J. (2023). Towards more reliable confidence estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 14

[197] Qu, H., Li, Y., Foo, L. G., Kuen, J., Gu, J., and Liu, J. (2022). Improving the reliability for confidence estimation. In *ECCV*. 4, 14, 36, 72, 75

[198] Ramanishka, V., Chen, Y.-T., Misu, T., and Saenko, K. (2018). Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning. In *CVPR*. 24

[199] Razavi, A., Van den Oord, A., and Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. In *NeurIPS*. 101

[200] Ren, H., El-Khamy, M., and Lee, J. (2019). Deep robust single image depth estimation neural network using scene understanding. In *CVPR Workshops*. 60, 61, 62, 63, 67, 68, 69, 70

[201] Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *ICML*. 15, 72

[202] Richter, S. R., Vineet, V., Roth, S., and Koltun, V. (2016). Playing for data: Ground truth from computer games. In *ECCV*. 24, 29

[203] Rohrer, B. (2022). Sharpened cosine similarity. `https://github.com/brohrer/sharpened-cosine-similarity`. 107

[204] Ros, G., Sellart, L., Materzynska, J., Vazquez, D., and Lopez, A. M. (2016). The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*. 24, 29

[205] Sakaridis, C., Dai, D., Hecker, S., and Van Gool, L. (2018a). Model adaptation with synthetic and real data for semantic dense foggy scene understanding. In *ECCV*. 43, 49

[206] Sakaridis, C., Dai, D., and Van Gool, L. (2018b). Semantic foggy scene understanding with synthetic data. *IJCV*. 23, 24, 25

[207] Sakaridis, C., Dai, D., and Van Gool, L. (2020). Map-guided curriculum domain adaptation and uncertainty-aware evaluation for semantic nighttime image segmentation. *arXiv preprint arXiv:2005.14553*. 24, 25

[208] Sakaridis, C., Dai, D., and Van Gool, L. (2021). Acdc: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *ICCV*. 24, 25

[209] Sensoy, M., Kaplan, L., and Kandemir, M. (2018). Evidential deep learning to quantify classification uncertainty. *NeurIPS*. 4, 6, 7, 9, 14, 72, 73, 75, 77, 78

[210] Settles, B. (2009). Active learning literature survey. xi, xii, xix

[211] Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55. xii, 65, 90

[212] Shen, M., Bu, Y., Sattigeri, P., Ghosh, S., Das, S., and Wornell, G. (2023). Post-hoc uncertainty learning using a dirichlet meta-model. In *AAAI*. 14, 35, 72, 73, 78

[213] Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *ECCV*. 43

[214] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489. xi

[215] Sinha, P. and Adelson, E. (1993). Recovering reflectance and illumination in a world of painted polyhedra. In *ICCV*. 59

[216] Snell, J., Swersky, K., and Zemel, R. S. (2017). Prototypical networks for few-shot learning. In *NeurIPS*. 102, 103

[217] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958. 11, 15, 72

[218] Sternberg, R. J. and Stemberg, K. (2012). *Cognitive Psychologh*. xi

[219] Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. (2018). PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*. 39, 44

[220] Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. (2019). 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. 36

[221] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D. (2020). Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*. 24

[222] Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*. 67

[223] Techapanurak, E. and Okatani, T. (2021). Practical evaluation of out-of-distribution detection methods for image classification. *arXiv preprint arXiv:2101.02447*. 21, 23, 73, 82

[224] Tong, Z., Xu, P., and Denoeux, T. (2021). An evidential classifier based on dempster-shafer theory and deep learning. *Neurocomputing*, 450:275–293. 6

[225] Tremblay, M., Halder, S. S., de Charette, R., and Lalonde, J.-F. (2021). Rain rendering for evaluating and improving robustness to bad weather. *ijcv*. 25, 26

[226] Tung, F., Chen, J., Meng, L., and Little, J. J. (2017). The raincouver scene parsing benchmark for self-driving in adverse weather and at night. 24, 25

[227] Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., and Geiger, A. (2017). Sparsity invariant CNNs. In *3DV*. 23, 43, 49, 59, 66, 73, 84, 106

[228] Ulmer, D., Hardmeier, C., and Frellsen, J. (2023). Prior and posterior networks: A survey on evidential deep learning methods for uncertainty estimation. *Transactions on Machine Learning Research*. 4, 6, 73, 75, 100

[229] Upadhyay, U., Karthik, S., Chen, Y., Mancini, M., and Akata, Z. (2022). Bayescap: Bayesian identity cap for calibrated uncertainty in frozen neural networks. In *ECCV*. 4, 14, 39, 72, 73, 74, 75, 83, 84

[230] van Amersfoort, J., Smith, L., Jesson, A., Key, O., and Gal, Y. (2021). On feature collapse and deep kernel learning for single forward pass uncertainty. *arXiv preprint arXiv:2102.11409*. 99, 100

[231] Van Amersfoort, J., Smith, L., Teh, Y. W., and Gal, Y. (2020). Uncertainty estimation using a single deep deterministic neural network. In *ICML*. 99, 100

[232] Van Den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. In *NeurIPS*. 101

[233] Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. (2018). The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778. 23

[234] Varma, G., Subramanian, A., Namboodiri, A., Chandraker, M., and Jawahar, C. (2019). Idd: A dataset for exploring problems of autonomous navigation in unconstrained environments. In *WACV*. 24

[235] Veach, E. (1998). *Robust Monte Carlo methods for light transport simulation*. Stanford University. 26

[236] Von Bernuth, A., Volk, G., and Bringmann, O. (2019). Simulating photo-realistic snow and fog on existing images for enhanced cnn training and evaluation. In *ITSC*. 26

[237] Wen, Y., Tran, D., and Ba, J. (2020). BatchEnsemble: an alternative approach to efficient ensemble and lifelong learning. In *ICLR*. 4, 12, 24

[238] Wen, Y., Zhang, K., Li, Z., and Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. In *ECCV*. 101, 104

[239] Wu, M. and Goodman, N. (2020). A simple framework for uncertainty in contrastive learning. *arXiv preprint arXiv:2010.02038*. 100

[240] Xia, G. and Bouganis, C.-S. (2022). Augmenting softmax information for selective classification with out-of-distribution data. In *Proceedings of the Asian Conference on Computer Vision*, pages 1995–2012. 17

[241] Xia, X., Liu, T., Han, B., Gong, M., Yu, J., Niu, G., and Sugiyama, M. (2021). Sample selection with uncertainty of losses for learning with noisy labels. *arXiv preprint arXiv:2106.00445*. xiii

[242] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*. 82

[243] Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. (2010). Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE. 23

[244] Yang, G., Hu, P., and Ramanan, D. (2019). Inferring distributions over depth from a single image. In *IROS*. xx, 59, 60, 62, 63, 65, 67, 68, 69, 70

[245] Yang, G. and Ramanan, D. (2019). Volumetric correspondence networks for optical flow. *NeurIPS*, 5:12. 44

[246] Yang, H.-M., Zhang, X.-Y., Yin, F., and Liu, C.-L. (2018). Robust classification with convolutional prototype learning. In *CVPR*. 101, 104

[247] Yang, J., Zhou, K., Li, Y., and Liu, Z. (2021). Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*. 21

[248] Yeo, T., Kar, O. F., and Zamir, A. (2021). Robustness via cross-domain ensembles. In *CVPR*. 73

[249] Yin, W., Liu, Y., Shen, C., and Yan, Y. (2019). Enforcing geometric constraints of virtual normal for depth prediction. In *ICCV*. 60

[250] Yoo, D. and Kweon, I. S. (2019). Learning loss for active learning. In *CVPR*. 36

[251] Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., and Darrell, T. (2020). Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*. 24, 25

[252] Yu, X., Franchi, G., and Aldea, E. (2021). SLURP: Side learning uncertainty for regression problems. In *BMVC*. xiv, xix, 4, 13, 15, 23, 30, 31, 60, 64, 67, 71, 72, 73, 74, 84, 100, 106, 113, 114

[253] Yu, X., Franchi, G., and Aldea, E. (2022). On monocular depth estimation and uncertainty quantification using classification approaches for regression. In *ICIP*. xv, 71, 76, 81

[254] Yu, X., Franchi, G., Gu, J., and Aldea, E. (2024). Discretization-induced dirichlet posterior for robust uncertainty quantification on regression. In *The 38th Annual AAAI Conference on Artificial Intelligence*. xv, 9, 15, 22, 23

[255] Yu, X., Zuo, Y., Wang, Z., Zhang, X., Zhao, J., Yang, Y., Jiao, L., Peng, R., Wang, X., Zhang, J., et al. (2023). The robust semantic segmentation uncv2023 challenge results. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4618–4628. xv

[256] Yuan, W., Gu, X., Dai, Z., Zhu, S., and Tan, P. (2022). NeW CRFs: Neural window fully-connected CRFs for monocular depth estimation. In *CVPR*. xxiii, 29, 30, 31

[257] Zendel, O., Honauer, K., Murschitz, M., Steininger, D., and Dominguez, G. F. (2018). Wilddash-creating hazard-aware benchmarks. In *ECCV*. 25

[258] Zeyde, R., Elad, M., and Protter, M. (2012). On single image scale-up using sparse-representations. In *ICCS*. 83, 84

[259] Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., and Torralba, A. (2017). Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464. 23

[260] Zhou, H., Greenwood, D., and Taylor, S. (2021). Self-supervised monocular depth estimation with internal feature fusion. In *British Machine Vision Conference (BMVC)*. xxiii, 30, 32

[261] Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC press. xii