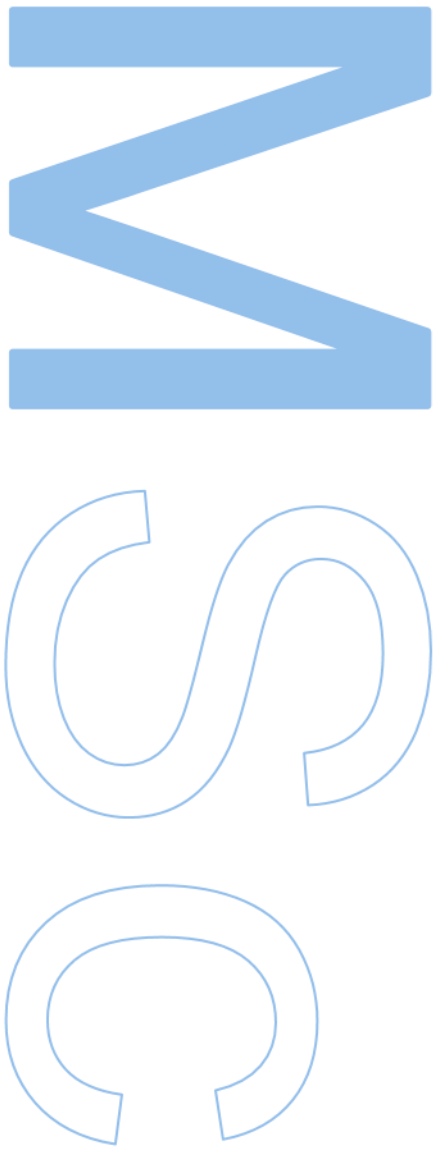
Yes, No, Maybe: Uncertainty Estimation in Autonomous Driving

Luís Miguel Mendonça Almeida
Mestrado em Data Science
Departamento de Ciência de Computadores
2023

Supervisor
Inês de Castro Dutra, Professora Auxiliar, Faculdade de Ciências da Universidade do Porto




U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Abstract

Recent advancements in AI have made fully autonomous driving closer. Usually, self driving systems are divided into multiple modules that are responsible for different tasks. The perception module is one of the most important, as the self driving vehicle needs to be able to accurately and robustly capture the environment surrounding it. Thus, a variety of sensors and deep learning models are used for this task. Unfortunately, said sensors and models do not perform as well under adverse weather conditions as under clear weather conditions, which impacts the performance of the whole self driving system. Hence, techniques that take the uncertainty that arises in these systems under these weather conditions into account are crucial to safe autonomous driving.

This work consists in a detailed review of autonomous driving datasets used for semantic segmentation tasks, state of the art uncertainty quantification methods and semantic segmentation models, with a focus on adverse weather conditions. After this review, a simple uncertainty aware training method for semantic segmentation models is presented. This method consists in the regular training of five individual models that are combined in an ensemble to generate uncertainty masks for each of the training samples. The same models are then trained again using these uncertainty masks to weight the training samples, encouraging the models to better learn classes and pixels with higher uncertainty values. Results suggest that this method improves the performance of the ensemble and leads it to predict high quality uncertainty values even under domain shift, which is important when driving under adverse weather conditions.

Keywords: Autonomous Driving; Deep Learning; Uncertainty Aware; Uncertainty Quantification; Semantic Segmentation; Adverse Weather

Resumo

Avanços recentes na IA têm tornado a condução totalmente autónoma cada vez mais próxima. Normalmente, os sistemas de condução autónoma são divididos em vários módulos responsáveis por diferentes tarefas. O módulo de perceção é um dos mais importantes, já que o veículo autónomo precisa de ser capaz de capturar com precisão e robustez o ambiente ao seu redor. Portanto, uma variedade de sensores e modelos de aprendizagem profunda são usados para essa tarefa. Infelizmente, estes sensores e modelos não funcionam tão bem em condições meteorológicas adversas quanto em condições meteorológicas limpas, o que afeta o desempenho de todo o sistema de condução autónoma. Assim, técnicas que levem em conta a incerteza que surge nestes sistemas sob condições meteorológicas adversas são cruciais para condução autónoma segura.

Este trabalho consiste numa revisão detalhada de literatura de datasets de condução autónoma usados para tarefas de segmentação semântica, métodos de quantificação de incerteza e modelos de segmentação semântica, com foco em condições meteorológicas adversas. Após esta revisão, é apresentado um método de treino simples para modelos de segmentação semântica que leva em conta incerteza. Este método consiste no treino normal (sem incerteza) de cinco modelos individuais que são combinados num ensemble para gerar máscaras de incerteza para cada uma das instâncias de treino. Em seguida, os mesmos modelos são treinados novamente usando estas máscaras de incerteza para pesar as instâncias de treino, incentivando os modelos a aprender melhor as classes e os pixeis com valores de incerteza mais altos. Os resultados obtidos sugerem que este método melhora o desempenho do ensemble e leva-o a prever valores de incerteza de alta qualidade, mesmo sob mudanças de domínio, o que é importante para a condução autónoma em condições meteorológicas adversas.

Palavras Chave: Condução Autónoma; Aprendizagem Profunda; Consciente de Incerteza; Quantificação de Incerteza; Segmentação Semântica; Condições Meteorológicas Adversas

Agradecimentos

I would like to thank my supervisor, professor Inês de Castro Dutra, for all the advice and insights she provided me with. The discussions that we had were crucial for the writing of this thesis, as well as her availability and support. I would also like to thank Bosch Car Multimedia SA, Ricardo Cerqueira and the team working in project Theia: Automated Perception Driving, for the opportunity of working in an exciting project and for the support and guidance in the elaboration of this work.

Additionally, I would like to thank my parents as well as my family for their unconditional support and patience, which made this work possible.

Finally, my friends were very encouraging, supportive, and were very important in helping me accomplish this work. Thank you!

This work is supported by European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project nº 047264; Funding Reference: POCI-01-0247-FEDER-047264].

To my parents

Contents

List of Tables

List of Figures

Listings

Acronyms

BNN	Bayesian Neural Network
CNN	Convolutional Neural Network
FCN	Fully Connected Network
MC	Monte Carlo
NN	Neural Network
OOD	Out Of Distribution
AI	Artifical Intelligence

Chapter 1

Introduction

Autonomous driving has long been the subject of dreams of many scientists and engineers. With each passing year, a world where we do not need to drive to work, where we do not need to drive to get our medicine, where we do not need to drive after having dinner with our friends, becomes closer to reality. According to [90], in 2019 alone, 1.3 million people worldwide died due to transport accidents. Safe and reliable autonomous driving, when it is achieved, has the potential to reduce this death toll to zero and also greatly improve the lives of millions of people with disabilities or that are simply unable to drive. Although self driving cars have been contemplated for many years, it was only recently that there were major advancements in the fields of AI and deep learning that fueled great progress in their pursuit. Thus, it is unsurprising to find that several companies have started a full-on race to solve autonomous driving, with projections signalling that the competition will only get fiercer [98]. In spite of the usage of more and more advanced sensors and deep learning algorithms, these systems are not completely reliable and exact, especially under adverse weather conditions, which justifies the need to account for the uncertainty associated with them. In this way, decision making systems will be more solid and lead to safer and more trustworthy autonomous driving.

1.1 Context

The system of a self driving car can usually be decomposed into a combination of different modules, each of them in charge of a different task [50]. The perception module must accurately and reliably capture the surrounding environment of the vehicle. This is done by using a variety of different sensors, such as LiDAR, GPS, radar and video cameras. This information is used to locate the vehicle, check for road signs and detect potential obstacles on its path. In other words, the perception module is the interface of the self driving car with the environment. After capturing the surrounding environment and locating the vehicle, the information collected by the sensors goes through a decision making module. This module is responsible for the decision of the next course of action that the self driving car will take, given the collected data. This involves planning the path that the vehicle will take as well as the actions needed to follow it,

such as turning the wheels, the amount of gas needed, etc. An illustration of these modules can be seen in Figure 1.1.

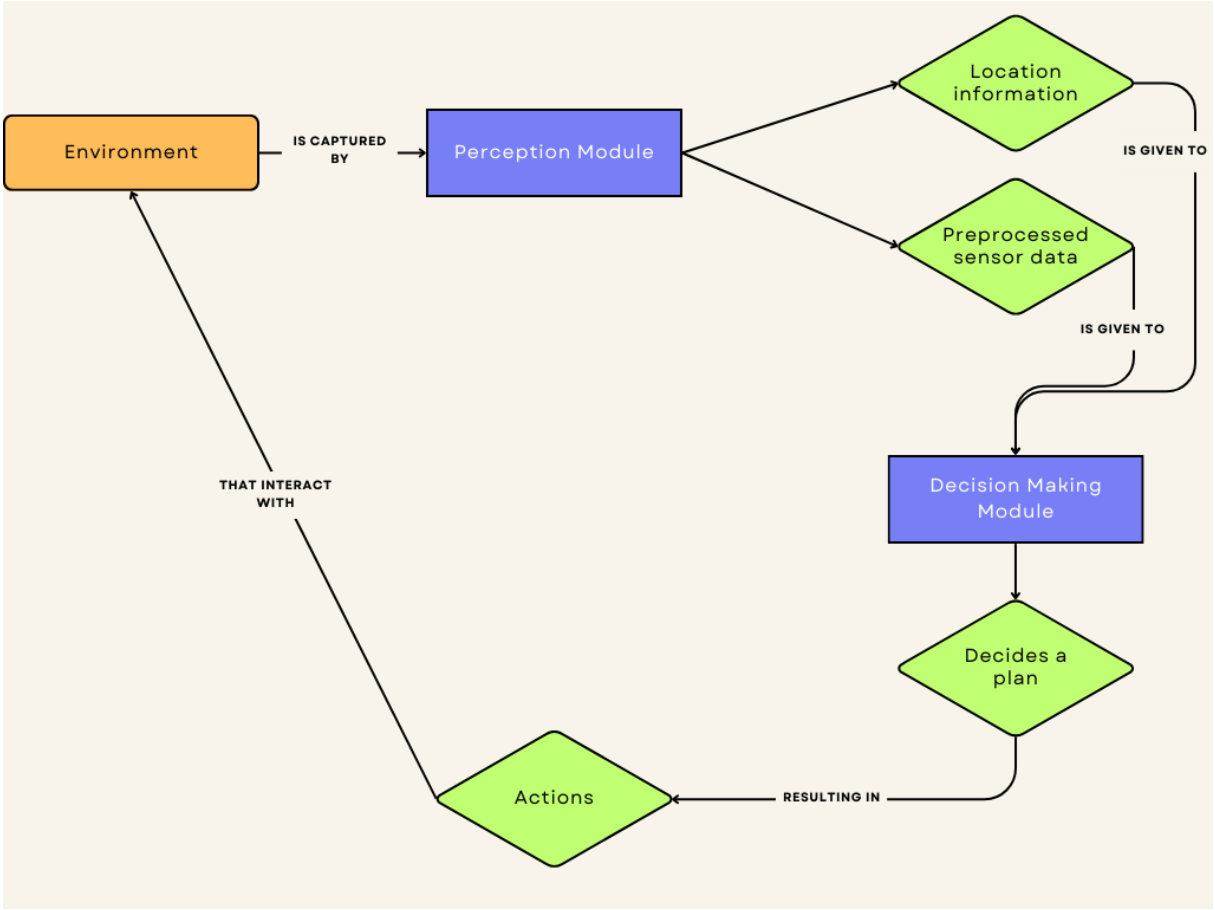


Figure 1.1: The architecture of a self driving car system. A perception module captures the raw data from the sensors, preprocesses it and locates the vehicle. A decision making module then uses said data to produce a motion plan and the corresponding actions to follow it.

1.2 Motivation

Autonomous Driving systems, as well as the deep learning models that are used in them, are not flawless and errors may occur. Unfortunately, situations like [103], [77] and [87] are not uncommon and efforts should be taken to prevent them. Usual causes of these accidents are sensor failures and wrong predictions by the deep learning perception models, which can be aggravated under adverse weather conditions. Deep learning perception models that are trained on clear weather data tend to produce overconfident predictions under adverse weather conditions, as this presents a large domain shift in the data that the models are trained on. Thus a way of modelling and dealing with the uncertainty associated with these components must be included in the decision making module.

1.3 Objectives

The main objectives for this work are the study of uncertainty quantification techniques in deep learning and semantic segmentation under adverse weather conditions in order to develop a method to improve the predictions produced by a semantic segmentation model. Thus, the first step involves applying an uncertainty quantification technique to the task of semantic segmentation. The second step consists in using these uncertainty predictions to boost the performance of a semantic segmentation model. Improving the performance of these perception models allows the decision making module of the self driving system to be more confident in the predictions given by the perception module.

1.4 Contributions

The main contributions of this work are a solid literature review of current state of the art solutions for uncertainty estimation and semantic segmentation under adverse weather conditions and the development of a simple training method for semantic segmentation models that estimates their uncertainty associated with each pixel of an image on the training set, using this information to boost the performance of said models on harder to segment classes.

1.5 Document Structure

This document was sorted into seven chapters. Besides the current Introduction chapter, the document presents the following structure:

- Background - This is used to introduce some basic and fundamental concepts related to the work that was developed.
- State of the Art - An extensive literature review across the fields of autonomous driving datasets, uncertainty quantification and semantic segmentation is presented in this chapter.
- Design an Development - Here, the method that was developed will be presented, as well as the materials and tools that were used to implement and test it.
- Experiments and Tests - In this chapter all the experiments that were conducted, accompanied by the reason that they were performed, will be presented.
- Results and Analysis - The results obtained by each of the experiments are critically analyzed here.
- Conclusions - An overall review of the work that was conducted is presented, with considerations on the developed method and suggestions for future work.

Chapter 2

Background

In this chapter, some concepts and definitions relevant to this work will be introduced. First, the fundamentals of modelling uncertainty in machine learning will be presented, followed by the definition of model uncertainty calibration and finally the introductory concepts of semantic segmentation.

2.1 Uncertainty in Machine Learning

Uncertainty plays a crucial role in Machine Learning, as its purpose is to learn models that are able to generalize from training data [49]. In applications such as autonomous driving, where important safety requirements must be met, uncertainty should be well represented. Uncertainty can be decomposed into two different components according to their source, often referred to as Aleatoric Uncertainty and Epistemic Uncertainty [57]. Aleatoric Uncertainty encompasses the uncertainty that stems from the data, mainly due to noise and the complexity of the process that generated it [67]. Epistemic Uncertainty refers to the uncertainty that arises from ignorance about the best model, that is, from unawareness about the best model assumptions and uncertainty about its parameters.

$$\text{Total Uncertainty} = \text{Aleatoric Uncertainty} + \text{Epistemic Uncertainty}$$

Another way to grasp these concepts is to think of Aleatoric Uncertainty as the irreducible part of Total Uncertainty, whereas Epistemic Uncertainty accounts for the reducible part. To exemplify an Aleatoric Uncertainty source, consider the problem of predicting heads or tails from an unbiased coin toss. The best model for this problem produces probabilities of $\frac{1}{2}$ for each of the outcomes and is not able to reduce this uncertainty, as it is caused by the stochastic nature of the process. Epistemic Uncertainty can be illustrated by a situation where the data used to train a model is different from the data used to test it. Here, uncertainty about the best model parameters is high but can be reduced by increasing the amount of training data.

In critical applications such as autonomous driving and medical diagnosis, the uncertainty associated with an individual instance is pratically more relevant than more general average

metrics [49]. The uncertainty related to the prediction $\hat{y}$ of a model f_θ for an instance x is called Predictive Uncertainty and is the focus of many uncertainty representation and quantification methods.

Uncertainty Quantification is the problem of computing both Aleatoric and Epistemic uncertainty values, which involves modelling these concepts in some way. There are several options to achieve this, but most Deep Learning works follow the Bayesian Neural Network approach. BNNs are especially important for the topic of uncertainty as instead of learning a set of fixed weights like a NN would, BNNs learn a distribution of their weights, which also induces a distribution on the prediction [13], given by (2.1). The goal of a BNN is to learn the posterior distribution of the weights w given the training data D, that is, $P(w|D)$, which can be achieved by Bayesian inference [10]. Throughout this work, model predictions will be denoted by the usage of a circumflex accent. From equations (2.1) to (2.4), x represents an input sample while $\hat{y}$ represents the vector of class confidences as predicted by a model.

$$P(\hat{y}|x, D) = \mathbb{E}_{P(w|D)}[P(\hat{y}|x, w)] \tag{2.1}$$

This distribution on the prediction can then be used to reason about uncertainty, for example with its variance or entropy. In practice, exact Bayesian inference is intractable, and approximations are used. Predictive Uncertainty can thus be defined as (2.2).

$$\text{Predictive Uncertainty} = H[\mathbb{E}_{P(w|D)}[P(\hat{y}|x, w)]] \tag{2.2}$$

and Aleatoric Uncertainty (2.3) is defined as the entropy H of the true conditional distribution $P(y|x)$ [67]. Here, y denotes the ground truth vector of probabilities that x belongs to each class.

$$\text{Aleatoric Uncertainty} = H[P(y|x)] \tag{2.3}$$

Although directly quantifying and modelling Epistemic Uncertainty is not simple, it is then given by the difference between the total Predictive and Aleatoric uncertainty in (2.4) [49].

$$\text{Epistemic Uncertainty} = H[\mathbb{E}_{P(w|D)}[P(\hat{y}|x, w)]] - H[P(y|x)] \tag{2.4}$$

2.2 Model Uncertainty Calibration

The degree of confidence in a Machine Learning model's predictions is an important issue in many applications and is very relevant to the problem of Uncertainty Quantification. When classifying an instance, a lot of models output a probability-like score of that instance belonging to a certain class. Unfortunately, a lot of times these scores are uncalibrated and cannot be interpreted as probabilities of class membership. Uncertainty calibration is the problem of predicting probability estimates representative of the true correctness likelihood [35]. Let Y be a random variable corresponding to the ground truth class labels. Let $\hat{Y}$, $\hat{P}$ be random variables corresponding to a model's class label and associated confidence predictions, respectively. Perfect calibration is given by (2.5).

$$\mathbb{P}(\hat{Y} = Y \mid \hat{P} = p) = p, \quad \forall p \in [0, 1] \tag{2.5}$$

Some metrics that assess the degree of calibration in a model's predictions will be presented. Expected Calibration Error (ECE) (2.6) models miscalibration as the expected difference between confidence and accuracy.

$$\mathbb{E}[|\mathbb{P}(\hat{Y} = Y \mid \hat{P} = p) - p|] \tag{2.6}$$

In practice, (2.6) can only be estimated. To achieve this, [74] follows a binning approach that divides the model's n predictions into M bins and uses them to compute accuracy (2.7) and confidence (2.8) metrics. These are then used to compute (2.9).

$$acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} 1(\hat{y}_i = y_i) \tag{2.7}$$

$$conf(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i \tag{2.8}$$

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)| \tag{2.9}$$

The Maximum Calibration Error (MCE) (2.10) can also be estimated with this approach, corresponding to the maximum difference between the accuracy and confidence of all bins. This metric may be used in conjunction with ECE to provide more information about the calibration error, as in critical applications a large MCE may not be tolerated.

$$MCE = \max |acc(B_m) - conf(B_m)|, \quad \forall m \in M \tag{2.10}$$

2.3 Semantic Segmentation

Semantic Segmentation can be viewed as a classification problem where the goal is to classify each pixel of an image as belonging to one of C classes. Thus, the output of a semantic segmentation method is a segmentation mask where each pixel is assigned a class. An example of a segmentation mask is presented in Figure 2.1. Semantic Segmentation allows for fine-level scene understanding, which is crucial in domains such as medical imaging and autonomous driving.

The performance of Semantic Segmentation methods is usually evaluated with the Intersection-Over-Union (IoU) metric. For a given class c, its IoU value is calculated by dividing the number of correctly classified class c pixels over the sum of the number of class c pixels in the ground truth and the number of incorrectly classified pixels as class c. In a multiple C classes Semantic Segmentation task, model performance is compared with the mean IoU of all classes. Equations 2.11 and 2.12 illustrate both IoU and mIoU, respectively. In equation 2.11, TP_c denotes the number of class c pixels that were correctly classified (True Positives), FN_c denotes the number of class c pixels that were classified as belonging to another class (False Negatives) and FP_c denotes the number of pixels of other classes that were incorrectly classified as belonging to class c (False Positives).

Figure 2.1: Semantic Segmentation ground truth mask taken from the CityScapes Dataset. Each pixel is annotated as either class "road", "car", "person", etc.

$$IoU(c) = \frac{TP_c}{TP_c + FN_c + FP_c} \tag{2.11}$$

$$mIoU = \sum_{c=1}^{C} \frac{IoU(c)}{C} \tag{2.12}$$

In the following chapter, an detailed review of many state of the art methods that use these concepts will be presented.

Chapter 3

State of the Art

This chapter provides a detailed overview of the research that is currently being conducted on topics that are relevant for the problem of autonomous driving in adverse weather conditions. Firstly, a survey of popular available image datasets for the task of semantic segmentation in the context of autonomous driving is presented, comparing the characteristics and differences of each of these datasets. Both datasets of images of clear and adverse weather conditions are reviewed, as well as datasets that comprise data beyond 2D RGB images. Secondly, a review of the most popular and promising uncertainty estimation methods is presented. Classical and more modern uncertainty estimation techniques are compared, analyzing both advantages and disadvantages of each of them. Finally, a summary of the ongoing research on semantic segmentation is provided, with a focus on autonomous driving scenarios under adverse weather conditions. Thus, this chapter adopts the following structure:

- **Autonomous Driving Datasets for Semantic Segmentation**
- **Uncertainty Estimation**
- **Semantic Segmentation**

3.1 Autonomous Driving Datasets for Semantic Segmentation

There are several autonomous driving datasets for the task of semantic segmentation available and open for usage to the general public. Datasets such as Cityscapes [24] and KITTI [33] have been widely used as benchmarks for semantic segmentation methods and autonomous driving research in general, as they contain a number of real images with corresponding high quality pixel-level semantic annotations of several different classes. In particular, KITTI also incorporates data collected from a Lidar sensor that allows for 3D object detection tasks and has been updated and extended throughout the years. Some examples of these extensions include Virtual-KITTI [30], which clones 5 videos from the original KITTI dataset and synthetically augments them by changing weather conditions and camera settings, and SemanticKITTI [9],

which annotates all sequences of the original KITTI Odometry benchmark and adds annotations for the complete 360 degree field of view of the original Lidar sensor.

In order to achieve robust and accurate scene understanding, data from multiple different sensors can be fused to exploit their complementary properties [29]. As a result, datasets such as nuScenes [14], which contains data from 6 different cameras, 5 radars and 1 Lidar sensor in 1000 different 20 seconds scenes, all with full 360 degree field of view, Waymo Open Dataset [100] which provides 360 degree data of 5 Lidar sensors and 5 cameras across 1950 different 20 second scenes, ApolloScapes [48] which comprises data collected from 2 Lidar sensors and 6 cameras across 73 different scenes, A2D2 [34] which offers data from 5 Lidar sensors and 6 different cameras, all with 360 degree view, and Oxford Robotcar [66] which consists in 100 recordings of the same route over a year across different weather conditions and time of day, captured by 4 cameras and 3 Lidar sensors, have become very popular benchmarks for deep multi-modal perception methods.

While there is an increasing interest and demand in multi-modal datasets, they remain very expensive to build. Thus, simpler datasets of 2D images and videos and their respective annotations are still useful for semantic segmentation tasks and subtasks, such as object detection and instance segmentation. The previously mentioned Cityscapes dataset [24] remains one of the most popular datasets related to Autonomous Driving, mainly due to its size (5000 images with high quality pixel level annotations and 20000 coarsely annotated images) and its suitability for different segmentation tasks. It has been updated and extended, now including a synthetically generated foggy version of all its original images [91] and 3475 images with 3D bounding box annotations for 3D object detection [38]. Mapillary Vistas [75] is another dataset comprised of 25000 images with fine pixel-level annotations. Unlike other datasets, it was built using images captured with different imaging devices, across different parts of the world and different weather conditions. BDD-100K [115] provides a total of 100 000 driving videos, each 40 seconds long and finely annotated. These videos were captured in the United States across different weather conditions. Camvid [12] is a smaller dataset of around 10 minutes of video footage recorded in Cambridge, England. It features around 700 pixel level semantic segmentation annotations.

To achieve full driving autonomy, self driving cars must be ready to deal with clear and adverse weather conditions. Most of these popular datasets, such as CityScapes and KITTI, mainly include images captured in clear weather conditions. Although BDD-100K, Mapillary Vistas and Oxford Robotcar do include images captured in adverse weather conditions, they aren't specialized for semantic segmentation tasks and thus don't provide many annotations. Datasets such as Dark Zurich [92], ACDC [93] and CADC [85] all focus in autonomous driving under adverse weather conditions, although the latter is a multi-modal dataset and, to this date, does not support 2D semantic segmentation. Dark Zurich features around 2500 images of night time and around 3000 images of twilight time urban driving scenarios, but only provides pixel-level semantic annotations for 151 night time images. ACDC is the more complete dataset, as it contains 1000 pixel-level annotated images for each of the following adverse conditions: Rain, Fog, Night and Snow. It also features the same classes in the semantic segmentation task

as CityScapes, which makes it very simple to use both datasets interchangeably.

3.2 Uncertainty Estimation

There are several approaches to Uncertainty Estimation. Monte Carlo Dropout [31] is an extremely popular Uncertainty Estimation method in NNs. Inspired by the dropout regularization technique introduced by [97] to prevent overfitting, MC Dropout approximates Bayesian inference and estimates model uncertainty by adding dropout during training and testing. Multiple forward passes of the same input are performed, in each of which some neurons will be "turned off" according to a certain probability (the neurons are sampled by a Bernoulli distribution). This will lead to a distribution of different outputs, where its entropy or variance corresponds to the model's uncertainty. An illustration of this method can be seen in Figure 3.1(d).

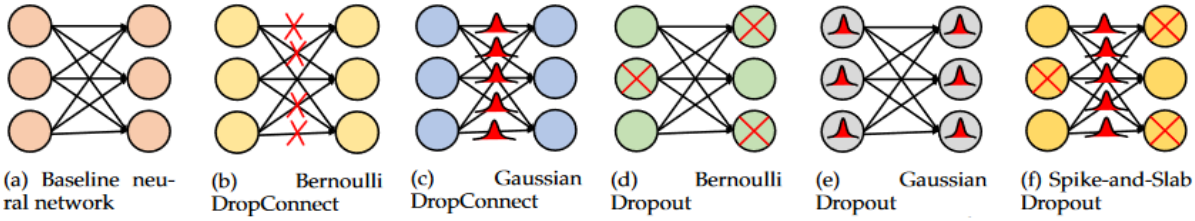


Figure 3.1: A visual representation of different regularization techniques that can be used to originate MC Dropout and MC DropConnect variants. Adapted from [5].

MC Dropout has been applied in several areas and has been extensively studied, with many variants emerging. MC DropConnect [73] has been proposed as an alternative to MC Dropout. It uses DropConnect [108], a generalization of dropout that instead of turning off an entire neuron only turns some of its weights (connections) off, leaving it partially active. A visual representation of this method is presented in Figure 3.1(b). The authors performed experiments on a semantic segmentation task across three datasets: Camvid, CityScapes and CT-Organ, with the latter being composed by 226 abdominal 3D CT scans not available to the public. A different semantic segmentation NN was used for each of these datasets, namely SegNet [8], ENet [82] and VNet [71]. Dropout and DropConnect layers were fit in the same parts of these networks. It was found that MC DropConnect outperformed MC Dropout both in model accuracy and uncertainty estimation. Bayesian SegNet [53] was proposed as an extension of SegNet that adds three dropout layers in its encoder and another three dropout layers in its decoder in order to estimate pixel-wise uncertainty in semantic segmentation. Bayesian SegNet is trained and tested using MC Dropout, with its pixel-wise class and uncertainty predictions being the mean and variance of the all the pixel-wise class predictions of the MC samples, respectively. A representation of Bayesian SegNet can be seen in Figure 3.2.

In [112], regularization techniques such as Gaussian Dropout (represented in Figure 3.1(e)) and Gaussian DropConnect (represented in Figure 3.1(c)) are proposed as a way to speed up the training of a NN using Bernoulli Dropout or Bernoulli Dropconnect, respectively. Since in the

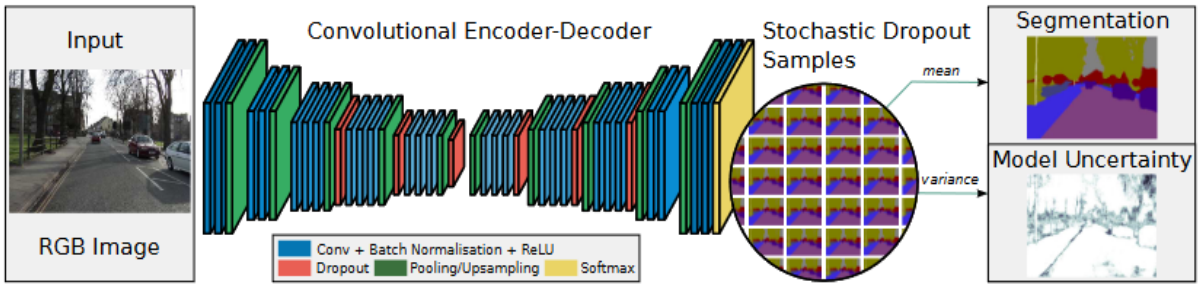


Figure 3.2: BayesianSegNet, image from the authors [53].

latter techniques some neurons or weights will be turned off during the forward pass, they won't be updated in the corresponding backwards pass, making the training process slower. Instead of turning off these neurons or weights according to a Bernoulli distribution, the Gaussian variants of dropout and dropconnect multiply the neurons or weights of a NN by a random variable with a Gaussian distribution, which leaves them available to be updated and trained. In [70] several experiments with Monte Carlo sampling are performed using all the mentioned regularization techniques on both FCNs and CNNs. It was found that, regardless of the regularization technique, using Monte Carlo sampling during training improved the NN's ability to represent its own uncertainty. It was also found that using both Gaussian DropConnect and Bernoulli Dropout led to the best performing and calibrated NNs. This technique is called Spike and Slab Dropout and is represented in Figure 3.1(f).

Several works related to autonomous driving use MC Dropout to estimate uncertainty. [84] studies the effect of MC Dropout in estimating uncertainty in a semantic segmentation task on a synthetic dataset [55] of road scenes under adverse weather conditions. The authors experiment with a DeepLabv3+ [20] semantic segmentation NN with a ResNet-50 backbone [41] altered to include four dropout layers. Two of these models are trained on two different sets of clear weather images where one is larger than the other. Their performance is then evaluated on adverse weather condition images and it was found that the model trained on more data generally had less epistemic uncertainty. It was also found that by increasing the intensity of the adverse weather condition the uncertainty of the models also increased. [6] uses MC Dropout with both Bernoulli Dropout and 2D Spatial Dropout [104] to estimate model uncertainty in a BNN for autonomous vehicle control. PURE [16] is a method that uses MC Dropout to measure uncertainty for object detection NNs in datasets such as KITTI and BDD-100k. [121] proposes a simplified version of the BayesianSegNet with only two dropout layers in both its encoder and its decoder, also adding pyramid pooling modules to the network. This method was tested on CityScapes, obtaining a better performance than the original Bayesian SegNet.

Although MC Dropout is seen as a simple and effective way to estimate uncertainty, the additional computational work involved in the multiple forward passes of the same input can be prohibitively expensive and thus make this method too slow for deployment in areas where fast decision making is needed, such as autonomous driving. Some works have tried to tackle the problem of obtaining MC Dropout's performance without the additional computational cost. [11]

proposes a single shot approach in which the expected value and the variance of the predicted output are estimated through moment propagation on the NN, successfully approximating MC Dropout's performance in a regression setting. [36] adopts a teacher-student methodology to distill knowledge from the teacher model trained with MC Dropout to the student model, calibrating its uncertainty predictions. In this way, MC sampling is avoided during test time. Inspired by this work, [95] also adopts a teacher-student paradigm. The difference is that the predictive distribution produced by the teacher model with MC Dropout is approximated by parametric distributions, of which the student model learns to predict their parameters. Therefore, in test time, MC sampling is not used and the networks predictions and uncertainty estimates come from the student NN.

Ensembles are a widely used method for modelling and predicting uncertainty [32]. Initially introduced to improve prediction performance, [59] was the first work to use Deep Ensembles for predictive uncertainty estimation. The authors independently train NN models with random initialization of their parameters on the same dataset which, for each NN model, will produce a predictive distribution $P(\hat{y}|x, \theta_m)$. These are then combined to yield the Ensemble predictive distribution as $P(\hat{y}|x) = \frac{1}{M}\sum_{m=1}^{M} P(\hat{y}|x, \theta_m)$. A number of experiments covering both classification and regression was performed, where the Ensemble's performance and uncertainty estimation was compared to MC Dropout using NLL and the Brier Score. It was found that the proposed Ensemble method significantly outperformed MC Dropout and, more interestingly, when both methods were tested on OOD data, MC Dropout predicted much higher confidence than the Ensemble method even when the latter used a low amount of models (five). It was also found that increasing the number of models in the Ensemble generally led to better uncertainty estimates. [79] provides an extensive comparison between Ensembles, MC Dropout and other calibration and uncertainty estimation methods across different classification datasets. It was found that Ensembles generally outperformed all other methods and were more robust to distributional shift. Interestingly, the calibration methods generally provided better calibration metrics for small distributional shifts, but were still outperformed by Ensembles when this shift increased. [37] is another work that aims to compare the performance of Ensembles with that of MC Dropout. The authors experiment with both methods on several different tasks such as regression, classification, depth completion and semantic segmentation. For the semantic segmentation experiments, the Deeplabv3 [19] NN was used for both MC Dropout and Ensembles. These methods were tested on the CityScapes dataset, using its validation set for evaluation. Ensembles significantly outperformed MC Dropout in both performance metrics (mIoU) and calibration metrics (ECE). Ensembles also outperformed MC Dropout in the other tasks. [7] argues that the commonly used metrics to compare different uncertainty estimation methods should only be applied to models that produce well calibrated probabilities. They experiment with Ensembles of different NN architectures for classification tasks, finding that although most Ensembles tipically contain a great number of models, they are equivalent to using only a few.

All of these works use the same approach to build an Ensemble as in [59], where each model architecture is the same and their diversity is achieved by randomly initializing each model's weights in the training process. However, this may not be enough to achieve Ensemble

diversity which may in turn lead to poorer uncertainty estimates. AutoDEUQ [28] is an Ensemble approach to uncertainty estimation that allows for both aleatoric and epistemic uncertainty to be estimated. Unlike the traditional approach, AutoDEUQ builds the Ensemble by searching in both architecture and hyperparameter space, building a set of models. A heuristic is then used to choose the models for the Ensemble. The authors compare the performance of AutoDEUQ with that of other methods such as MC Dropout and the traditional Deep Ensembles approach in a regression setting, and find that AutoDEUQ performs better. [116] also tries to achieve Ensemble diversity through the usage of different NN architectures. The authors propose an algorithm that explores a NN architecture search space and chooses the ones that will be trained to build the Ensemble. It was found that this approach to achieve Ensemble diversity outperformed the less sophisticated random initialization of weights approach in classification experiments in both uncertainty estimation and robustness to distributional shift. These results may be justified by the greater diversity in predictions achieved by ensembles with different NN architectures, illustrated in Figure 3.3. [64] proposes a framework that is built on top of Ensemble approaches

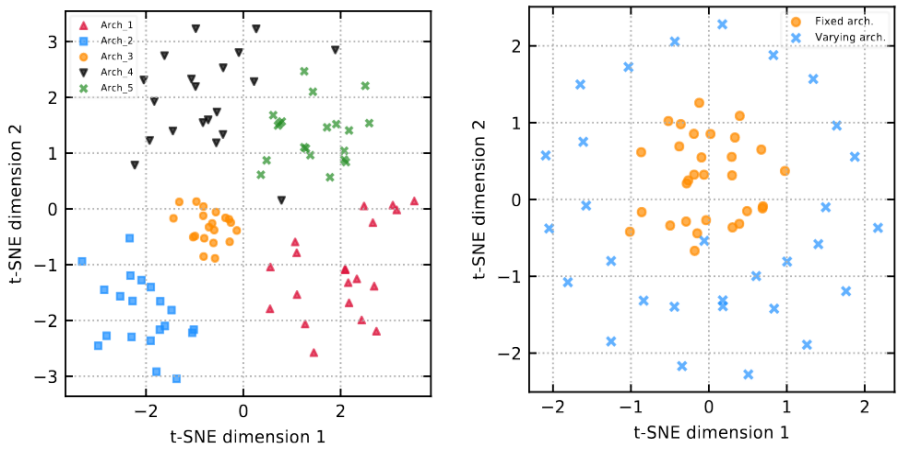


Figure 3.3: Clustering the variety of predictions of different NN architectures. In the left image, it can be seen how the architectures of NNs affect predictions since there is almost a direct correspondence between each cluster and a different NN architecture. In the right image, it can be seen that varying the architecture of the NNs that form an ensemble results in more diverse predictions. Image by the authors [116].

that allows for the decomposition of uncertainty into three categories: aleatoric, parametric and structural. Structural uncertainty is referred to by the authors as model misspecification, that is, the uncertainty that arises due to the choice of a method that doesn't correctly model the given data, whereas parametric uncertainty corresponds to the uncertainty associated with the model's parameters. To achieve this, the predictive distribution of the Ensemble is augmented by a constrained gaussian process and a residual gaussian process is added to the Ensemble's prediction. [25] explores the usage of an Ensemble to estimate uncertainty under domain shift on the task of semantic segmentation. The authors chose a DeepLabv3+ architecture and use different backbones and data augmentation techniques to achieve model diversity. The final Ensemble consists of five models and its performance is evaluated on three different datasets:

CityScapes, BDD-100k and GTA [89] which is a synthetically generated dataset. The models trained with data augmentation techniques performed better under domain shift, but performed worse than the models trained with no data augmentation when evaluated under no domain shift.

Ensembles have been used to estimate uncertainty in autonomous driving applications. [43], [44] [122] use this approach in Reinforcement Learning tasks that aim to build a decision making agent for vehicle control. [102] proposes a motion prediction model that incorporates an uncertainty estimation module based on a Deep Ensemble method.

Although Ensembles achieve state of the art uncertainty estimation performance, they still present many computational challenges that may hinder their use in real world applications like autonomous driving. They generally perform better than MC Dropout and can be much faster, as each model on the Ensemble can be run in parallel, however that may not be enough. Some approaches have been proposed to solve this problem, such as Deep Sub-Ensembles [106], which suggests using only the last layers of the models to build the Ensemble. A single model then aggregates these layers as its final layers, acting as the Ensemble. Experiments are made in classification and OOD detection tasks, with Deep Sub-Ensembles slightly losing in performance to Deep Ensembles but being considerably faster. Knowledge distillation approaches can also be adopted, for example [68] explores the effects of Ensemble distillation onto a single model. The authors performed experiments on classification tasks and found that the single model retained classification performance as well as uncertainty estimation performance.

Most deep learning works involving Markov Chain Monte Carlo use this method for the training of NNs, but MCMC is a sampling method that has also been used for uncertainty estimation. The goal of MCMC is to build a Markov Chain whose stationary distribution converges to a distribution of interest, in this case, $P(w|D)$. [61] uses Stochastic Gradient MCMC to incorporate uncertainty in the training process of a NN, boosting its predictive performance. In SG-MCMC, the model weights represent the states in the Markov Chain, with each training iteration requiring that the gradient is estimated on a mini-batch. This gradient is then used to transition the state in the Markov Chain. There are multiple options for the transition rule, such as using Hamiltonian dynamics [21] or Langevin dynamics [113]. To approximate Bayesian inference, Monte Carlo integration is used by sampling T weights w_i from the stationary distribution of the Markov Chain, producing predictions in the following way $\mathbb{E}_{P(w|D)}[P(\hat{y}|x,w)] \simeq \frac{1}{T}\sum_{i=0}^{T} P(\hat{y}|x,w_i)$. In this way, predictive uncertainty can also be quantified, although these works only focus in the predictive performance of the model. [42] combines Dropout with SG-HMCMC and makes a qualitative comparison of the predictive uncertainty estimated by different MCMC methods in a classification setting across different datasets. It was found that this combination of Dropout and SG-HMCMC generally outperformed SG-LMCMC and SG-MCMC with no dropout not only in predictive accuracy, but also presented lower confidence in similar classes whereas the other methods were overconfident. Unfortunately, MCMC methods face many challenges, such as computational cost, the choice of the prior distribution to be optimized by the Markov Chain and convergence issues [80].

Variational inference is another method that has been used to approximate the posterior $P(w|D)$ [32]. Using a family of parametric distributions $Q(w)$, variational inference finds the parameters that minimizes the difference between $Q(w)$ and the posterior. This difference between distributions is measured by the Kullback-Leibler Divergence in 3.1.

$$KL(q||P) = \mathbb{E}_q \left[\log \frac{Q(w)}{P(w|D)} \right] \tag{3.1}$$

Since this equation explicitly uses the posterior, it cannot be directly minimized. Thus, the Evidence Lower Bound (3.2) is maximized instead, since this is equivalent to minimizing the KL divergence.

$$ELBO = \mathbb{E}_q \left[\log \frac{P(y|x,w)}{Q(w)} \right] \tag{3.2}$$

The KL divergence can then be obtained by $KL(q||P) = -ELBO + \log P(y|x)$ [32]. One very popular variational inference technique is Bayes By Backprop [10]. The authors use the Gaussian reparameterization trick [78] to approximate the ELBO as

$$F(D,\theta) \simeq \sum_{i=1}^{n} \log Q(w_i|\theta) - \log P(w_i) - \log P(D|w_i) \tag{3.3}$$

where θ represents the parameters of the variational posterior, which is chosen to be a diagonal Gaussian distribution with mean μ and standard deviation ρ. For the prior distribution $P(w)$ the authors suggest using a scale mixture of two Gaussians. The transform that yields a sample of weights w is defined as $w = \mu + \log(1 + \exp(\rho)) \cdot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$. In each optimization step, a sample of weights is drawn according to this transform and these are then used in the NN to compute the gradients with the usual backpropagation algorithm. Parameters μ and ρ are then updated according to these gradients. Experiments are then conducted in reinforcement learning, classification and regression settings. In the classification problem, Bayes By Backprop achieved comparable performance to that of Dropout. In [86], a similar approach is proposed but the parameters for the variational posterior and the prior are specified layer-wise, which leads to some alterations. Thus, weight uncertainty is specified layer by layer and less variational parameters are introduced. A qualitative evaluation of predictive uncertainty estimation by this approach was performed in the MNIST [60] dataset, with the network almost always correctly predicting examples with low uncertainty. [15] proposes the usage of Gaussian Processes as both the prior and variational posterior family of distributions. The authors focus on computer vision tasks, such as semantic segmentation, arguing that related works using variational inference rely on sampling for the ELBO gradients estimation, which is not viable in semantic segmentation due to its high-dimensionality of inputs and outputs. The covariance kernel of the prior Gaussian Process is constructed using $K(x_i, x_j) = f(x_i)^T f(x_j) dp(f(x_i), f(x_j))$ as its covariance function. For these two images x_i and x_j, the covariance kernel would be $\begin{pmatrix} K(x_i,x_i) & K(x_i,x_j) \\ K(x_i,x_j) & K(x_j,x_j) \end{pmatrix}$. This means that in the kernel, the diagonals contain information about the variance and covariance of the pixels in the same image and in different images. The covariance kernel of the Gaussian

Process of the variational posterior is similar, making usage of feature mappings of each image and pixel correlations among different images. The authors experiment with semantic segmentation on the CamVid dataset, comparing the proposed method with MC Dropout. It was found that this method outperformed MC Dropout in IoU, accuracy and mean calibration, as seen in Figure 3.4.

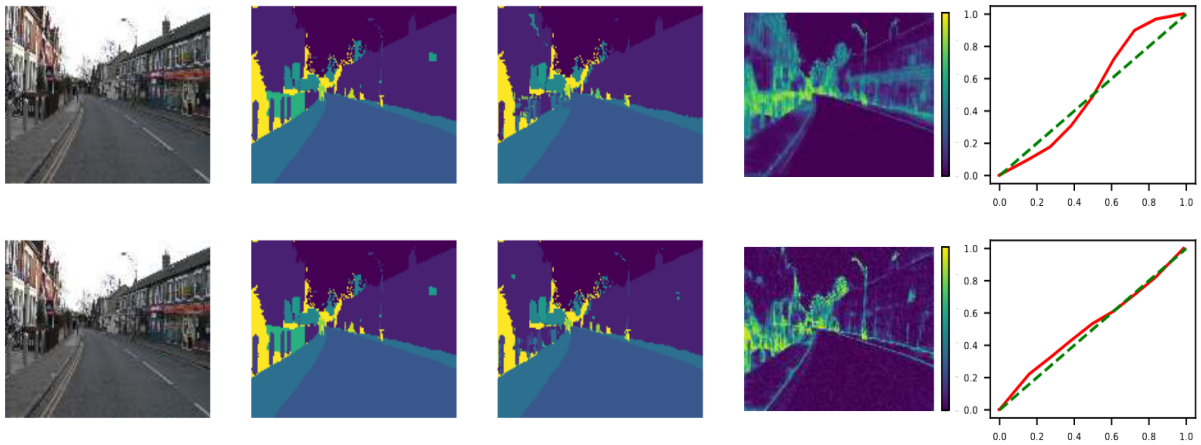


Figure 3.4: Comparison between MC Dropout (top) and the proposed method (bottom). From left to right: rgb input, ground truth, prediction, entropy of the predictions, calibration plot. Image by the authors [15].

[118] shows that the updates of the parameters of the variational posterior can be approximated by a variant of the usual gradient descent algorithms that learn the weights of a NN by introducing adaptive weight noise, allowing for the usage of slightly altered optimization algorithms, such as a noisy version of Adam [56], for the training of variational posteriors. The authors then use this method to perform variational inference with a matrix-variate gaussian posterior, finding that their method outperformed methods such as Bayes By Backprop in regression settings. In [26], the authors introduce tensor Normal distributions over the convolutional kernels of a NN, as these can accurately model correlations and variances along all dimensions [69]. To approximate the means and covariances of the tensor Normal distributions, they are propagated through the network. This is illustrated in Figure 3.5. These tensor Normal distributions are hence used as the variational posterior, which will then yield the predictive uncertainty as their covariance. The authors then perform experiments on a computer vision classification task, comparing the accuracy obtained by their method with the one obtained by Bayes By Backprop and MC Dropout. They find that their method significantly outperforms its counterparts, with the difference being larger when the number of adversarial examples is increased. The concept of Normalizing Flows was introduced to variational inference in [88]. In this work, the authors argue that independent Gaussians and other mean-field approximations aren't flexible enough to approximate the true posterior. Normalizing flows are a transformation of a probability density through a sequence of invertible mappings and so can be viewed as a hierarchical probability distribution, which the authors argue is better suited to approximate the true posterior of the weights of a NN.

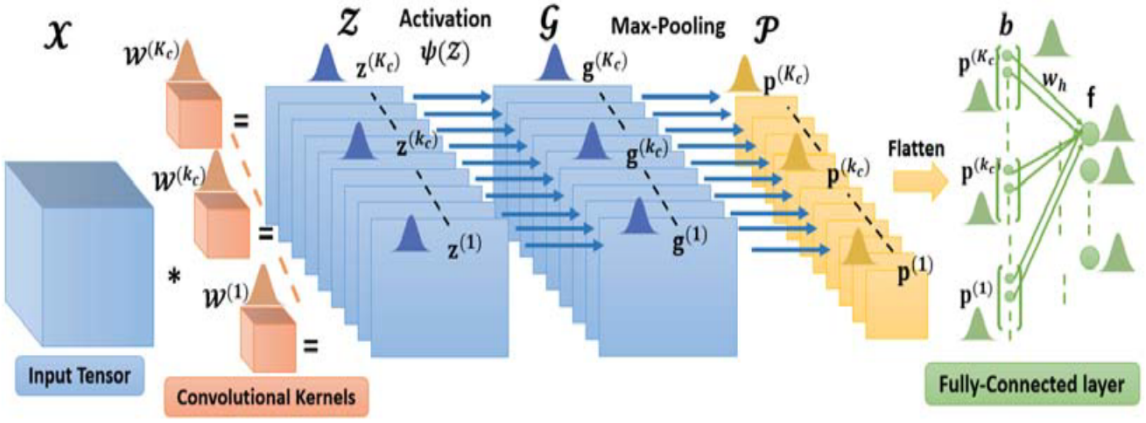


Figure 3.5: An example of the propagation of the mean and covariance of the variational posterior across the layers of a CNN. Image by the authors [26].

Up until this point, only uncertainty quantification methods based on Bayesian inference have been reviewed. Although these types of methods are the most common, other methodologies exist. In [51], the authors propose a training framework called DEUP where two models are present: the main model which is trained for the original task and a secondary model trained to predict the epistemic uncertainty of the main model in out-of-sample inputs. DEUP is presented in two different learning settings. A fixed dataset setting, where a portion of the original training dataset is held out as out-of-sample data and used as input to the main model (previously trained on the other part of the data) to compute its out-of-sample error and use it to train the secondary epistemic uncertainty estimator model. There's also an interactive setting, where new samples are generated in each training iteration, the uncertainty of the main model in these new samples is computed and is used as training data for the secondary model. To generate these new samples, a different distribution from the distribution of the original dataset can be used, in which case the generated samples are out-of-distribution. The authors tested DEUP in Sequential Model Learning, Reinforcement Learning and OOD Detection tasks. DEUP performed better than other Uncertainty Estimation techniques such as MC Dropout and Deep Ensembles in the OOD Detection task, which consisted in training a ResNet-18 model in the CIFAR-10 [58] dataset and use the estimated prediction uncertainty to reject OOD inputs. [81] argues that a model's predictive uncertainty should be proportional to the density of the input, that is, a model should produce higher uncertainty if the input is improbable according to the training data and lower uncertainty otherwise. To achieve this, the authors introduce an auxiliary energy-based model of the input $E(x)$, that will serve to define the density uncertainty criterion $\text{Var}_{q(\theta)}[f(x,\theta)] \propto E(x)$. This criterion is then integrated into the ELBO objective function and this will be the optimization objective. The energy model is decomposed into a Gaussian energy model for each layer, which allows for the density uncertainty criterion to be imposed layer-wise. Lastly, the authors define a density uncertainty layer, that is designed to satisfy the density uncertainty criterion and to add noise to its predictive variance, which promotes diversity of predictions. Experiments are performed in regression, classification and

OOD detection tasks, where this method outperforms variational inference and MC Dropout. [65] proposes an algorithm called Spectral-normalized Neural Gaussian Process (SNGP) that improves the uncertainty estimation of a model by adding spectral normalization to the model's layers and replacing the dense output layer with a distance-aware Gaussian Process. The authors define the concept of input distance-awareness as the existence of a summary statistic $u(x)$ that summarizes model uncertainty and reflects the distance between input x to the training data. To achieve this criterion, the output layer of the model (the softmax layer g that maps the hidden space representation of the input $h(x)$ to the label space) must be made distance aware, that is, g must output uncertainties that are compatible with the previously mentioned distance and the hidden mapping $h(x)$ must be distance preserving, so that the distance between the input and the training data is not affected by the network. A Gaussian process with a shift invariant kernel replaces the ouput layer to make it distance aware, and spectral normalization is used in the hidden mapping to make it distance-preserving. Several experiments across different tasks and datasets are conducted, with SNGP's performance being compared to that of other methods. Figure 3.6 compares the uncertainty estimates of different methods on the two ovals and the two moons datasets, generated by two gaussian distributions and by two moon-shaped distributions, respectively. Both datasets include OOD points represented in red. Other experiments suggest

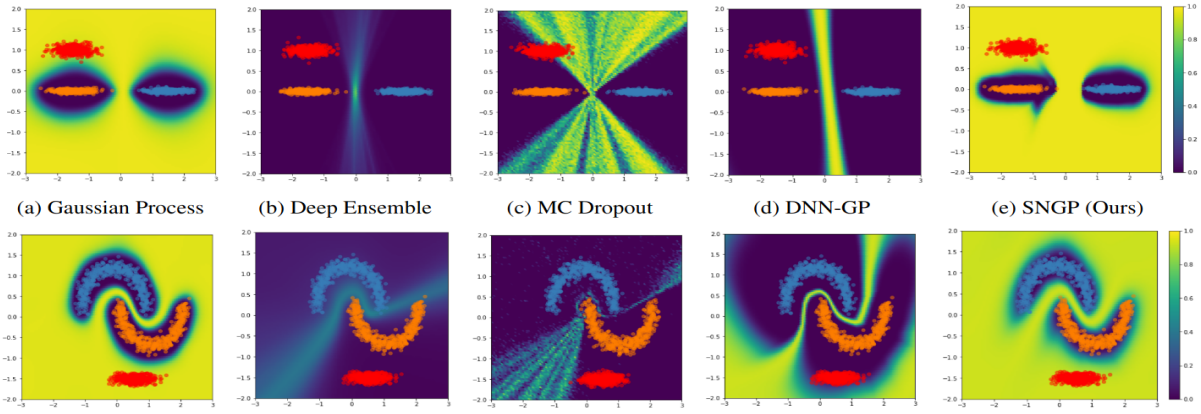


Figure 3.6: Uncertainty estimates of different methods on the two ovals (top row) and two moons (bottom row) datasets. Image by the authors [65].

that SNGP has similar uncertainty estimation performance to that of multi-model techniques such as Deep Ensembles and MC Dropout and significantly outperforms other single-model methods. [23] proposes MC Dropweak, a combination of MC Dropout and MC Dropconnect that both turns off entire hidden units and weights. The authors apply this method to lung cancer detection (a dataset that consisted of CT scans) and found that it increased model performance.

3.3 Semantic Segmentation

Semantic segmentation allows for scene understanding and is thus fundamental for autonomous driving. Using semantic segmentation, an autonomous vehicle is able to locate and segment areas of interest, such as roads and sidewalks, which is crucial for safe autonomous driving. Although simple to understand, semantic segmentation is a challenging task in computer vision. Many more difficulties arise when dealing with adverse weather conditions such as rain, as this can impact the quality of the video and images that are recorded by a camera and data captured by sensors, subsequently impacting the quality of the segmentation. Thus, a semantic segmentation solution that is robust to all weather conditions is crucial for autonomous driving. Deep learning has emerged in the literature as a promising approach to semantic segmentation [40], hence this section will focus on deep learning based methods. An overview of general semantic segmentation methods will be presented and then more attention will be given to methods that deal with adverse weather conditions.

The more popular semantic segmentation methods typically employ an encoder-decoder structure where the encoder is responsible to learn a down-scaled latent space representation of the input using extracted features by a CNN. The decoder then learns to transform this latent space representation to the output segmentation mask, usually by upsampling. VGG [96] is a popular CNN that has been used in semantic segmentation methods. The most famous variant, VGG-16, is composed by thirteen convolutional layers and five max-pooling layers. ResNet [41] introduced residual connections to deep CNNs. The output of a layer is used as input to the following layers, but a residual connection is added so that the original input is preserved. This alleviates the vanishing gradient problem and allows for more stable training. DenseNet [47] is a CNN that uses the output feature maps of all preceding layers as inputs to the next layers. This has the advantage of alleviating the vanishing gradient problem and promoting a more efficient use of features. An illustration of dense and residual blocks is presented in Figure 3.7. When compared to ResNet, DenseNet has been shown to outperform it even with less parameters, although requiring more computational costs [117]. MobileNet [46][94][45] is a family of lightweight CNNs designed to be efficient that can be deployed in real time applications. Their architectures include the usage of depthwise separable convolutions which enables parallelization and attention mechanisms. Xception [22] optimizes the Inception [101] modules by reformulating them as depthwise separable convolutions, performing better than Inception on image classification experiments.

One of the most popular semantic segmentation methods is the Pyramid Scene Parsing Network [119]. This method works by extracting an image feature map with a CNN which is then used as input to a pyramid pooling module. This pyramid pooling module is composed of four pooling kernels of different sizes: 1x1,2x2,3x3 and 6x6 whose output is then followed by a 1x1 convolution to reduce their dimension. Each of these outputs are then upsampled and concatenated before going trough another convolution that produces the final prediction. An illustration of this method is given in Figure 3.8. The authors propose the usage of ResNet-50 to

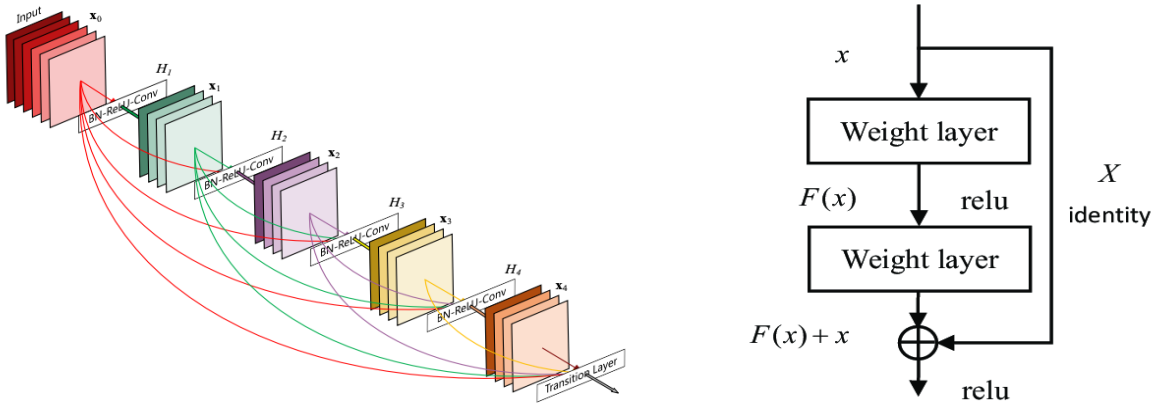


Figure 3.7: A representation of the dense block (left) and a residual block (right). Adapted from [4] and [47].

extract feature maps from the images, and this is the CNN that is used in their experiments. It was found that PSPNet outperformed other methods in datasets such as Pascal VOC 2012 and CityScapes, particularly in classes that are harder to capture because of their small size, such as *bike* and *dog*. This suggests that the pyramid pooling module is an effective way to capture context and increase the accuracy on small objects.

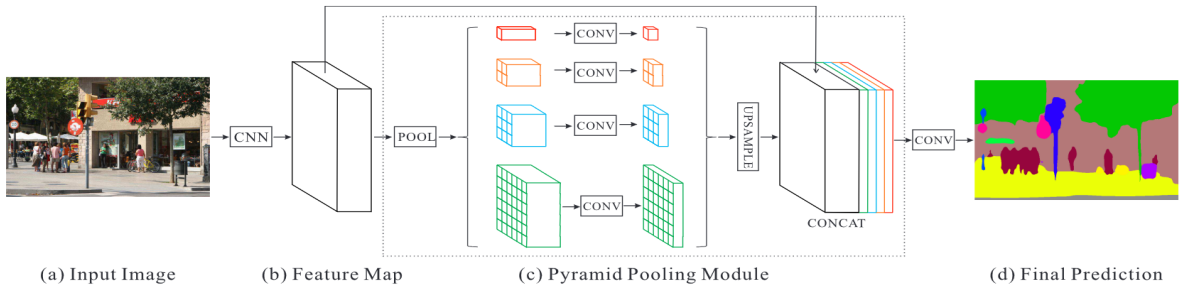


Figure 3.8: The architecture of PSPNet. Image by the authors [119].

Deeplab [17] is a family of models that uses depthwise atrous separable convolutions, atrous spatial pyramid pooling and, in its third iteration, an encoder-decoder architecture that uses Xception as the backbone of its encoder. SegNet [8] follows a simple encoder-decoder architecture to perform semantic segmentation. Its encoder is composed by the VGG-16 backbone, having both its convolutional and pooling layers. The decoder presents convolutional and upsampling layers, where upsampling is performed based on the pooling indices of the encoder. DeconvNet [76] uses an architecture composed by a CNN followed by a deconvolutional neural network. Analogous to an encoder-decoder architecture, the CNN is responsible of downsizing the original image, while the deconvolutional neural network produces the final segmentation map by upsizing the output of the CNN. This deconvolutional neural network possesses unpooling and deconvolutional layers. An unpooling layer performs the inverse of a pooling layer, restoring the size of the pooled feature map. This is achieved by storing the pooled locations in switch variables and using them to

unpool the input. A deconvolutional layer, unlike a convolutional layer, does not compress a kernel-sized region of the input feature map to a single activation and instead performs the opposite, that is, takes a single feature/activation as input and decompresses it, enlarging it to a kernel-sized window.

Unlike the previously presented works, Segmenter [99] is a transformer-based method for semantic segmentation. Originally introduced in [107] for natural language processing, the transformer architecture has been extended to computer vision tasks in [27], which introduced a pure transformer approach named ViT to image classification tasks. The authors found that dividing images into small 16x16 patches and feeding them to a transformer allowed it to globally process the image, without the need for convolutions. Segmenter extends ViT for semantic segmentation tasks using a purely transformer-based approach. Segmenter presents an encoder-decoder architecture where a transformer encodes image patches that are then fed to a mask transformer to decode and produce the final prediction. An illustration of this method can be seen in Figure 3.9. It was found that Segmenter achieved the same performance as DeepLab on the CityScapes validation set.

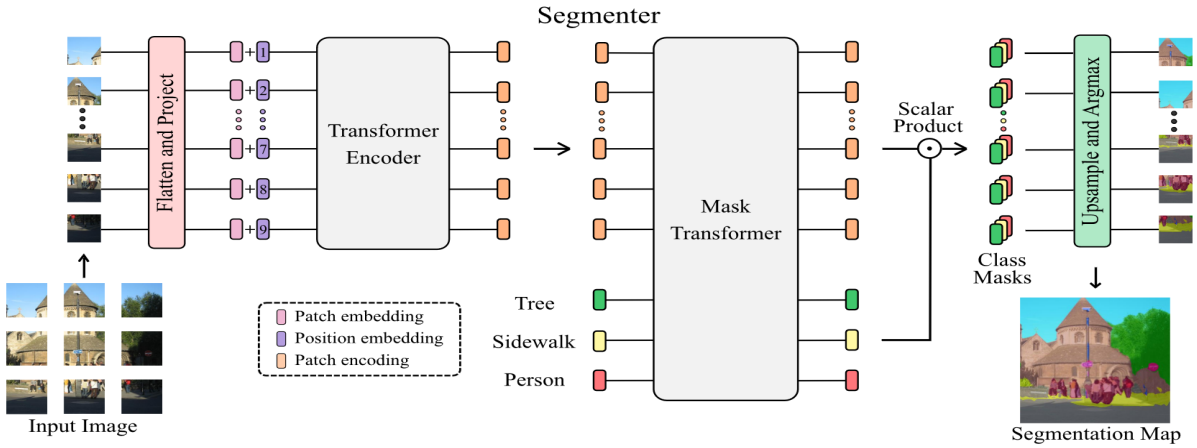


Figure 3.9: The architecture of Segmenter, a transformer-based approach to semantic segmentation. Image by the authors [99].

Although these approaches achieve extremely good performance, their expensive computational costs hinder their use in real world applications [72]. In autonomous driving, where the environment can change very fast, the usage of slow semantic segmentation methods is unfeasible. This motivates the development of faster and lighter methods that can be used in real time. ENet [82] is an early work that presented a real time semantic segmentation method. The authors adopt an encoder-decoder architecture where the encoder is much larger than the decoder, as the authors argue that the decoder should operate on small resolution data and only upsample the output of the encoder. Large frames of data are expensive to process and consequently early pooling of data is implemented. Inspired by the double-branching idea of ResNet, the convolutional blocks of the encoder consist of two branches: one that max-pools and pads the input and another that performs 1x1 projections to reduce dimensionality, followed by a convolution, another 1x1 projection and finally a regularization. The output of these branches

is then added and used as input to the next block. ENet performs well on CityScapes, achieving slightly lower performance than SegNet while being much smaller. BiSeNet [114] is another real time semantic segmentation method. Its architecture presents two main branches: a spatial path, where the input goes through three convolutional layers with a small stride that preserves spatial information and yields detailed feature maps, and a context path, which consists of a small model (the authors use Xception) followed by global average pooling and an attention module to efficiently downsample the input and yield context-rich feature maps. These two branches run simultaneously and are independent of one another, with their output being combined through a feature fusion module. When tested on the CityScapes dataset, BiSeNet achieves slightly lower mIoU performance than other real time semantic segmentation methods while being much faster. However, BiSeNet outperforms the other methods in mIoU when using larger models in the context path, but this considerably lowers its speed. ICNet [120] is a similar approach to BiSeNet. ICNet's architecture consists in three different branches whose output is combined in a cascade-like way. Each branch operates with different resizings of the original input, one branch uses the original input, the second branch uses a resized input to half of the original size and the third branch uses an input that is a quarter of the original size. The main idea is to use a heavier CNN (the authors use PSPNet) on the lower resolution input to produce a coarse segmentation map that is subsequently combined with the output of the other branches through a cascade feature fusion module, refining it and producing the final prediction. The heavier network is used on the lower resolution input because it saves inference time, while the other branches use lighter weight CNNs as they work with higher resolution inputs. An illustration of this method can be seen in Figure 3.10. DFANet [62] further explores the combination of feature

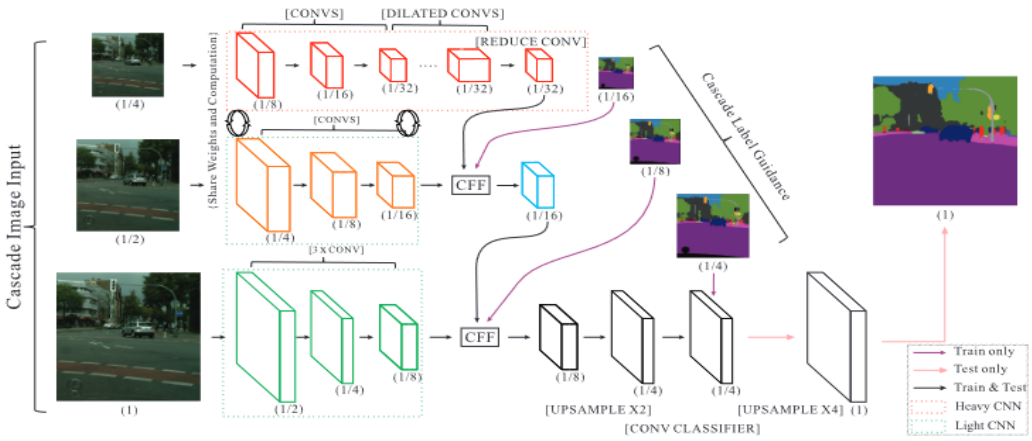


Figure 3.10: The architecture of ICNet, image by the authors [120].

maps at different layers of a CNN to produce a segmentation map. The proposed architecture is an encoder-decoder one, where the main encoder has three encoding blocks. The output of each of these encoding blocks is then encoded separately, concatenating the output of several encoding blocks in the process and passing them through a final attention layer. The decoder receives the three encodings and combines them with the output of the attention layer to produce final prediction. Experiments on the CityScapes dataset suggest that DFANet is slightly faster than

BiSeNet but achieves slightly lower mIoU on the test set. DFANet outperforms ICNet both in inference time and mIoU performance.

Until now, only generic semantic segmentation methods have been reviewed. These methods perform very well on clear weather images where there is good illumination, but unfortunately their performance drops when dealing with adverse weather conditions as this represents a large domain shift. As such, semantic segmentation methods that deal with different weather conditions are extremely important. In [54], an approach to deal with adverse weather conditions in semantic segmentation through multi-task learning is proposed. The Deeplabv3+ [20] architecture is augmented through two small convolutional modules, TAS and WAS, that are trained to predict day or night time and the weather conditions, respectively. These modules are used during the training process to force the model to learn time of day and weather condition specific features. For the training of this method, a combination of the CityScapes and AWSS (a synthetically-generated adverse weather conditions dataset built by the authors) is used, and its performance is evaluated on ACDC and CityScapes validation sets. Under the same training datasets, this methodology obtains better mIoU results than the regular Deeplabv3+ and PSPNet methods. AdapNet [105] is an approach to semantic segmentation that combines data from different sources (modalities) to enhance its predictions instead of only relying on 2D RGB images, as the authors argue that such reliance hinders performance under adverse weather conditions. Each modality is processed through an independent network that outputs a segmentation map. All produced feature maps are then combined together using a weighted fusion. For example, if a RGB 2D image and a corresponding depth map are used as input, they will be processed by different networks, with their outputs fused. If the image is of bad quality, more weight will be given to the segmentation map produced by the network that processed the depth map. MFNet [39] also uses more than just 2D RGB images to perform semantic segmentation. Unlike AdapNet, only thermal RGB counterparts of the 2D RGB images are considered and processed. Its architecture consists of two encoders, where one receives the 2D RGB image and the other receives the thermal RGB counterpart. The output of both encoders is then fused in a decoder that produces the final prediction. SFNet-N [109] focuses on semantic segmentation of low-light images such as night time. This is a two step method. The input first goes through a CNN responsible for light enhancement, which increases the quality of low light images, and its output is then fed to a semantic segmentation encoder-decoder model to obtain the final prediction. The architecture of this model is very complex, processing the input in two different branches and sizes, combining different feature maps at several different layers through a feature alignment module and using attention layers. In their experiments, the authors compare SFNet-N with unsupervised domain adaptation methods, which are outperformed in mIoU. [63] follows a similar approach to SFNet-N, also incorporating both an image enhancement network and a segmentation network. In contrast to SFNet-N, this image enhancement network contains an image enhancement module that produces a clear image given the input adverse weather image, but this is not used as a direct input to the segmentation network. Instead, the segmentation network receives the original input and produces a segmentation map which is fused with the clear image from the image enhancement network. This fusion is then used as input to an edge prediction network that

produces an edge map. Posteriorly, the clear image, the segmentation map and the edge map pass through convolutional layers to produce the final prediction. This is done in the training process only, as the segmentation network is the only predictor when testing. This method is illustrated in Figure 3.11.

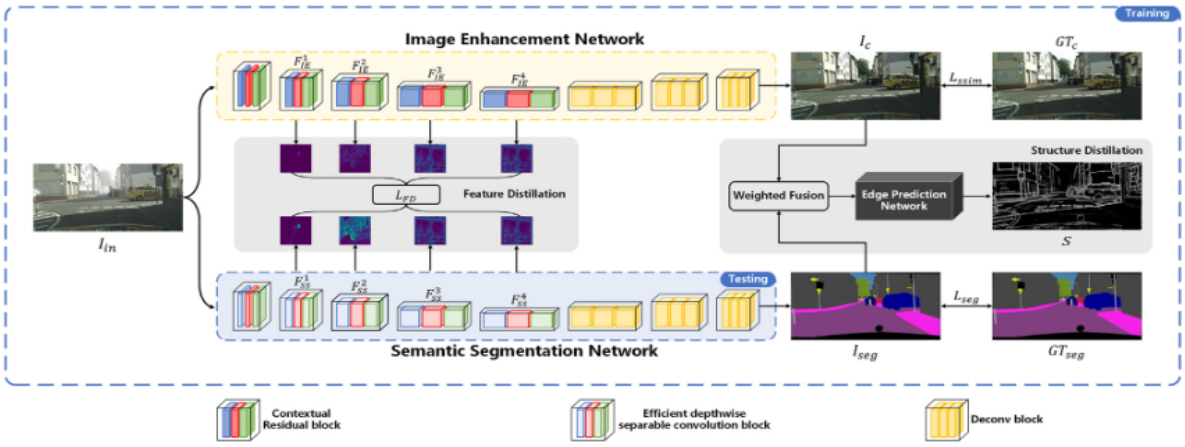


Figure 3.11: An illustration of the method in [63]. Image by the authors.

[52] applies a contrastive learning methodology to train a real time semantic segmentation network, SwiftNet [110], to obtain reliable segmentation maps from images under adverse weather conditions. The authors make SwiftNet output two feature maps, one that is directly included in an image-level contrastive loss and another that is upsampled to a segmentation map that is then incorporated in a pixel-level contrastive loss. The training process tries to minimize both of these losses.

A wide variety of methods that try to solve the semantic segmentation task have thus been proposed, each with their own advantages and disadvantages. Heavier methods tend to achieve the best performance but can't be deployed in real time applications. Methods that are specifically designed to operate in real time usually reduce their computational weight by downsizing the input or removing layers from their architecture, achieving comparable performance to heavier methods but being considerably lighter. To deal with adverse weather conditions, the main methods proposed in the literature usually involve some kind of fusion between different data modalities, such as sensors [83]. Indeed, methods that perform multimodal fusion significantly outperform unimodal methods since they have access to more information and, under adverse weather conditions, the quality of the RGB image is degraded. For autonomous driving, real time semantic segmentation methods that perform multimodal fusion may provide a robust solution for road scene perception.

In the next chapter, a simple method that estimates the uncertainty on the training samples and uses this information to improve its performance in semantic segmentation will be presented, thus bringing topics of all the fields described in this chapter together.

Chapter 4

Design and Development

In the previous chapter, several works were reviewed across three major areas: uncertainty estimation, semantic segmentation and adverse weather conditions. Hence, the idea for this work was to develop a method that acted on these three areas combined. This chapter will then cover the steps and decisions that were taken to design a methodology that uses uncertainty information to boost model performance in semantic segmentation under adverse weather conditions. First, a general overview of this methodology will be presented, along with its implementation and detailed explanation of all training parameters used. This is followed by an in-depth analysis of the adopted CNN architecture, the reasoning for this choice, and the tools and materials used on its implementation. Also, a review of two datasets used for experiments (one dataset comprised of clear weather images and another of adverse weather images) will be presented. Thus, this chapter adopts the following structure:

- **Boosting Model Performance with Uncertainty**
- **Deeplab**
- **Dataset Benchmarks**

4.1 Boosting Model Performance with Uncertainty

The performance of a model can be improved using uncertainty information by encompassing the uncertainty related with a training sample in the loss function, making the training of the model uncertainty-aware. As these uncertainty-aware training procedures typically yield better calibrated models and boost their performance, an uncertainty-aware training approach was considered in this work. Semantic segmentation is the classification of each pixel in an image to one of C classes, and as such can be viewed as the classification of $H \times W$ instances, resulting in $H \times W$ predictions each associated with its own predictive uncertainty. Hence, a 2-dimensional matrix of size $H \times W$ where each element is the predictive uncertainty of the corresponding pixel in the image can be used in the training process. Generating these matrices, dubbed uncertainty

masks, can be achieved by an uncertainty estimation method. Since ensemble methods have been shown to estimate good quality uncertainty values, to increase model performance and are simple to implement, an ensemble approach was used to estimate the predictive uncertainty associated with each pixel of each sample on the training set, thus generating their corresponding uncertainty masks. These masks are then used in the training process of a new model by increasing the weight of "hard to predict" pixels in the loss function, thereby encouraging the model to learn them better. An illustration of this method can be seen in Figure 4.1. The

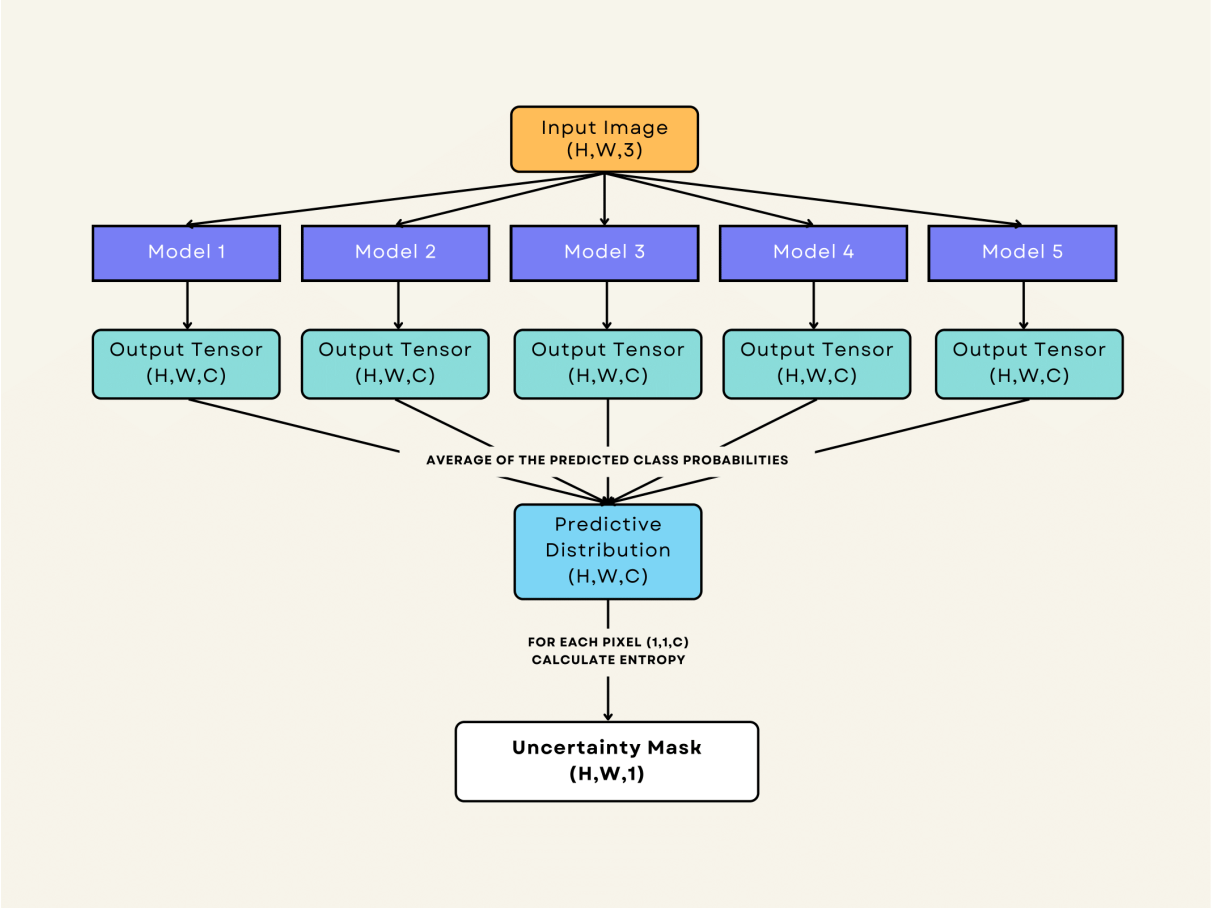


Figure 4.1: An illustration of the suggested method to generate uncertainty masks.

ensemble is comprised of five models, as this is not a huge amount of models and is usually enough to provide good quality uncertainty estimations. It should be noted that the output of each of these models is not a segmentation mask but instead is the output of their softmax layer, a tensor of dimensions (H, W, C), where C is the number of classes. Thus, each element (h, w, c) in this tensor represents the predicted confidence that pixel (h, w) belongs to class c. These tensor outputs of each model m are then combined and averaged, yielding the ensemble confidence predictions tensor as $\frac{1}{5}\sum_{m=0}^{5}(H, W, C)_m$. This corresponds to computing an approximation of $\mathbb{E}_{P(w|D)}[P(\hat{y}|x, w)]$ in equation (2.2). To obtain the pixel-wise predictive uncertainty, the entropy of the predictive distribution of each pixel is computed, generating the final uncertainty masks. After the uncertainty masks are generated, they are inserted into the training process of a model in a very simple fashion. After the forward pass of a training sample, the corresponding

uncertainty mask is loaded. The cross entropy loss between the output of the model and the ground truth is computed, yielding a (H,W) tensor where each element represents the pixel-wise loss value. This loss tensor is then multiplied by the uncertainty mask, effectively increasing the loss value of pixels with higher uncertainty and decreasing the loss value of pixels with less uncertainty. This is illustrated in Figure 4.2.

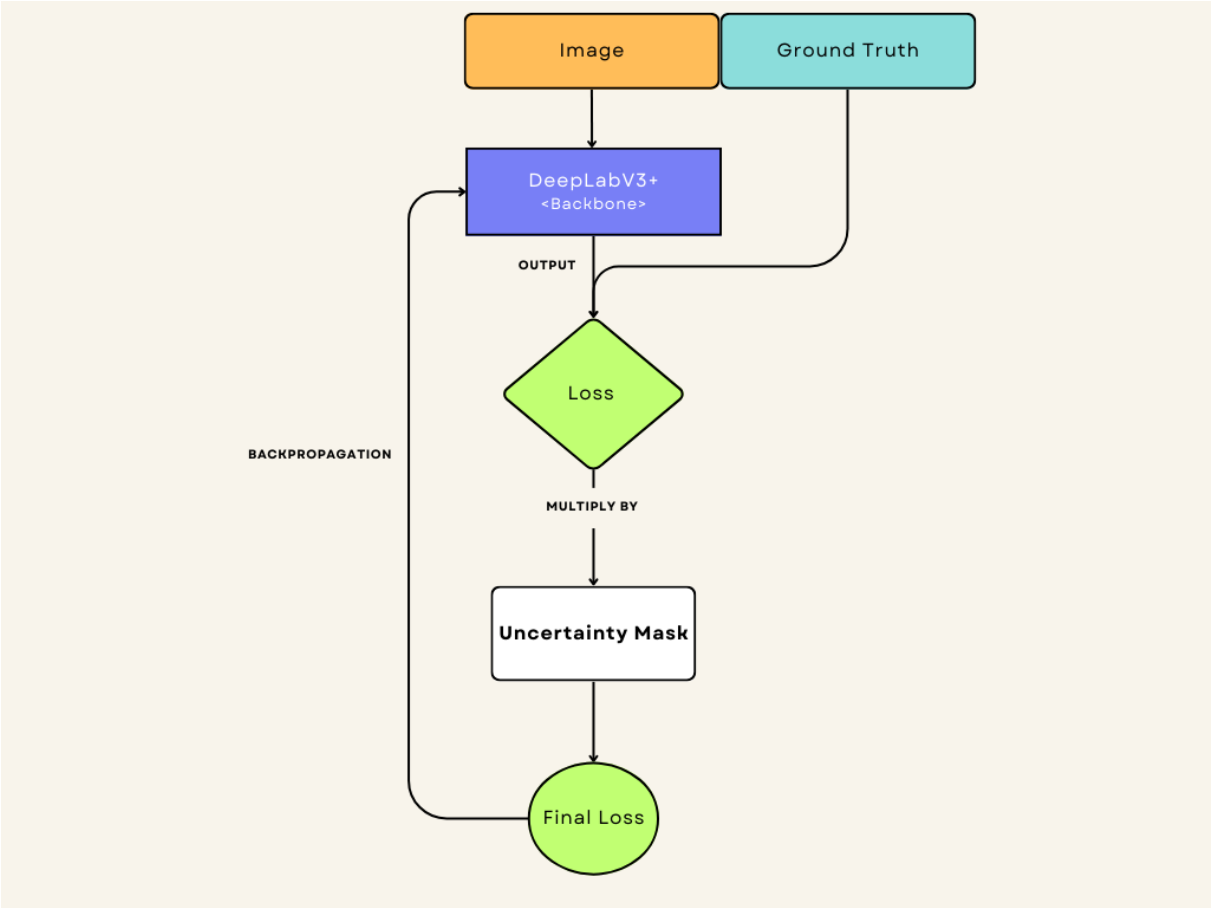


Figure 4.2: An illustration of the usage of uncertainty masks in the training process.

The implementation of this method was based on the github in [1]. This github comprises a solid implementation of the DeepLabV3+ model in Pytorch [3] with a variety of different backbone architectures, such as ResNet, Xception, MobileNet and HRNet [111]. Hence, this github was used and extended to implement the described method. To ensemble the models and produce their combined predictions, a wrapper class presented in listing 4.1 was implemented.

To generate the uncertainty masks according to Figure 4.1, the function in listing 4.2 was implemented. Each sample of the dataset is fed as input to the ensemble model, which outputs the predictive distribution tensor. The entropy of each (1,1,C) pixel is then computed, yielding the final uncertainty mask.

Finally, to multiply the loss values by the uncertainty mask of the sample, a loss class was implemented as defined in listing 4.3. Unlike what is shown in Figure 4.2, the final loss is obtained

```
class DeepEnsemble(nn.Module):
    def __init__(self,models):
        super().__init__()
        self.models = models

    def forward(self,x):
        outputs = [model(x) for model in self.models]
        outputs = [F.softmax(output,dim=1) for output in outputs]
        out = torch.mean(torch.stack(outputs), dim=0)
        return out
```

Listing 4.1: The DeepEnsemble class that wraps the 5 models and produces their combined predictions. It receives the trained models as input and outputs the mean of their predictions.

```
def make_uncertainty_masks(opts, model, loader, device, metrics,
    ret_samples_ids=None):
    with torch.no_grad():
        for i, (images,labels) in tqdm(enumerate(loader)):
            labels, uncertainty_masks = labels[0], labels[1]

            img_name = images[1][0].split('/')[-1][:-4]
            images = images[0].to(device, dtype=torch.float32)
            labels = labels.to(device, dtype=torch.long)

            probs = model(images)
            log_probs = torch.log(probs+1e-10)
            entropy = -torch.sum(probs * log_probs, dim=1)

            uncertainty = entropy.detach().cpu().numpy()
            np.save(f'datasets/data/ACDC/uncertainty_masks/
                {img_name}_uncertainty_mask',uncertainty)
```

Listing 4.2: The make_uncertainty_masks function that generates uncertainty masks. It receives the trained models as input and iterates through the dataset loader to generate its corresponding uncertainty masks.

through standard matrix multiplication, as empirical tests suggested that this was the best option. In this way, the loss value of a pixel is not only multiplied by the single corresponding uncertainty value but also by the uncertainty values of the pixels in the same column. With this implementation the uncertainty masks are loaded by the dataset loader before the training begins, avoiding the loading of these masks during the loss computation.

Since the goal of this work is to analyze if this method increases the performance of a model and the focus is not on obtaining the best possible performance on a dataset, the DeepLabV3+ semantic segmentation model was chosen to conduct all the performed experiments, due to its popularity, flexibility and available documentation and code. An in-depth review of the DeepLab family of models will be presented in the next section.

```
class UncertaintyAwareCrossEntropyLoss(nn.Module):
    def __init__(self,ignore_index=255,reduction='none'):
        super(UncertaintyAwareCrossEntropyLoss, self).__init__()
        self.ignore_index = ignore_index
        self.reduction = reduction

    def forward(self, inputs, targets, uncertainty_masks):
        ce_loss = F.cross_entropy(inputs,targets,reduction='none',
        ignore_index=self.ignore_index)
        uncertainty_aware_cross_entropy_loss =
            torch.matmul(ce_loss,uncertainty_masks)
        return uncertainty_aware_cross_entropy_loss.mean()
```

Listing 4.3: The UncertaintyAwareCrossEntropyLoss class that encapsulates the weighting of the usual cross entropy loss by an uncertainty mask.

4.2 DeepLab

As mentioned in the previous section, the DeepLab family of models was chosen to test the presented method. The DeepLab architectures are extremely popular and achieve state of the art performance across several different datasets. Thus, in the following subsections, the DeepLab family will be presented.

4.2.1 DeepLabv1

The DeepLabv1 system [17] is a pixel-level semantic segmentation method that combines a Convolutional Neural Network and a Conditional Random Field. This system was proposed to tackle the two main problems of applying Convolutional Neural Networks to pixel-level semantic segmentation tasks: signal down-sampling and spatial invariance. Signal down-sampling leads to the loss of information present in the image by consecutively applying pooling and convolutional filters. Spatial invariance is a desirable property for Convolutional Neural Networks in classification tasks, allowing for the correct classification of the image regardless of spatial transformations. However, this leads the network to be insensitive to spatial details which is not desirable for pixel-level semantic segmentation. To solve the first problem DeepLabv1 employs atrous convolutions, which enlarge the network's field of view and allow for the incorporation of larger context at the same computational cost of a standard convolution. Unlike standard convolutions, atrous convolutions have an extra parameter called the atrous rate which corresponds to a spacing between elements of the kernel. This is illustrated in Figure 4.3.

Conditional Random Fields are DeepLabv1's solution to the second problem. The coarse score maps produced by the convolutional neural network are upsampled using bilinear interpolation and then used as input to a fully connected Conditional Random Field that predicts the segmentation labels. This scheme is presented in Figure 4.4.

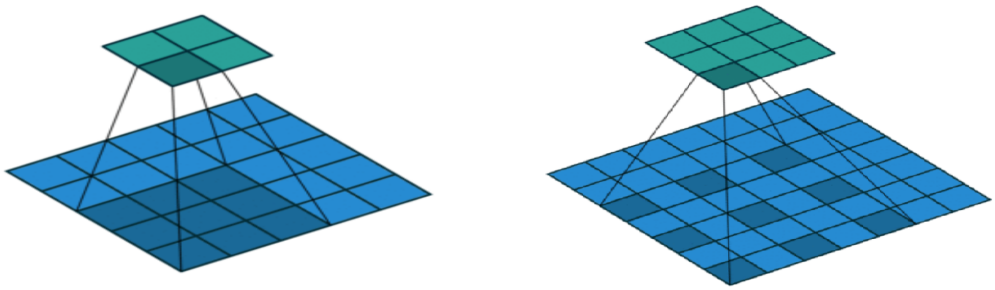


Figure 4.3: Difference between a standard convolution (left) and an atrous convolution (right). Both convolutions use a 3x3 kernel, with the atrous convolution using an atrous rate of 2. Image adapted from [2].

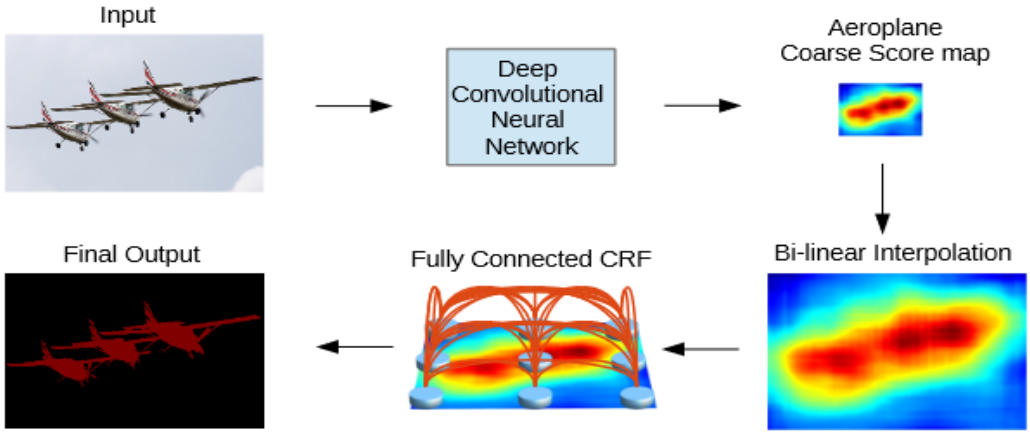


Figure 4.4: Illustration of the Deeplabv1 method

All DeepLabv1 experiments were conducted using a VGG-16 derived fully convolutional neural network adapted to the semantic segmentation task. This network was suited with atrous convolutions and different atrous rates were tested to evaluate the impact of increasing the network's field of view on its performance on the PASCAL VOC 2012 dataset. The best results were obtained by using a 7x7 kernel on the first convolutional layer of the network with an atrous rate of 4. However, the same performance was achieved by using a 3x3 kernel with a larger atrous rate of 12, with this being a much lighter model. The authors also experiment with multiscale prediction by adding a two layer MLP to the first four max pooling layers of the convolutional neural network. The feature maps produced by these MLPs are then concatenated to the main network's feature maps and fed to the softmax layer. This methodology improved DeepLabv1's performance by only 1.3% mIoU.

4.2.2 DeepLabv2

Deeplabv2 [18] follows the same general methodology of its predecessor presented in Figure 4.4. Instead of only using a VGG-16 convolutional neural network, experiments were conducted

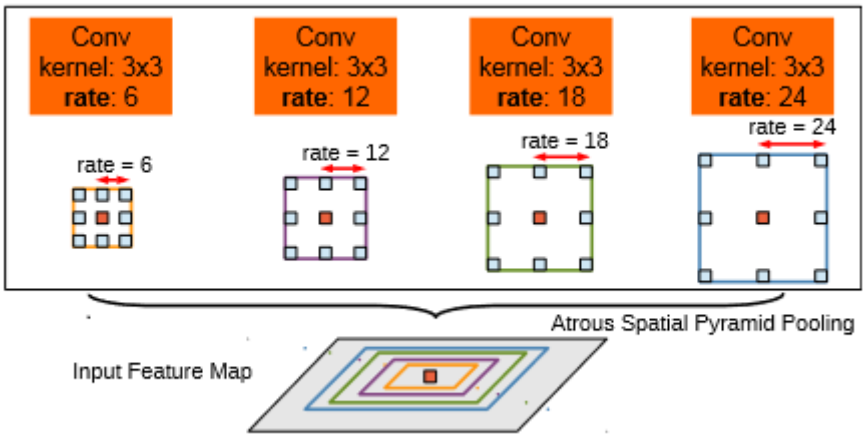


Figure 4.5: The Atrous Spatial Pyramid Pooling Module. To classify the center pixel, atrous filters with different atrous rates are considered.

with a more complex ResNet-101 which achieved better performance. The major change was in the multi scale processing of the images, which was previously done by using the intermediate feature maps of the network as input to the softmax layer. The authors introduce Atrous Spatial Pyramid Pooling, a pooling module that exploits several multi scale features to classify a pixel. Figure 4.5 illustrates this idea.

This module employs parallel atrous convolutions with different atrous rates, processing the feature map obtained by each convolution in individual branches and fusing them together in the end. In the VGG-16 and ResNet-101 networks, this module is attached to the final pooling and convolutional layers, respectively. Experiments were conducted across different datasets, such as PASCAL VOC 2012 and CityScapes. The DeepLab variant that used ResNet-101 always outperformed the VGG variant, and the ASPP module was found to increase DeepLab's performance independently of the convolutional neural network used.

4.2.3 DeepLabv3

In its third iteration, some major changes were made to the DeepLab system [19]. The ASPP module was altered to exclude the branch with an atrous rate of 24, now only employing three different atrous rates (6,12,18) when using an output stride of 16. When this value is changed to 8, these rates are doubled to compensate for the larger feature map resolution. Output stride is the ratio between the original input resolution and the final post-convolution feature map resolution. For example, an output stride of 16 means that the final feature map resolution is 16 times smaller than the original image resolution. Another change to the ASPP module was the addition of an average pooling layer that allows for global context information to be considered by the model, as larger atrous rates make the center weight of the filter considerably more important than the others.
This new ASPP module is inserted in the fourth convolutional block of the modified ResNet-101

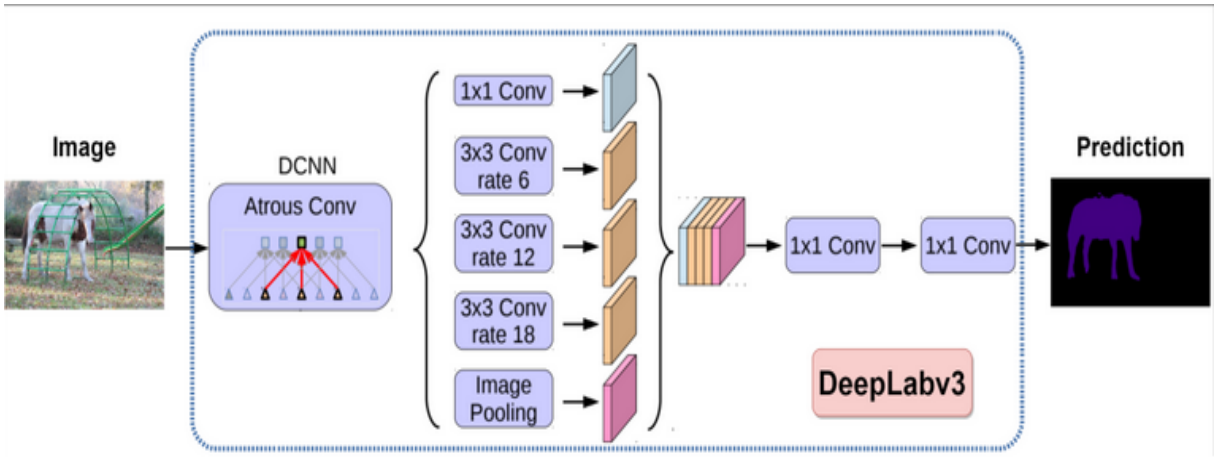


Figure 4.6: General scheme for the architecture of Deeplabv3 with an output stride of 16. The input image is fed to a DCNN with atrous convolutions (ResNet-101) and then passes through the ASPP module. Finally, the produced logits are bilinearly upsampled and the final prediction is produced.

used in DeepLabv2. The output of each of the individual branches of this ASPP module are then concatenated, passed through another 1x1 convolutional layer and then fed to the final 1x1 convolutional layer that produces the logits. An illustration of this scheme is presented in Figure 4.6.

Unlike its previous iteration, Deeplabv3 no longer employs a CRF to refine the score maps and produce the final prediction, as it was found that it had no effect in performance and was computationally expensive. The authors found that using a lower output stride increased accuracy but also increased training and inference time. Ultimately, Deeplabv3 is a significant improvement from its previous iterations as it achieved better results and is computationally faster.

4.2.4 DeepLabv3+

DeepLabv3+ is an extension to DeepLabv3 that adds a decoder module and an Xception based backbone that implements atrous depthwise separable convolutions [20]. This updated method uses DeepLabv3 as to encode rich semantic information and uses the decoder to refine segmentation results. This is illustrated in Figure 4.7.

A depthwise separable convolution is a composition of a depthwise convolution followed by a pointwise convolution that combines its output. A depthwise convolution is a spatial convolution performed over each individual channel of an input, while a pointwise convolution is performed across all channels of an input. Figure 4.8 illustrates these concepts.

The authors modify the Xception network by adding more layers and replacing all max pooling layers with atrous depthwise separable convolutional layers to extract features at several

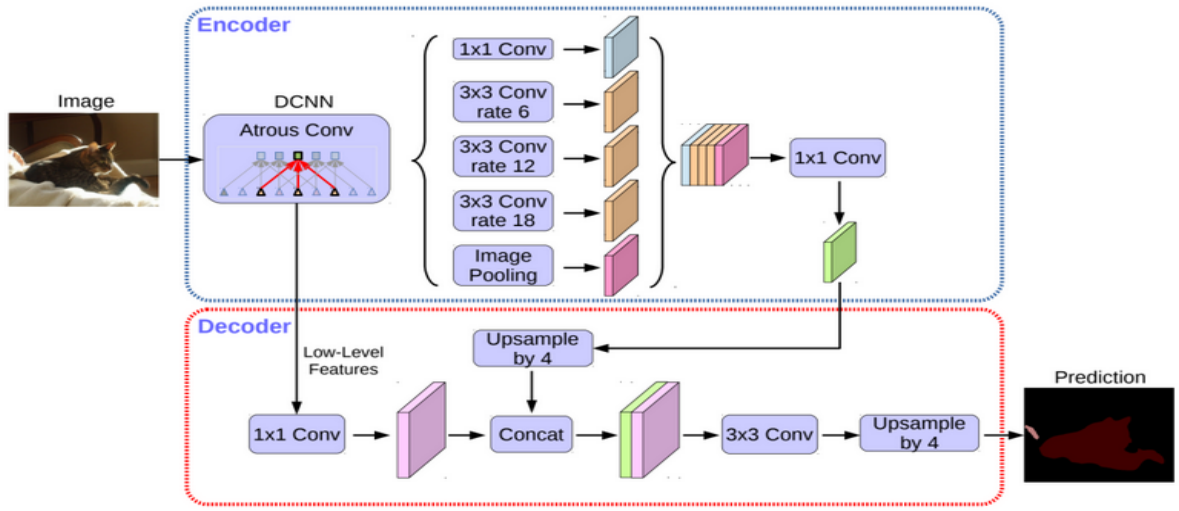


Figure 4.7: The architecture of DeepLabv3+. Here, the DCNN used is the modified Xception with atrous depthwise separable convolutions. The input image is fed to this network, which then passes the output feature maps to the ASPP module and to the decoder. This decoder thus produces the final prediction.

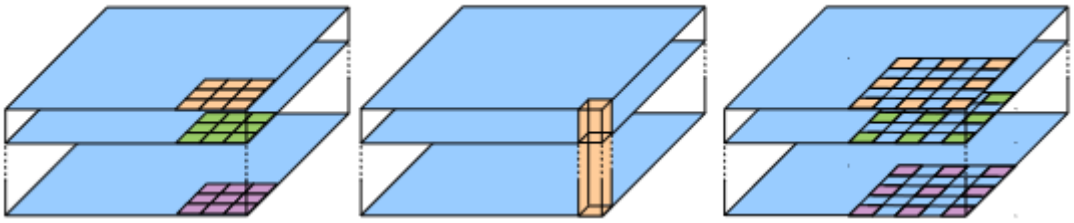


Figure 4.8: Depthwise convolution (left) is applied over each individual channel of the input, while a pointwise convolution (middle) is applied across all channels of the input. An example of an atrous depthwise convolution can be seen on the right.

different resolutions. The decoder module, as illustrated in Figure 4.7 is very simple. The features obtained by the modified Xception backbone are fed into it, passing through a 1x1 convolution to reduce their channels. This is then concatenated to the output of the encoder previously upsampled by the output stride (in Figure 4.7 an output stride of 4 was used). A final 3x3 regular convolution is applied and an upsampling by the output stride is employed to produce the final prediction.

Although DeepLabv3+ can be used with different backbones other than the modified Xception, such as the ResNet-101 used in the previous iterations, in the author's experiments DeepLabv3+ achieves the best performance using the modified Xception backbone. At the time it was published, Deeplabv3+ set a new state of the art in Cityscapes and PASCAL VOC 2012.

4.3 Dataset Benchmarks

Two datasets were used to conduct experiments and test the described method: CityScapes, a dataset of road scene images under clear weather and ACDC, also a dataset of road scene images, but under adverse weather conditions. The usage of both of these datasets allows the evaluation of the performance of the method under domain shift. Both of these datasets will be presented next.

4.3.1 The CityScapes Dataset

CityScapes [24] is an extremely popular dataset for semantic segmentation and other computer vision tasks. It is a very large dataset, being comprised of 5000 finely annotated images and another 20000 coarsely annotated images of road scenes. The training and validation sets amount to 3475 finely annotated images, with the test set comprising the remaining 1525 images. The coarse annotations and the annotations for the training and validation sets are public, but the annotations for the test set are not. These images were captured in several different german cities, across different months and seasons, although no images under adverse weather conditions were captured. This makes CityScapes a valuable benchmark for semantic segmentation in autonomous driving, as model performance on CityScapes can be viewed as a baseline in non adverse weather conditions for posterior comparison to performance in adverse weather conditions. Each pixel in the fine annotations belongs to one of 30 classes, although some of them are very scarce so only 19 classes are taken into account in the evaluation process. These classes are : *road, sidewalk, building, wall, fence, vegetation, terrain, car, train, truck, bus, bicycle, motorcycle, sky, pole, traffic sign, traffic light, person, rider* and *other* (these pixels are annotated by the number 255 and are not taken into account for training and evaluation purposes). A visualization of the proportion of pixels of each class in the training set can be viewed in 4.9. CityScapes has been

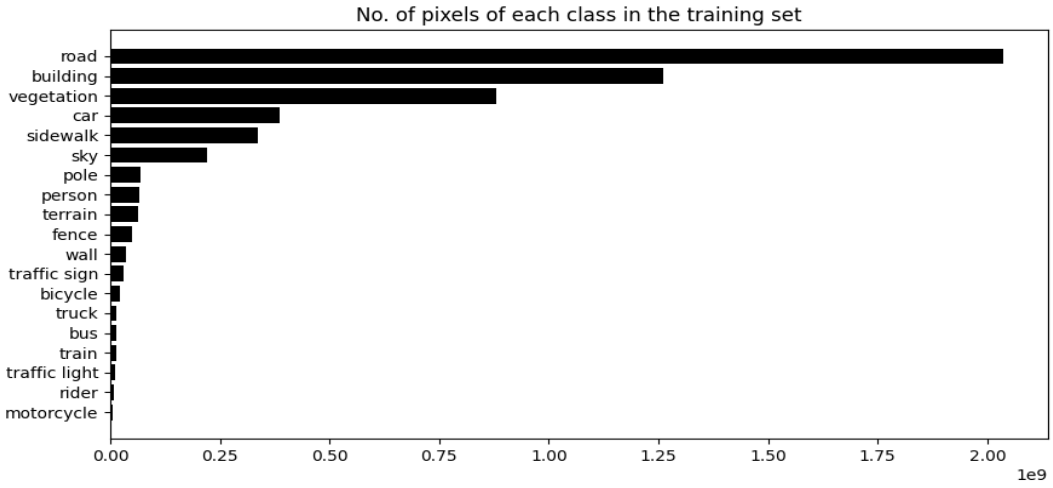


Figure 4.9: Number of pixels per class on the CityScapes training set

extended and augmented throughout the years, now supporting a variety of different tasks beyond traditional semantic segmentation, such as 3D object detection and panoptic segmentation. For the purpose of this work, only the finely annotated images were used for semantic segmentation.

4.3.2 The ACDC Dataset

The ACDC Dataset [93] is dataset designed for semantic segmentation comprised of 4006 pixel-wise annotated road scene images under four different adverse weather conditions: snow, rain, night and fog. ACDC is split into a training set of 1600 images, that is, 400 images per adverse weather condition, a validation set of 406 images (100 images per adverse weather condition except for night time, which corresponds to 106 images) and a test set of 2000 images. Annotations for both the training and validation sets are public, but not for the test set. ACDC also includes a clear weather image that corresponds to every adverse weather image, although sometimes the scene slightly changes and there are no annotations for the clear weather images. There is also an invalid mask for each image that contains information about which regions of the image are impossible for a human to segment/annotate, which were produced during the annotation process. ACDC's classes are the same as the classes in CityScapes, making both datasets compatible and easily usable together. Class balance is very important in machine learning, as complex models tend to overfit to majority classes while neglecting minority ones. In the ACDC dataset, the number of pixels of each class in the training set can be viewed in Figure 4.10. This Figure indicates that the dataset is highly unbalanced, with a huge difference in representation of classes such as *sky* and *bicycle*. Furthermore, it seems that classes that occupy a smaller portion of the image such as *rider* and *motorcycle* correspond to less pixels, increasing the difficulty of identifying these classes.

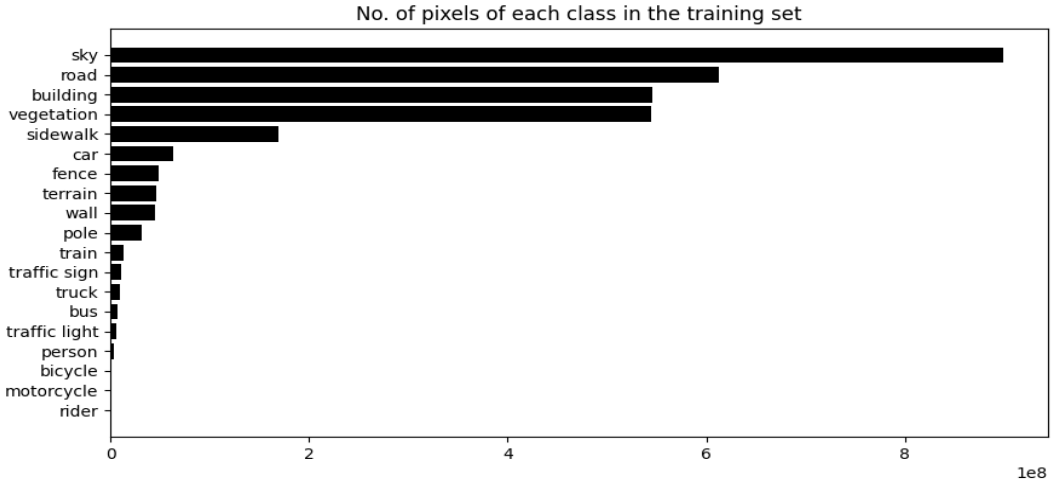


Figure 4.10: Number of pixels per class on the ACDC training set

In the next chapter, the experiments that were conducted to test the method and using tools here described will be presented, as well as a justification for the choice of performing them.

Chapter 5

Experiments and Tests

This chapter will be used to present several experiments that were performed to evaluate and analyze the uncertainty aware training method that was presented previously. Since datasets of images in adverse weather conditions are generally harder to work with than datasets of clear weather images, experiments 1 through 4 were conducted on the ACDC dataset while experiment 5 was conducted on both the ACDC and the CityScapes datasets. Experiments 1 and 2 compare the performance of the proposed method with its regular training counterpart, using an ensemble of DeepLabv3+ models with the same backbone and with different backbones, respectively. In experiment 3 the uncertainty predictions of the ensembles trained in the previous experiments are compared, both qualitatively and quantitatively. An analysis of the results that are obtained from retraining an ensemble from experiments 1 and 2 is performed in experiment 4. In experiment 5, ensembles are trained on the ACDC dataset to make uncertainty predictions on the CityScapes validation set, and vice versa. These uncertainty predictions on both validation sets are then compared. The results of these experiments are reported in the next chapter.

Before introducing the experiments, some important choices of training parameters and procedures will be reported and discussed. The first step in the training pipeline is to load the desired dataset. For both the CityScapes and ACDC datasets, a corresponding dataset class was implemented, which are used to convert the ground truth masks ids to the correct training ids, as these masks present the CityScapes label ids format. In Listing 5.1, the ___getitem___ method of the ACDC class is presented. This method is used to provide training samples to the training loop, as well uncertainty masks. The path of the image in the sample is also provided, as this is necessary when generating the uncertainty masks.

For both datasets, all images are resized to 768x768 pixels using bilinear interpolation and this remains the used size in all experiments. Smaller and larger image sizes were tried, but this size was chosen as it yielded the best results without being too large. The ground truth mask is also resized to 768x768 but instead using nearest neighbor interpolation. As is common practice, all images were normalized using the mean and standard deviation of the imagenet dataset. Listing 5.2 shows the transformations that the data goes through.

```
def __getitem__(self, index):
        image = Image.open(self.images[index]).convert('RGB')
        target = Image.open(self.targets[index])
        if self.transform:
            image, target = self.transform(image, target)
        target = self.encode_target(target)

        uncertainty_mask_name = self.images[index].split('\\')[-1][:-4]
        uncertainty_mask_path =
            f'uncertainty_masks\\{uncertainty_mask_name}_uncertainty_mask.npy'

        if os.path.exists(os.path.join(self.root,uncertainty_mask_path)):
            uncertainty_mask =
                np.load(os.path.join(self.root,uncertainty_mask_path))
        else:
            uncertainty_mask = []
        return (image,self.images[index]), (target,uncertainty_mask)
```

Listing 5.1: The ___getitem___ method of the ACDC dataset class.

```
   if opts.dataset == 'acdc':
        train_transform = et.ExtCompose([
            et.ExtResize(opts.crop_size),
            et.ExtToTensor(),
            et.ExtNormalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
                0.225]),])

        val_transform = et.ExtCompose([
            et.ExtResize(opts.crop_size),
            et.ExtToTensor(),
            et.ExtNormalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
                0.225]),])

        train_dst = ACDC(root=opts.data_root,split='train',
            transform=train_transform)
        val_dst = ACDC(root=opts.data_root,split='val', transform=val_transform)

    return train_dst, val_dst
```

Listing 5.2: The preprocessing of the images on the ACDC dataset, this is the same as on the CityScapes dataset.

For the training parameters, the standard cross entropy loss function with a learning rate of 10^{-3} was chosen for the backbone, as training with a lower value usually caused the model to be stuck in a local minimum. The remaining layers of the model were trained using a learning rate of an order of magnitude larger than the one used in the backbone, in this case, 10^{-2}. This is due to the transfer learning methodology used in the experiments, using a lower learning rate when training the backbone prevents the degradation of the transferred weights and yields better

results. A total of 48000 iterations was used because this revealed to be the point where the networks usually stopped improving, as seen in Figure 5.1. This is also the number of iterations that the authors of the ACDC dataset use in their experiments.

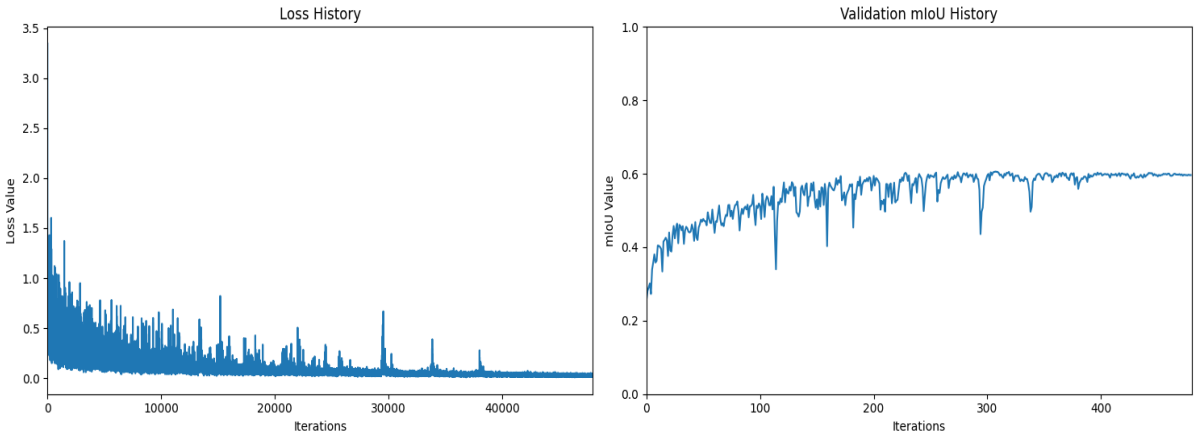


Figure 5.1: Training loss and validation mIoU of a DeeplabV3+ model with a ResNet50 backbone trained using pretrained image net weights.

An output stride of 16 for the DeepLabV3+ model was used in all experiments, as this is the value used by most works involving this model and is also the value used in the ACDC dataset publication. An example of the command used to train a DeepLabV3+ model with a ResNet50 backbone is presented in listing 5.3.

```
py main.py --model deeplabv3plus_resnet50 --gpu_id 0 --crop_val --lr 0.001
    --crop_size 768 --batch_size 2 --val_batch_size 2 --output_stride 16
    --dataset acdc --data_root ./datasets/data/ACDC --total_itrs 48000
    --random_seed 1
```

Listing 5.3: The parameters used to train a DeepLabV3+ model with a ResNet50 backbone on the ACDC training set.

5.1 Experiment 1

The goal of the first experiment is to understand if weighting the loss of a sample by its corresponding uncertainty mask improves the performance of the model. To achieve this, an ensemble of five DeepLabv3+ models with a ResNet50 backbone was trained. Each of these models was trained independently of the others, each of them with randomly initialized weights following the deep ensemble methodology. After training the networks individually, they were ensembled and the uncertainty masks were produced according to the method presented in the previous chapter.

These uncertainty masks were then used to train an additional five DeepLabV3+ models

with the ResNet50 backbone, once again with randomly initialized weights. These models were posteriorly combined in an ensemble and its perfomance compared with the previous ensemble. It was expected that the ensemble of models trained with the uncertainty masks performed slightly worse in mIoU than the ensemble of models trained without uncertainty masks, as these masks would encourage the models to learn harder classes and give less weight to majority classes, which machine learning models tend to be biased towards. Although mIoU performance was expected to be lower, the IoU value for minority classes that were neglected by models not trained with uncertainty masks was expected to be higher. The results for this experiment are reported in Table 6.1.

5.2 Experiment 2

Randomly initializing the weights of a network is the deep ensemble approach to achieve predictive diversity, but as observed in the previous experiment, doing so leads to a very large drop in model performance and consequently lower quality uncertainty predictions. To achieve prediction diversity in the ensemble while also preserving model performance, one can ensemble the same model with different network architectures as the backbone. This should lead to a big jump in ensemble performance and higher quality uncertainty masks, which in turn would improve the performance of the ensemble of models trained with said masks, although it is still expected that overall mIoU performance slightly lowers (but the IoU for harder classes improves). For this experiment, five DeepLabV3+ models with different backbones: ResNet50, ResNet101, Xception, MobileNetv2 and HRNetv2 were used. All backbones were initialized with imagenet weights, with the exception of HRNetv2 and Xception, as such weights were not available. The same training parameters and methodology as the previous experiment were used: Initially, these five models were trained individually and ensembled to generate uncertainty masks. These uncertainty masks were then inserted in the training process of another five models with the same architectures, initialized once again with imagenet weights. These models were then combined in an ensemble and its performance compared to the ensemble of models trained without uncertainty masks. Results are reported in Table 6.2.

5.3 Experiment 3

The results from experiments 1 and 2 seemed to suggest that the ensembles were overconfident in their predictions, as the performance of the ensemble of models trained with uncertainty masks severely lowered in minority classes *motorcycle* and *rider*. If the ensemble was overconfident in these classes, it would predict lower uncertainty values for them, resulting in lower loss weights and consequently worse performance. To uncover if this was the case, plots presenting the total uncertainty for each class was produced, adjusted by the corresponding proportion of pixels of each class in the dataset. It was expected that the ensemble was overall well calibrated and producing fair uncertainty values for most classes, as this is usually the case for ensembles, with

the exception of the aforementioned classes that were expected to have lower uncertainty values.

5.4 Experiment 4

This experiment tested the introduction of the uncertainty masks in the middle of the training process. Five models with different backbones were individually trained, the same as in experiment 2, with no uncertainty masks in the first 48000 iterations. Next, the models were ensembled to generate uncertainty masks and these were introduced to continue training the same models individually for another 48000 iterations. The goal was to understand if letting the models absorb the majority classes unrestrictedly in the first half of their training would prevent a drop in performance on such classes, maybe actually improving the overall mIoU when introducing the uncertainty masks in the second half, as minority classes would get higher weights in the loss function. In Table 6.3, the performance of the ensemble of models trained in the first half and the performance of the ensemble of models trained with the full training process are compared.

5.5 Experiment 5

This final experiment was conducted in order to understand how an ensemble trained with the developed method performed under domain shift. Incorporating uncertainty in the training of the models may result in an ensemble more robust to sharp changes in data. To test this hypothesis, the models obtained in the previous experiment by applying the uncertainty masks in the middle of their training process were combined in an ensemble. These models were trained on ACDC and so, they were tested on the validation set of the CityScapes dataset, resulting in a domain shift from adverse weather conditions to clear weather conditions. It was expected that the ensemble would drop a considerable amount of mIoU but would make high uncertainty predictions, showing that it is aware of the domain shift. The ensemble of models trained without uncertainty from the previous experiment was also tested in the cityscapes validation for comparison purposes.

The goal of these experiments is to rigorously test the method that was developed in order to identify if using it brings an advantage over regular training with no uncertainty masks. In the next chapter, the results for these experiments will be reported and analyzed.

Chapter 6

Results and Analysis

In this chapter, the results for the experiments that were described in the previous chapter are presented and discussed. The performance of the ensembles on the validation set in experiments 1, 2 and 4 is summarized in tables to facilitate the comparison of the IoU values obtained by each of them in each individual class. For experiments 3 and 5, bar plots presenting the total uncertainty per class predicted by the ensembles are chosen.

6.1 Experiment 1

The results for experiment 1 are reported in Table 6.1 and the detailed performance for each of the models is reported in annex A. This Table presents the IoU performance value for each class as obtained for each of the two ensembles in the experiment: the ensemble trained on regular training and that was used to generate the uncertainty masks and the ensemble trained on uncertainty aware training by using these uncertainty masks.

It was found that the performance of the ensemble is much lower when trained with the uncertainty masks, as there is a 0.042 decrease in mIoU. Unlike what was expected, IoU performance dropped for all classes, even the ones that were expected to improve. This suggests that the uncertainty masks produced by the ensemble are not of great quality. During this experiment, it was noted that randomly initializing the network's weights led to a very large drop in performance, for example, when initializing the DeepLabV3+ model with a ResNet50 backbone with imagenet weights the model was able to achieve around 0.61 mIoU, while the backbones initialized with random weights usually achieved around 0.44 mIoU. This may be due to the admittedly small size of the ACDC training set which seems to hinder the ability of the model to learn relevant features of the data. By initializing the models with imagenet weights, this problem is reduced. Models that have lower performance are typically less calibrated and more overconfident, which may result in lower uncertainty values for minority classes and consequently lower weights for them in the loss function on uncertainty aware training, which may explain the reduction in performance across the board.

Table 6.1: The performance of the ensemble of models with randomly initialized weights when trained with no uncertainty masks (left) and with uncertainty masks (right).

	Regular Training	**Uncertainty Aware Training**
road	0.93	0.93
sidewalk	0.719	0.72
building	0.785	0.758
wall	0.397	0.386
fence	0.287	0.23
pole	0.467	0.451
traffic light	0.483	0.43
traffic sign	0.305	0.264
vegetation	0.807	0.785
terrain	0.419	0.368
sky	0.941	0.93
person	0.191	0.168
rider	0.000	**0.026**
car	0.712	0.634
truck	0.256	0.22
bus	0.466	0.201
train	0.728	0.702
motorcycle	0.088	0.0069
bicycle	0.153	0.151
mIoU	0.481	0.439

6.2 Experiment 2

Table 6.2 presents the findings of experiment 2 and the detailed performance for each of the models is reported in annex B. As in the previous experiment, each entry represents the IoU value obtained by each ensemble in each class.

Compared to the previous experiment, the performance drop when training the ensemble with the uncertainty masks was much lower (0.017 mIoU). The ensemble even improved its IoU performance in classes *fence*, *traffic light*, *person* and *bicycle*, which are minority classes and were hard for the model to learn. Interestingly, IoU in classes *bus*, *rider*, and *motorcycle* significantly dropped in both experiments, once again suggesting that the ensemble predicts very low uncertainty in pixels of these classes. Since classes *rider* and *motorcycle* are minority classes, this may signal that it is overconfident in its predictions and consequently miscalibrated, just like in the previous experiment. Since there was a slighter drop in performance when training with uncertainty masks compared to the previous experiment, and that there was actually some improvement in some classes, this suggests that obtaining ensemble diversity through the usage of different backbone architectures and initializing them with imagenet weights seems to produce

Table 6.2: The performance of the ensemble of models with different backbones initialized with imagenet weights when trained with no uncertainty masks (left) and with uncertainty masks (right).

	Regular Training	**Uncertainty Aware Training**
road	0.958	0.953
sidewalk	0.808	0.798
building	0.847	0.842
wall	0.557	0.549
fence	0.412	**0.427**
pole	0.543	0.543
traffic light	0.626	**0.632**
traffic sign	0.531	0.518
vegetation	0.854	0.852
terrain	0.517	0.497
sky	0.952	0.951
person	0.424	**0.441**
rider	0.315	0.216
car	0.828	0.804
truck	0.449	0.444
bus	0.866	0.743
train	0.879	0.855
motorcycle	0.307	0.203
bicycle	0.349	**0.437**
mIoU	0.633	0.616

better quality uncertainty masks than the usage of the same backbone with randomly initialized weights.

6.3 Experiment 3

To compare the uncertainty values predicted by the ensemble of models with randomly initialized weights of experiment 1 and the ones predicted by the ensemble of models with different backbones and imagenet weights of experiment 2, the plots in Figures 6.1 and 6.2 were produced. These plots represent the total uncertainty that the ensembles predict for each class, that is, the sum of the predicted uncertainty for each pixel, for each class. Finally, this total uncertainty value is adjusted to the proportion of pixels of each class in the dataset.

By comparing Figures 6.1 and 6.2 it can be seen that both ensembles predict similar levels of uncertainty for each class. It was expected that the uncertainties predicted by the ensemble

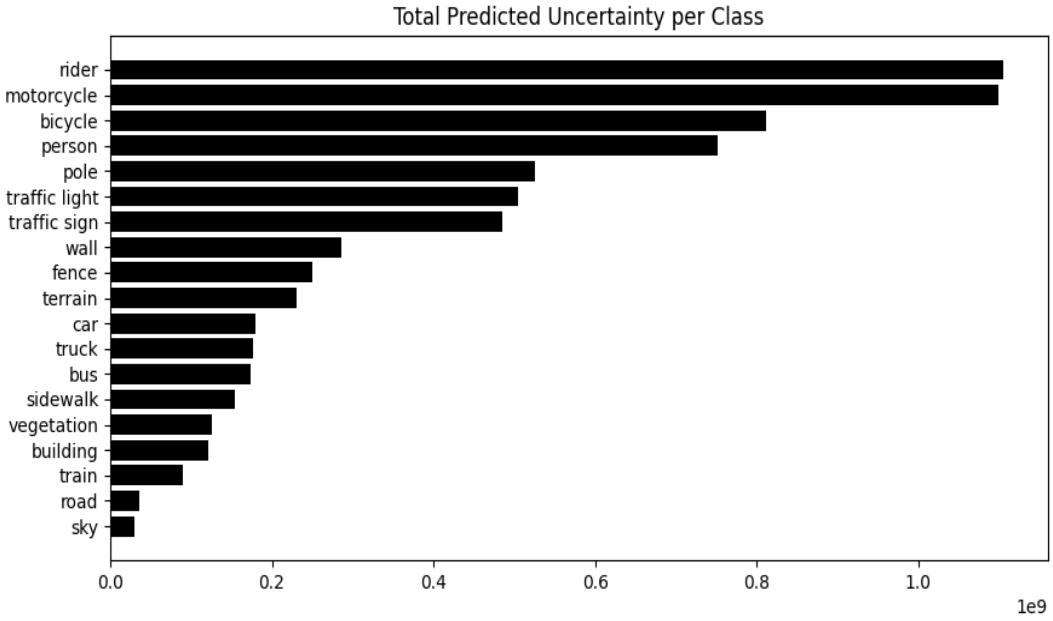


Figure 6.1: The total predicted uncertainty per class, adjusted by the proportion of the pixels in each class. These uncertainty predictions were made by the ensemble in experiment 1.

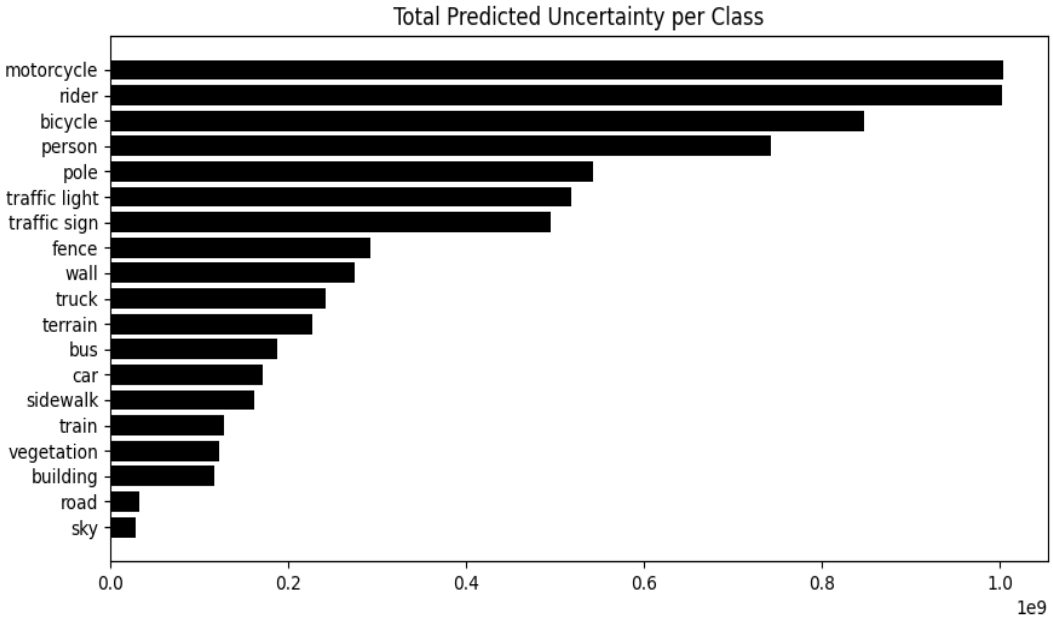


Figure 6.2: The total predicted uncertainty per class, adjusted by the proportion of the pixels in each class. These uncertainty predictions were made by the ensemble in experiment 2.

of experiment 1 (Figure 6.1) were of significant lower quality than the ones predicted by the ensemble of experiment 2 (Figure 6.2), since there is a large drop in performance when using the uncertainty masks in experiment 1, but they are indeed very similar. This is in agreement with

the literature, as ensembles are a very powerful way of estimating uncertainty. The reason for this performance drop may thus be due to the lower weights given to the majority classes during training, which hinders the overall mIoU performance of the models, and since minority classes are very scarce giving them greater weights in the loss function does not compensate for their low amount of pixels. Since the models of the ensembles in experiment 2 are initialized with imagenet weights this problem is mitigated.

6.4 Experiment 4

The results for experiment 4 are presented in Table 6.3. Just as in previous experiments, each entry represents the IoU value for an individual class. The Regular Training column reports the ensemble performance for the first half of the training process, that is, without using the uncertainty masks. The Uncertainty Aware Training column reports the ensemble performance at the end of the second half of the training process, after introducing the uncertainty masks.

Table 6.3: The performance of the ensemble of models with different backbones initialized with imagenet weights when trained with no uncertainty masks (left) and with uncertainty masks (right).

	Regular Training	**Uncertainty Aware Training**
road	0.958	0.959
sidewalk	0.808	0.809
building	0.847	0.847
wall	0.557	0.56
fence	0.412	**0.435**
pole	0.543	**0.558**
traffic light	0.626	**0.665**
traffic sign	0.531	**0.547**
vegetation	0.854	0.856
terrain	0.517	0.511
sky	0.952	0.952
person	0.424	0.416
rider	0.315	0.256
car	0.828	0.828
truck	0.449	**0.492**
bus	0.866	0.841
train	0.879	0.873
motorcycle	0.307	0.243
bicycle	0.349	**0.371**
mIoU	0.633	0.633

The overall mIoU performance after the uncertainty masks are introduced in the training

process does not drop which was as expected. Unfortunately, the model didn't overall improve. However, the model improved in minority classes, such as *fence*, *pole*, *traffic light*, *traffic sign*, *truck* and *bicycle*. Although there is an improvement in more classes than in experiment 2, the improvement in class *bicycle* was not as great, but the drop in performance in classes *rider* and *motorcycle* was much lower. These results seem to suggest that training the models first with no uncertainty masks and then retrain them with uncertainty masks serves to introduce a bias toward the majority classes, which is then adjusted by the training with the uncertainty masks, yielding more balanced models.

6.5 Experiment 5

The performance of the ensembles on the CityScapes validation set is summarized in Table 6.4, with the IoU value per class being presented. The detailed performance for each of the models is reported in annex C. In Figure 6.3, the uncertainty predictions on the same CityScapes validation set of the ensemble trained with uncertainty masks are plotted.

Table 6.4: The performance of the ensembles of models with different backbones initialized with imagenet weights when tested on the CityScapes validation set.

	Regular Training	**Uncertainty Aware Training**
road	0.901	0.844
sidewalk	0.476	0.399
building	0.783	0.785
wall	0.151	0.118
fence	0.174	0.177
pole	0.294	0.302
traffic light	0.178	0.205
traffic sign	0.413	0.425
vegetation	0.807	0.82
terrain	0.112	0.074
sky	0.764	0.781
person	0.25	0.202
rider	0.056	0.051
car	0.822	0.736
truck	0.241	0.142
bus	0.136	0.127
train	0.099	0.087
motorcycle	0.076	0.028
bicycle	0.36	0.361
mIoU	0.373	0.351

Contrary to what was expected, the ensemble trained with uncertainty masks performed much worse on the CityScapes validationset. With the mIoU on the ACDC validation set being 0.63, an mIoU of 0.35 on the CityScapes validation set is quite a performance drop. It seems that training the model with uncertainty information is not enough to overcome the large domain shift, even when the models are trained in the harder domain of adverse weather conditions. The biggest performance drop is on minority classes, but interestingly, the model seems to lose its ability to classify the *train* class on the CityScapes validation set, while it scores 0.873 IoU on the ACDC validation set. In Figure 6.3, it can be seen that the model predicts high uncertainty values for this class and generally predicts higher uncertainty values for the classes where performance drops the most. Although the model has better performance in classes such as *bicycle*, *sidewalk*

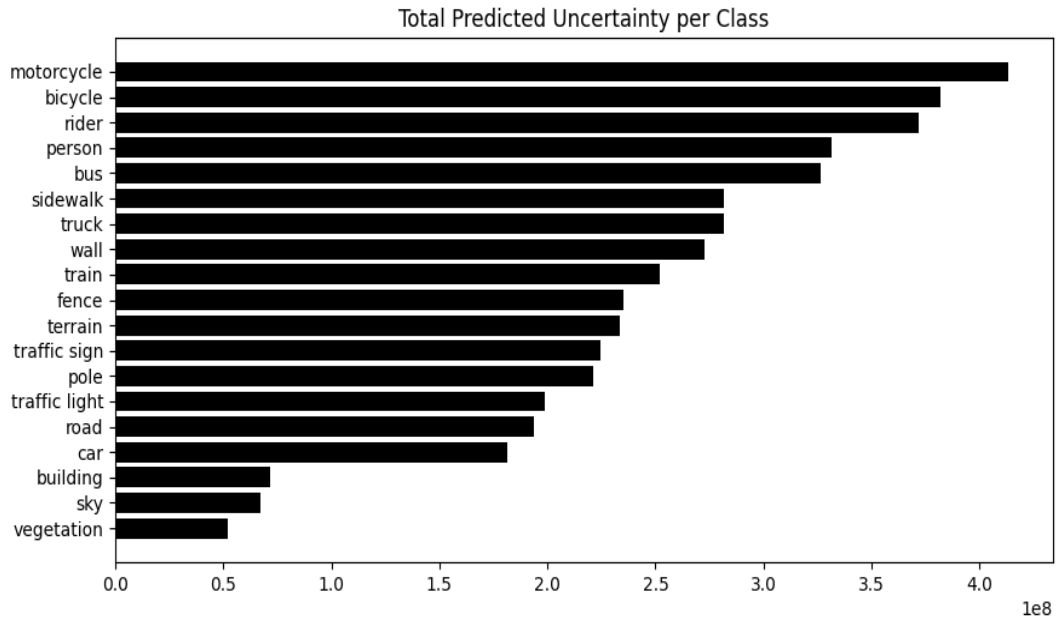


Figure 6.3: Uncertainty predictions on the validation set of Cityscapes of an ensemble of models trained on ACDC.

and *bus* than in the *train* class, it predicts higher uncertainty values for the former classes, which signals overconfidence and miscalibration. It also predicts lower total uncertainty values than in the previous experiment, signalling overconfidence once again. Despite this overconfidence, these uncertainty predictions are not absurd and make sense. However, training models with uncertainty masks does not seem to make them more robust to domain shift, as the performance of the ensemble trained without uncertainty masks is higher, even in minority classes, although more experiments should be performed to confirm this.

Chapter 7

Conclusions

Recent advancements in AI and autonomous driving have made self driving cars an increasingly closer reality. Although challenges remain, such as the reliability of perception systems in adverse weather conditions, with each passing day, these cars are getting smarter, safer and more autonomous. Uncertainty quantification and calibration plays a crucial role in the interpretation of the data collected by these perception systems, as it ensures that they are not overconfident in their predictions and increases their robustness and reliability. Decision systems can thus put more trust in perception systems, which will in turn enable them to make better decisions and increase the safety of self driving cars. Nowadays, most companies involved in autonomous driving use a variety of different sensors for the perception task, but these sensors are quite expensive and the fusion of all the information collected by them is not a simple problem to solve. This makes regular cameras still relevant for the perception of the surrounding environment, and the processing of the images collected by them is not solved yet. Having this in mind, the focus of this work was to study uncertainty quantification techniques in semantic segmentation of road scene images under adverse weather conditions, as this is the domain where perception systems usually struggle the most. A detailed literature review of the current state of the art solutions for these difficulties was presented in this work, as well as a very simple method to increase the performance of a perception deep learning model using uncertainty information.

The first step in this work was the choice of open access datasets to use as benchmarks. Two well established datasets related to autonomous driving were used: one consisting of road scene images under clear weather conditions and the other comprising road scene images under adverse weather conditions. This enabled the rigorous testing and evaluation of the performance of the method across different settings. This simple uncertainty aware training method started by using an uncertainty quantification method, namely an ensemble of five CNN models, to generate uncertainty predictions for each pixel of the images in the training set, resulting in an uncertainty mask for each of these images. The uncertainty masks were then integrated into the training loss of new (or even the same) models to try and improve the performance of the ensemble. The experiments that were conducted found that using this method usually increased performance on minority classes of the dataset, which was its goal.

Although the usage of uncertainty information in the training process of a model seemed to improve its performance on harder classes, this usually came at the cost of a degrading of performance in easier classes as the pixels of such classes were assigned lower weights in the training loss. A limitation of the developed method is that it is entirely dependent on the uncertainty predictions made by an uncertainty quantification method. Thus, its performance may vary with the choice of other uncertainty quantification methodologies. Using ensembles is also quite computationally heavy, as CNNs are not cheap to train and the method requires that the same CNNs twice: once to produce the ensemble that predicts the uncertainty masks and again to incorporate them in training. These problems can be addressed in future work.

The developed uncertainty aware training method achieved its original goal but steps can be taken to improve it. As previously noted, the repeating of the training of ensembles is extremely expensive and may even be prohibitively costly when using larger models. Thus, lighter uncertainty methods can be studied and employed to generate the uncertainty masks. An interesting finding from the experiments that were conducted was the drop in performance in classes *motorcycle* and *rider* after training with uncertainty masks. At first thought, this seemed to indicate that the ensemble was predicting low uncertainty for said classes, but this revealed to not be the case. Further research should be carried out to find the reason for this problem, as solving this drop in performance on these classes would improve the ensemble's performance significantly. Since the literature recommends the usage of an explicit calibration method for ensembles to perform uncertainty quantification, this would also be an interesting experiment to conduct, though the ensembles that were used in this work were not miscalibrated and no such method was used. Another line of experiments that should be followed would be the usage of lighter, real-time semantic segmentation CNNs. Although DeepLabV3+ achieves good performance in many different datasets, this architecture may be too heavy to be deployed in an autonomous driving application, as fast predictions are necessary. Since these real-time models usually achieve lower performance than heavier models, this uncertainty aware training method may serve to improve their performance. Further exploration of the usage of semantic segmentation models under adverse weather conditions should be conducted. It would be interesting to try to model the aleatoric uncertainty present in the data according to each weather condition and also use it to enhance the model.

In this work, uncertainty quantification was only addressed as a means to improve the performance of a deep learning model. However, the utility of uncertainty predictions is not limited to this specific case and goes well beyond it. For example, before self driving cars are fully autonomous, there is the need for a safety system that transfers the control of the vehicle to the driver when the system recognizes that it does not know how to drive safely under the conditions that may be present. This safety system would then have to "know what it does not know" and this is where uncertainty predictions and calibrated models would have to come in. Future work could also involve the development of such a system, where perception and decision making models are calibrated and trained with the method that was developed in this work.

Appendix A

Performance of the Individual Models in Experiment 1

This appendix contains the detailed performance of each model in the ensembles of experiment 1. Table C.1 reports the results obtained from each model when trained with no uncertainty masks, while Table A.2 reports the results of the models trained with uncertainty masks.

Table A.1: The performance of each of the models in the ensemble of models with randomly initialized weights when trained with no uncertainty masks.

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**
road	0.919	0.922	0.914	0.917	0.918
sidewalk	0.684	0.689	0.68	0.681	0.687
building	0.756	0.762	0.76	0.76	0.762
wall	0.378	0.35	0.37	0.331	0.351
fence	0.265	0.236	0.26	0.237	0.246
pole	0.422	0.438	0.32	0.426	0.429
traffic light	0.401	0.411	0.42	0.453	0.442
traffic sign	0.274	0.267	0.265	0.254	0.283
vegetation	0.786	0.787	0.793	0.784	0.788
terrain	0.369	0.38	0.389	0.392	0.376
sky	0.933	0.933	0.935	0.934	0.934
person	0.153	0.16	0.171	0.184	0.172
rider	0.000	0.014	0.000	0.02	0.02
car	0.653	0.673	0.666	0.657	0.663
truck	0.256	0.192	0.243	0.196	0.238
bus	0.424	0.375	0.312	0.35	0.389
train	0.691	0.711	0.55	0.714	0.685
motorcycle	0.083	0.061	0.015	0.112	0.072
bicycle	0.181	0.116	0.097	0.11	0.114
mIoU	0.454	0.446	0.435	0.448	0.451

Table A.2: The performance of each of the models in the ensemble of models with randomly initialized weights when trained with uncertainty masks.

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**
road	0.914	0.9	0.915	0.914	0.912
sidewalk	0.663	0.629	0.666	0.677	0.665
building	0.726	0.711	0.732	0.733	0.736
wall	0.329	0.317	0.347	0.336	0.346
fence	0.221	0.196	0.189	0.212	0.191
pole	0.401	0.392	0.404	0.406	0.406
traffic light	0.386	0.361	0.41	0.354	0.396
traffic sign	0.227	0.229	0.22	0.236	0.234
vegetation	0.764	0.758	0.763	0.769	0.769
terrain	0.334	0.342	0.334	0.34	0.332
sky	0.93	0.924	0.925	0.927	0.926
person	0.14	0.137	0.14	0.129	0.153
rider	0.013	0.013	0.021	0.018	0.016
car	0.594	0.556	0.585	0.596	0.594
truck	0.21	0.145	0.241	0.144	0.216
bus	0.166	0.179	0.188	0.194	0.182
train	0.613	0.65	0.637	0.592	0.606
motorcycle	0.017	0.016	0.024	0.09	0.009
bicycle	0.113	0.116	0.105	0.11	0.175
mIoU	0.409	0.398	0.413	0.405	0.414

Appendix B

Performance of the Individual Models in Experiment 2

This appendix contains the detailed performance of each model in the ensembles of experiment 2. Table B.1 reports the results obtained from each model when trained with no uncertainty masks, while Table B.2 reports the results of the models trained with uncertainty masks.

Table B.1: The performance of each of the models in the ensemble of models initialized with imagenet weights in experiment 2, trained with no uncertainty masks.

	ResNet50	**ResNet101**	**HRNet**	**Xception**	**MobileNet**
road	0.954	0.954	0.925	0.922	0.945
sidewalk	0.789	0.794	0.7	0.686	0.764
building	0.836	0.835	0.77	0.76	0.824
wall	0.521	0.524	0.383	0.397	0.5
fence	0.412	0.413	0.265	0.25	0.363
pole	0.507	0.506	0.454	0.375	0.494
traffic light	0.634	0.63	0.398	0.35	0.581
traffic sign	0.542	0.524	0.278	0.216	0.492
vegetation	0.848	0.85	0.797	0.789	0.837
terrain	0.49	0.503	0.366	0.382	0.492
sky	0.946	0.95	0.935	0.934	0.946
person	0.392	0.425	0.161	0.151	0.396
rider	0.207	0.217	0.021	0.000	0.28
car	0.816	0.835	0.663	0.661	0.796
truck	0.375	0.48	0.326	0.206	0.407
bus	0.775	0.825	0.54	0.537	0.784
train	0.841	0.856	0.675	0.673	0.843
motorcycle	0.246	0.3	0.0471	0.025	0.265
bicycle	0.386	0.284	0.075	0.085	0.404
mIoU	0.606	0.616	0.462	0.442	0.6

Table B.2: The performance of each of the models in the ensemble of models initialized with imagenet weights in experiment 2, trained with uncertainty masks

	ResNet50	**ResNet101**	**HRNet**	**Xception**	**MobileNet**
road	0.952	0.95	0.897	0.922	0.941
sidewalk	0.791	0.784	0.626	0.695	0.749
building	0.838	0.835	0.722	0.743	0.815
wall	0.526	0.529	0.287	0.37	0.474
fence	0.43	0.428	0.199	0.236	0.379
pole	0.517	0.521	0.404	0.368	0.492
traffic light	0.644	0.651	0.322	0.356	0.597
traffic sign	0.54	0.527	0.231	0.215	0.471
vegetation	0.846	0.847	0.766	0.779	0.834
terrain	0.485	0.474	0.344	0.368	0.462
sky	0.947	0.947	0.922	0.933	0.944
person	0.44	0.452	0.131	0.122	0.405
rider	0.143	0.317	0.03	0.004	0.172
car	0.794	0.826	0.552	0.606	0.77
truck	0.441	0.501	0.166	0.237	0.344
bus	0.705	0.76	0.192	0.304	0.647
train	0.829	0.812	0.582	0.678	0.802
motorcycle	0.227	0.028	0.0471	0.01	0.07
bicycle	0.462	0.418	0.114	0.113	0.33
mIoU	0.608	0.624	0.396	0.424	0.563

Appendix C

Performance of the Individual Models in Experiment 5

This appendix contains the detailed performance of each model in the ensembles of experiment 5. Table C.1 reports the results obtained from each model when trained with uncertainty masks, initialized with the weights of the previous training with no uncertainty masks.

Table C.1: The performance of each individual model of the ensemble in experiment 5.

	ResNet50	**ResNet101**	**HRNet**	**Xception**	**MobileNet**
road	0.955	0.953	0.931	0.923	0.942
sidewalk	0.796	0.794	0.723	0.704	0.758
building	0.833	0.836	0.781	0.766	0.822
wall	0.516	0.535	0.402	0.404	0.5
fence	0.395	0.435	0.271	0.275	0.41
pole	0.522	0.516	0.472	0.398	0.498
traffic light	0.643	0.636	0.46	0.38	0.6
traffic sign	0.545	0.513	0.336	0.257	0.49
vegetation	0.849	0.848	0.802	0.795	0.836
terrain	0.496	0.481	0.395	0.387	0.482
sky	0.95	0.949	0.938	0.935	0.94
person	0.419	0.388	0.183	0.133	0.417
rider	0.201	0.153	0.07	0.046	0.264
car	0.824	0.812	0.689	0.67	0.786
truck	0.456	0.474	0.307	0.254	0.381
bus	0.79	0.797	0.519	0.455	0.709
train	0.839	0.853	0.695	0.686	0.818
motorcycle	0.224	0.247	0.079	0.023	0.2
bicycle	0.337	0.337	0.14	0.132	0.404
mIoU	0.61	0.608	0.484	0.454	0.592

Bibliography

[1] https://github.com/vainf/deeplabv3plus-pytorch.

[2] https://towardsdatascience.com/a-primer-on-atrous-convolutions-and-depth-wise-separable-convolutions-443b106919f5.

[3] Pytorch: An imperative style, high-performance deep learning library, 2019.

[4] Resnet - neural network architectures. https://neurohive.io/en/popular-networks/resnet/, Accessed 2023-10-01.

[5] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, dec 2021. doi:10.1016/j.inffus.2021.05.008.

[6] Alexander Amini, Ava Soleimany, Sertac Karaman, and Daniela Rus. Spatial uncertainty sampling for end-to-end control, 2019.

[7] Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning, 2021.

[8] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation, 2016.

[9] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences, 2019.

[10] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks, 2015.

[11] Kai Brach, Beate Sick, and Oliver Dürr. Single shot mc dropout approximation, 2020.

[12] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009. ISSN: 0167-8655. Video-based Object and Event Analysis. doi:https://doi.org/10.1016/j.patrec.2008.04.005.

[13] Kirill Bykov, Marina M. C. Höhne, Adelaida Creosteanu, Klaus-Robert Müller, Frederick Klauschen, Shinichi Nakajima, and Marius Kloft. Explaining bayesian neural networks, 2021.

[14] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving, 2020.

[15] Eduardo D. C. Carvalho, Ronald Clark, Andrea Nicastro, and Paul H. J. Kelly. Scalable uncertainty for computer vision with functional variational inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[16] Ferhat Ozgur Catak, Tao Yue, and Shaukat Ali. Prediction surface uncertainty quantification in object detection models for autonomous driving. In *2021 IEEE International Conference on Artificial Intelligence Testing (AITest)*, pages 93–100, 2021. doi:10.1109/AITEST52744.2021.00027.

[17] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs, 2016.

[18] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, 2017.

[19] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[20] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation, 2018.

[21] Tianqi Chen, Emily B. Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo, 2014.

[22] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2017.

[23] Mehmet Akif Cifci. A deep learning-based framework for uncertainty quantification in medical imaging using the dropweak technique: An empirical study with baresnet. *Diagnostics*, 13(4), 2023. ISSN: 2075-4418. doi:10.3390/diagnostics13040800.

[24] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016.

[25] Sebastian Cygert, Bartlomiej Wroblewski, Karol Wozniak, Radoslaw Slowinski, and Andrzej Czyzewski. Closer look at the uncertainty estimation in semantic segmentation under distributional shift. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2021. doi:10.1109/ijcnn52387.2021.9533330.

[26] Dimah Dera, Ghulam Rasool, and Nidhal Bouaynaya. Extended variational inference for propagating uncertainty in convolutional neural networks. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2019. doi:10.1109/MLSP.2019.8918747.

[27] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[28] Romain Egele, Romit Maulik, Krishnan Raghavan, Bethany Lusch, Isabelle Guyon, and Prasanna Balaprakash. Autodeuq: Automated deep ensemble with uncertainty quantification. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 1908–1914, 2022. doi:10.1109/ICPR56361.2022.9956231.

[29] Di Feng, Christian Haase-Schutz, Lars Rosenbaum, Heinz Hertlein, Claudius Glaser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1341–1360, mar 2021. doi:10.1109/tits.2020.2972974.

[30] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis, 2016.

[31] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, 2016.

[32] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu. A survey of uncertainty in deep neural networks, 2022.

[33] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. doi:10.1109/CVPR.2012.6248074.

[34] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S. Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, Tiffany Fernandez, Martin Jänicke, Sudesh Mirashi, Chiragkumar Savani, Martin Sturm, Oleksandr Vorobiov, Martin Oelker, Sebastian Garreis, and Peter Schuberth. A2D2: Audi Autonomous Driving Dataset. 2020.

[35] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. *CoRR*, abs/1706.04599, 2017.

[36] Corina Gurau, Alex Bewley, and Ingmar Posner. Dropout distillation for efficiently estimating model confidence, 2018.

[37] Fredrik K. Gustafsson, Martin Danelljan, and Thomas B. Schön. Evaluating scalable bayesian deep learning methods for robust computer vision, 2020.

[38] Nils Gählert, Nicolas Jourdan, Marius Cordts, Uwe Franke, and Joachim Denzler. Cityscapes 3d: Dataset and benchmark for 9 dof vehicle detection, 2020.

[39] Qishen Ha, Kohei Watanabe, Takumi Karasawa, Yoshitaka Ushiku, and Tatsuya Harada. Mfnet: Towards real-time semantic segmentation for autonomous vehicles with multi-spectral scenes. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5108–5115, 2017. doi:10.1109/IROS.2017.8206396.

[40] Shijie Hao, Yuan Zhou, and Yanrong Guo. A brief survey on semantic segmentation with deep learning. *Neurocomputing*, 406:302–321, 2020. ISSN: 0925-2312. doi:https://doi.org/10.1016/j.neucom.2019.11.118.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[42] Sergio Hernández, Diego Vergara, Matías Valdenegro-Toro, and Felipe Jorquera. Improving predictive uncertainty estimation using dropout–hamiltonian monte carlo. *Soft Computing*, 24(6):4307–4322, 2020. ISSN: 1433-7479. doi:10.1007/s00500-019-04195-w.

[43] Carl-Johan Hoel, Krister Wolff, and Leo Laine. Tactical decision-making in autonomous driving by reinforcement learning with uncertainty estimation. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1563–1569, 2020. doi:10.1109/IV47402.2020.9304614.

[44] Carl-Johan Hoel, Krister Wolff, and Leo Laine. Ensemble quantile networks: Uncertainty-aware reinforcement learning with applications in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 24(6):6030–6041, 2023. doi:10.1109/TITS.2023.3251376.

[45] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[46] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[47] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[48] Xinyu Huang, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The ApolloScape open dataset for autonomous driving and its application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(10):2702–2719, oct 2020. doi:10.1109/tpami.2019.2926463.

[49] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: A tutorial introduction. *CoRR*, abs/1910.09457, 2019.

[50] Built In. Self-driving cars: A comprehensive guide, 2023. Accessed on 2023-09-27.

[51] Moksh Jain, Salem Lahlou, Hadi Nekoei, Victor Butoi, Paul Bertin, Jarrid Rector-Brooks, Maksym Korablyov, and Yoshua Bengio. DEUP: direct epistemic uncertainty prediction. *CoRR*, abs/2102.08501, 2021.

[52] Jongoh Jeong and Jong-Hwan Kim. Doubly contrastive end-to-end semantic segmentation for autonomous driving under adverse weather, 2022.

[53] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding, 2016.

[54] Abdulrahman Kerim, Felipe Chamone, Washington Ramos, Leandro Soriano Marcolino, Erickson R. Nascimento, and Richard Jiang. Semantic segmentation under adverse conditions: A weather and nighttime-aware synthetic data-based approach, 2022.

[55] Samin Khan, Buu Phan, Rick Salay, and Krzysztof Czarnecki. Procsy: Procedural synthetic dataset generation towards influence factor studies of semantic segmentation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[56] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[57] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural Safety*, 31(2):105–112, 2009. ISSN: 0167-4730. Risk Acceptance and Risk Communication. doi:https://doi.org/10.1016/j.strusafe.2008.06.020.

[58] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

[59] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[60] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi:10.1109/5.726791.

[61] Chunyuan Li, Andrew Stevens, Changyou Chen, Yunchen Pu, Zhe Gan, and Lawrence Carin. Learning weight uncertainty with stochastic gradient mcmc for shape classification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5666–5675, 2016. doi:10.1109/CVPR.2016.611.

[62] Hanchao Li, Pengfei Xiong, Haoqiang Fan, and Jian Sun. Dfanet: Deep feature aggregation for real-time semantic segmentation, 2019.

[63] Zhi Li, Xing Wu, Jianjia Wang, and Yike Guo. Weather-degraded image semantic segmentation with multi-task knowledge distillation. *Image and Vision Computing*, 127: 104554, 2022. ISSN: 0262-8856. doi:https://doi.org/10.1016/j.imavis.2022.104554.

[64] Jeremiah Liu, John Paisley, Marianthi-Anna Kioumourtzoglou, and Brent Coull. Accurate uncertainty estimation and decomposition in ensemble learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[65] Jeremiah Zhe Liu, Shreyas Padhy, Jie Ren, Zi Lin, Yeming Wen, Ghassen Jerfel, Zack Nado, Jasper Snoek, Dustin Tran, and Balaji Lakshminarayanan. A simple approach to improve single-model deep uncertainty via distance-awareness, 2022.

[66] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1): 3–15, 2017. doi:10.1177/0278364916679498.

[67] Andrey Malinin. *Uncertainty Estimation in Deep Learning with application to Spoken Language Assessment*. PhD thesis, Department Of Engineering, University Of Cambridge, 2019. doi:10.17863/CAM.45912.

[68] Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. Ensemble distribution distillation, 2019.

[69] Ameur M. Manceur and Pierre Dutilleul. Maximum likelihood estimation for the tensor normal distribution: Algorithm, minimum sample size, and empirical bias and dispersion. *Journal of Computational and Applied Mathematics*, 239:37–49, 2013. ISSN: 0377-0427. doi:https://doi.org/10.1016/j.cam.2012.09.017.

[70] Patrick McClure and Nikolaus Kriegeskorte. Representing inferential uncertainty in deep neural networks through sampling, 2017.

[71] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation, 2016.

[72] Yujian Mo, Yan Wu, Xinneng Yang, Feilin Liu, and Yujun Liao. Review the state-of-the-art technologies of semantic segmentation based on deep learning. *Neurocomputing*, 493: 626–646, 2022. ISSN: 0925-2312. doi:https://doi.org/10.1016/j.neucom.2022.01.005.

[73] Aryan Mobiny, Hien V. Nguyen, Supratik Moulik, Naveen Garg, and Carol C. Wu. Dropconnect is effective in modeling uncertainty of bayesian deep networks, 2019.

[74] Mohammad Pezeshki Naeini, George F Cooper, and Milo Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. *Proc Conf AAAI Artif Intell*, pages 2901–2907, 2015.

[75] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5000–5009, 2017. doi:10.1109/ICCV.2017.534.

[76] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation, 2015.

[77] NPR. Nearly 400 car crashes in 11 months involved automated tech, 2022. Accessed on 2023-09-27.

[78] Manfred Opper and Cédric Archambeau. The Variational Gaussian Approximation Revisited. *Neural Computation*, 21(3):786–792, 03 2009. ISSN: 0899-7667. doi:10.1162/neco.2008.08-07-592.

[79] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift, 2019.

[80] Theodore Papamarkou, Jacob Hinkle, M. Todd Young, and David Womble. Challenges in Markov Chain Monte Carlo for Bayesian Neural Networks. *Statistical Science*, 37(3):425 – 442, 2022. doi:10.1214/21-STS840.

[81] Yookoon Park and David M. Blei. Density uncertainty layers for reliable uncertainty estimation, 2023.

[82] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation, 2016.

[83] Andreas Pfeuffer and Klaus Dietmayer. Robust semantic segmentation in adverse weather conditions by means of sensor data fusion. In *2019 22th International Conference on Information Fusion (FUSION)*, pages 1–8, 2019. doi:10.23919/FUSION43075.2019.9011192.

[84] Buu Phan, Samin Khan, Rick Salay, and Krzysztof Czarnecki. Bayesian uncertainty quantification with synthetic data. In *Computer Safety, Reliability, and Security: SAFECOMP 2019 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Turku, Finland, September 10, 2019, Proceedings*, page 378–390, Berlin, Heidelberg, 2019. Springer-Verlag. ISBN: 978-3-030-26249-5. doi:10.1007/978-3-030-26250-1_31.

[85] Matthew Pitropov, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. Canadian adverse driving conditions dataset. *The International Journal of Robotics Research*, 40(4-5):681–690, dec 2020. doi:10.1177/0278364920979368.

[86] Konstantin Posch, Jan Steinbrener, and Jürgen Pilz. Variational inference to measure model uncertainty in deep neural networks, 2019.

[87] The Washington Post. Tesla autopilot system involved in recent crash, 2023. Accessed on 2023-09-27.

[88] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2016.

[89] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games, 2016.

[90] Max Roser. Causes of death globally: what do people die from? *Our World in Data*, 2021. https://ourworldindata.org/causes-of-death-treemap.

[91] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Semantic foggy scene understanding with synthetic data. *International Journal of Computer Vision*, 126(9):973–992, mar 2018. doi:10.1007/s11263-018-1072-8.

[92] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Guided curriculum model adaptation and uncertainty-aware evaluation for semantic nighttime image segmentation, 2019.

[93] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Acdc: The adverse conditions dataset with correspondences for semantic driving scene understanding, 2021.

[94] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. doi:10.1109/CVPR.2018.00474.

[95] Yichen Shen, Zhilu Zhang, Mert R. Sabuncu, and Lin Sun. Real-time uncertainty estimation in computer vision via uncertainty-aware distribution distillation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 707–716, January 2021.

[96] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[97] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[98] Statista. Global autonomous vehicle market size forecast, 2023. Accessed on 2023-09-27.

[99] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7262–7272, October 2021.

[100] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao,

Aditya Joshi, Sheng Zhao, Shuyang Cheng, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset, 2020.

[101] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[102] Xiaolin Tang, Kai Yang, Hong Wang, Jiahang Wu, Yechen Qin, Wenhao Yu, and Dongpu Cao. Prediction-uncertainty-aware decision-making for autonomous vehicles. *IEEE Transactions on Intelligent Vehicles*, 7(4):849–862, 2022. doi:10.1109/TIV.2022.3188662.

[103] TechCrunch. Waymo self-driving car incident involving a dog, 2023. Accessed on 2023-09-27.

[104] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christopher Bregler. Efficient object localization using convolutional networks, 2015.

[105] Abhinav Valada, Johan Vertens, Ankit Dhall, and Wolfram Burgard. Adapnet: Adaptive semantic segmentation in adverse environmental conditions. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4644–4651, 2017. doi:10.1109/ICRA.2017.7989540.

[106] Matias Valdenegro-Toro. Deep sub-ensembles for fast uncertainty estimation in image classification, 2019.

[107] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[108] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[109] Hai Wang, Yanyan Chen, Yingfeng Cai, Long Chen, Yicheng Li, Miguel Angel Sotelo, and Zhixiong Li. Sfnet-n: An improved sfnet algorithm for semantic segmentation of low-light autonomous driving road scenes. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):21405–21417, 2022. doi:10.1109/TITS.2022.3177615.

[110] Haochen Wang, Xiaolong Jiang, Haibing Ren, Yao Hu, and Song Bai. Swiftnet: Real-time video object segmentation, 2021.

[111] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition, 2020.

[112] Sida Wang and Christopher Manning. Fast dropout training. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 118–126, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[113] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. ICML'11, page 681–688, Madison, WI, USA, 2011. Omnipress. ISBN: 9781450306195.

[114] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[115] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning, 2020.

[116] Sheheryar Zaidi, Arber Zela, Thomas Elsken, Chris C Holmes, Frank Hutter, and Yee Teh. Neural ensemble search for uncertainty estimation and dataset shift. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 7898–7911. Curran Associates, Inc., 2021.

[117] Chaoning Zhang, Philipp Benz, Dawit Mureja Argaw, Seokju Lee, Junsik Kim, Francois Rameau, Jean-Charles Bazin, and In So Kweon. Resnet or densenet? introducing dense shortcuts to resnet, 2020.

[118] Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5852–5861. PMLR, 10–15 Jul 2018.

[119] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network, 2017.

[120] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images, 2018.

[121] Yang Zhao, Wei Tian, and Hong Cheng. Pyramid bayesian method for model uncertainty evaluation of semantic segmentation in autonomous driving. *Automotive Innovation*, 5, 01 2022. doi:10.1007/s42154-021-00165-x.

[122] Weitao Zhou, Zhong Cao, Nanshan Deng, Kun Jiang, and Diange Yang. Identify, estimate and bound the uncertainty of reinforcement learning for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–11, 2023. doi:10.1109/TITS.2023.3266885.