SAFETY OF AUTOMOTIVE SYSTEMS

Using Machine Learning

Victor **Besnier**

November 28th, 2022

A dissertation submitted for the degree of
Doctor of Philosophy from CY Cergy Paris University

École doctorale n°407 EM2PSI
Économie, Management, Mathématiques, Physique & Sciences Informatiques

David Picard	Senior Research Scientist, École des Ponts ParisTech	Directeur
Aymeric Histace	Full Professor, ENSEA	Co-Directeur
Alexandre Briot	Research Engineer, Valeo	Co-Encadrant
Andrei Bursuc	Research Scientist, Valeo.ai	Co-Encadrant
Isabelle Bloch	Full Professor, Sorbonne Université	Présidente
Nicolas Thome	Full Professor, Sorbonne Université	Rapporteur
Vincent Fremont	Full Professor, École Centrale de Nantes	Rapporteur
Fatma Güney	Asst Professor, Koç University	Examinatrice
Slobodan Ilic	Senior Research Scientist, Siemen and TUM	Examinateur

Acknowledgments

Avant toute chose, j'aimerai remercier tous ceux qui ont contribué de près ou de loin à l'aboutissement de mon doctorat.

En premier lieu, je souhaite exprimer toute ma gratitude envers David Picard, qui a su me conseiller et m'orienter tout au long de ces trois années. J'ai apprécié chacune des discussions que j'ai eues avec lui. Il m'a donné confiance et liberté dans l'exploration de mes idées de recherche, tout en me partageant ses brillantes intuitions. Je n'aurai pas pu imaginer un meilleur directeur de thèse.

Ensuite, mes pensées vont naturellement vers Alexandre Briot qui m'a épaulé à travers chaque épreuve. J'ai particulièrement aimé le soutien et la sollicitude qu'il m'a apporté durant ces années très particulières. Grâce à lui, j'ai pu concilier recherche théorique et appliquée pendant cette thèse. C'est avec son aide que j'ai pu apprécier pleinement mon expérience au sein de Valeo.

De plus, je tiens également à remercier Andrei Bursuc pour son implication sans faille dans le projet, sa disponibilité permanente et ses nombreux encouragements. Le partage de ses connaissances m'a beaucoup apporté dans l'aboutissement de mes différents projets. Il a su valoriser l'ensemble des travaux que nous avons entrepris ensemble.

Je souhaite aussi remercier les membres de mon jury : Isabelle Bloch, Nicolas Thome, Vincent Fremont, Fatma Güney et Slobodan Ilic pour leurs commentaires pertinents sur mon manuscrit, mais également pour les discussions qui ont suivi la soutenance de ma thèse. Cela laisse place à de belles réflexions, qui pourront aider dans la suite des recherches sur le sujet.

Pendant trois ans, j'ai eu le privilège de travailler auprès de nombreuses personnes chez Valeo. J'ai particulièrement apprécié les discussions avec Patrick Perez, Souhaiel Khalfaoui, Islam Adel, Mateus Riva et Yesmina Jaafra. Ainsi qu'avec Abdelillah Ymlahi et Stephane Lhostis qui m'ont permis de me former sur des sujets connexes à ma thèse. Je serai éternellement reconnaissant envers Lihao Wang et Rachid Benmokhtar d'avoir accepté de me livrer leurs secrets sur le fonctionnement de la Learning Car. Subséquemment, j'aimerai également remercier Aymeric Histace de l'ENSEA pour sa bienveillance. Il a permis l'aboutissement et la concrétisation du projet de la Learning Car.

Je souhaite aussi remercier tous les doctorants du laboratoire des Ponts et particulièrement Elliot Vincent, Lucas Ventura, Charle Raude, Hannah Bull, Nicolas Dufour, Yue Zhu, Michaël Ramamonjisoa, Tom Monnier, Thibault Issenhuth, Marie Morgane Paumard, Monika Wysoczanska, Philippe Chibère, Simon Roburin, Romain Loiseau, Rahima Djahel, Pierre-Alain Langlois, Nicolas Gonthier, Nguyen Nguyen, Nermin Samet, Xi Shen, Mathis Petrovich, Liza Belos, Hugo Germain, Corentin Sautier, Clément Riu, Antoine Guédon, Yanis Siglidis, Abdou Bedouhene, ainsi que tous les autres membres du laboratoire Imagine de l'école des Ponts ParisTech.

Pour finir, j'aimerai remercier mes parents Alexandra et Denis, ainsi que ma soeur Madeleine et mes frères Lucien et Émile pour leurs encouragements. Mais également Clémence Lanco pour son soutien et son aide pour la relecture de ce manuscrit. Enfin, j'aimerai remercier mes amis d'avoir été aussi disponibles et compréhensifs pendant toutes ces années.

Résumé

Les réseaux de neurones profonds sont impliqués dans le processus de prise de décision des voitures autonomes où des vies sont en jeu quand bien même ces réseaux ne sont pas toujours fiables. Les voitures autonomes utilisent des modèles de Deep Learning pour construire une représentation de leurs environnements, c'est-à-dire où sont les piétons, vers où se dirige la moto, ou encore, quelle est la couleur du feu de signalisation. Les réseaux de neurones profonds sont le résultat d'un schéma d'apprentissage sophistiqué, d'une architecture de modèle complexe et de la construction d'un jeu de données à la fois abondantes et variées. Ainsi, les prédictions des réseaux de neurones s'avèrent parfois difficiles à interpréter et peu fiables.

Dans cette thèse, nous proposons d'améliorer la sûreté des réseaux de neurones en utilisant des réseaux d'observateurs, pour détecter les comportements anormaux d'un réseau de neurones cible. Nous introduisons une architecture mettant en œuvre réseau neuronal supplémentaire, appelé ObsNet, un modèle dédié à la détection rapide et précise des anomalies dans la segmentation des scènes de route. Ce réseau auxiliaire observe les activations internes d'un réseau cible afin de déclencher une alerte lorsqu'il rencontre une prédiction non fiable. Nous entraînons le réseau observateur sur les erreurs d'un réseau cible gelé, laissant ses performances prédictives inchangées pour la tâche primaire de la segmentation d'images.

Pour notre première contribution, nous utilisons les prédictions d'un oracle pour superviser l'entraînement du réseau auxiliaire. L'ObsNet apprend à prédire une incertitude basée sur la divergence de prédiction entre le réseau cible et l'oracle. Notre méthode met en évidence une forte détection des erreurs pour les images corrompues par des artefacts comme l'éblouissement du soleil ou la pluie.

Pour notre deuxième contribution, nous utilisons des attaques adverses locales pour aider à stabiliser l'apprentissage lorsque peu de données significatives peuvent être fournies à l'observateur. Nous montrons que l'ObsNet résultant réussit à détecter les objets en dehors de la distribution pour la segmentation sémantique. Nous obtenons des résultats compétitifs pour la détection d'objets hors distribution, tout en limitant le temps d'exécution total.

Dans notre troisième contribution, nous ciblons la détection d'anomalies pour la segmentation d'instances. Nous récupérons des masques d'instance pour agréger et filtrer les prédictions d'erreur par pixel afin d'améliorer la localisation et la segmentation de l'objet d'intérêt. Notre méthode permet de segmenter de façon homogène les instances et de supprimer le bruit en arrière-plan comparé aux autres méthodes.

Enfin, nous intégrons un ObsNet à une LearningCar pour construire, à travers ce démonstrateur, une application de nos résultats de notre recherche. Nous montrons comment intégrer un réseau observateur et un réseau de segmentation dans un système embarqué en temps réel avec des ressources limitées.

Mots clés Réseau de Neurones Profonds, Réseaux Observateurs, Segmentation d'Images, Détection d'Objets Hors Distribution, Sécurité, Conduite Autonome.

Abstract

Deep neural networks (DNNs) are involved in the decision making process of automated cars where lives are at stake even if their predictions are not always reliable. Automated cars use Deep Learning models to gather a representation of their environment, *i.e.*, where are the pedestrians, where the motorcycle is heading to, or which color is the traffic sign. DNNs are the outcome of a complex training scheme, architecture design, and data feeding. Thus, the predictions of DNNs are, in many occasions, difficult to explain and untrustworthy.

In this thesis, we propose to improve safety of DNNs by using Observer Networks, to monitor behavior and identify abnormal events of a target neural network. We introduce an additional neural network, called ObsNet, a dedicated framework for fast and accurate anomaly detection in urban scene segmentation. This auxiliary network observes the internal activations of a target network to raise an alert when it encounters an unsafe prediction. We train the observer network on the failure of a frozen target network, leaving its predictive performance unchanged for the downstream task of image segmentation.

For our first contribution, we use an oracle prediction to guide the training of the auxiliary network. The ObsNet learns to predict a divergence based uncertainty. We highlight strong errors detection for images corrupted with artifacts like sun glare or rain.

For our second contribution, we leverage Local Adversarial Attack (LAA) to help stabilize the training when a few meaningful data can be provided to the observer. We show that the resulting ObsNet succeeds in detecting Out-of-Distribution (OoD) objects for semantic segmentation. We obtain State-of-the-Art results for OoD detection on challenging dataset, while limiting the run-time overhead.

For our third contribution, we target anomaly detection for instance segmentation. We go beyond pixel-wise error maps which do not enable an automatic decision of the system. We propose to identify precisely unknown objects in the scene. We retrieve instance masks to aggregate and filter pixels-wise error predictions to improve the localization of the object of interest. Our method better segments individual instances and removes background noise compared to other methods.

Lastly, we embedded an ObsNet on a LearningCar to build a real-time demo of our research outcomes. We show how to integrate an observer network and a segmentation network into a safety-critical system with limited resources.

Keywords Deep Neural Network, Observer Network, Image Segmentation, Out-of-Distribution Detection, Safety, Automated Driving.

Publications

This thesis draws heavily on earlier work and writing in the following papers:

Journals

- Victor Besnier, Andrei Bursuc, Alexandre Briot, David Picard (2021). Triggering Failures: Panoptic Out-Of-Distribution detection by learning with Observer Network. Under review;

Refereed conferences

- [BBPB22] Victor Besnier, Andrei Bursuc, David Picard, Alexandre Briot (2022). Instance-Aware Observer Network for Out-of-Distribution Object Segmentation. In Neural Information Processing Systems Machine Learning Safety Workshop;
- [BBPB21] Victor Besnier, Andrei Bursuc, David Picard, Alexandre Briot (2021). Triggering Failures: Out-Of-Distribution detection by learning from local adversarial attacks in Semantic Segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV);
- [BPB21] Victor Besnier, David Picard Alexandre Briot (2021). Learning Uncertainty for Safety-Oriented Semantic Segmentation in Autonomous Driving. In Proceedings of the IEEE International Conference on Image Processing (ICIP);

Talks

- Victor Besnier (2021). Failure detection for safety-oriented semantic segmentation in autonomous driving with Observer Networks. Société des Ingénieurs de l'Automobile (SIA);

Artwork 1: Safe automated cars in the future. DreamStudio, image generated from Stable Diffusion [RBL+21] model.

Contents

List of Figures

List of Tables

Part I

INTRODUCTION

1

OoD for Image Segmentation

Chapter 2 ➤

Synopsis Unlike humans, Deep Neural Networks (DNNs) fail at predicting what they have never seen. Looking at the softmax layer values of a DNN does not provide relevant information about the certainty of the predicted class. Furthermore, even though neural networks perform as expected on a given dataset, their performance might decrease when we test our model in the wild. In the following sections, we will present the background of the thesis. First, we will explain why the reliability of DNNs should be considered when they are integrated into safety-critical applications like automated driving (AD). Then, we will introduce the source of hazardous situations through uncertainty and Out-of-Distribution. And finally, we will present the downstream task of my research: image segmentation.

1.1 MOTIVATION

1.1.1 *Build Safe Models for Safe Application*

In 2016, an accident involving a Tesla in Autopilot mode[2] and an overturned truck resulted in the death of the driver. The automotive system mistook the truck for a bright sky. On March 18, 2018, the first pedestrian died involving an autonomous car. Elaine Herzberg was hit by an Uber test vehicle in Arizona. According to James Arrowood, a lawyer specialized in automated cars, the sensor was unable to detect the pedestrian because she was walking next to her bike[3]. The car misinterpreted the woman as another vehicle moving away from the Uber test vehicle.

According to the Washington Post, 273 cars crashed because of Tesla's Autopilot. In most cases, there is a repeating pattern. First, an unusual situation occurs: unknown animal, pedestrian on the highway, overturned car. Then, an erroneous prediction of the perception system: for example, confusing an animal with a pedestrian or a bicycle with a motorcycle. Finally, an erroneous behavior of the system occurs: the car does not brake or does not avoid the obstacle.

1.1.2 *Uncertainty in DNNs*

One reason why DNNs are not reliable is because modern networks do not properly model confidence in their prediction. Even worse, deep architectures tend to produce overconfident predictions Figure 1.1.

[2] Advanced driving assistance for line-tracking, self-park, or autonomous navigation in constrained environment.

[3] full article available at https://eu.azcentral.com/story/money/business/tech/2018/03/22/what-went-wrong-uber-volvo-fatal-crash-tempe-technology-failure/446407002/

In other words, the predictive score, *i.e.*, maximum of the softmax layer, does not correspond to a calibrated score. Among several reasons, we can cite three main ones. First, the cross-entropy loss penalizes fuzzy predictions; the model tends to predict either one or zero for binary problems or a Dirac for multi-class tasks. Secondly, the high network capacity increases its ability to overfit on the data. Indeed, recent advance in DNNs such as residual connections, batch normalization, increased depth, and other artefacts to better scale DNN up, will ultimately increase the overfitting of the data. Moreover, the softmax output is a smooth approximation of the argmax function and converges to a very sharp prediction. Finally, longer training and well crafted training heuristics (*i.e.*, data augmentation, learning rate recipes) push further the confidence on their prediction.

Looking at Figure 1.3, we can see an example of image classification that cannot be solved even with the best models. For example, DNNs cannot predict on which side a coin will fall, because it is a random process with 1/2 chance for tail and 1/2 change for head. As shown in Figure 1.2, Deep Learning performance has not improved significantly over the last three years on ImageNet [DDS$^+$09]. There are still almost 10% of misclassified images. We can ask ourselves if we will be able to achieve 100% accuracy on this dataset, but the answer is probably no. For many computer vision tasks, overconfidence is problematic when we cannot guarantee the right result.

One of the reasons neural networks cannot work perfectly on many tasks is hidden in the data. Some data contain more ambiguity than others and confident prediction cannot be given with these data. Data ambiguity and overconfidence implies an unreliable prediction, which can cause a crash.

To better understand where data ambiguity comes from, we need to introduce uncertainty. We can point to at least three sources of uncertainty: Aleatoric, Epistemic [DKD09] and Out-of-Distribution.

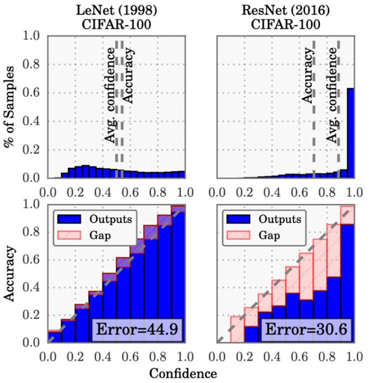


Figure 1.1: **Models over-confident**: Recent architecture of Deep Neural Network are more confident than old network, even when it is wrong, credit [GPSW17]

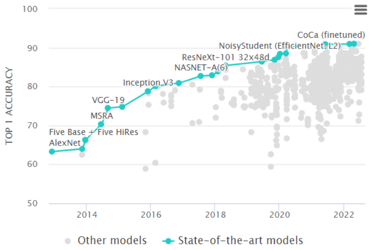
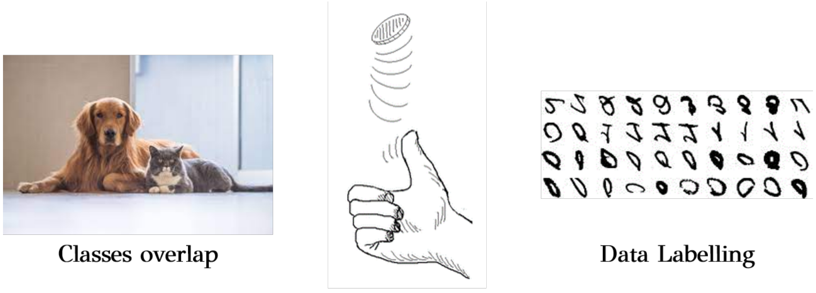


Figure 1.2: **State-Of-The-Art on ImageNet** After more than 10 years of improvement, the SOTA accuracy stabilize around 90% over the last three years.


Classes overlap

Random Process

Data Labelling

Figure 1.3: **Ambiguity in the data**: **Classes overlap**: For classification, this image is not predictable because the model must predict either the cat or the dog, but not both. **Random Process**: Heads or tails? No model can predict this, it is a random process. **Data Labeling**: The input data may be ambivalent. No clear label can be assigned.

Epistemic is the uncertainty in the model parameters. It captures what we do not know about the process that generates the data. This uncertainty can be explained if there is enough data. In other words, the more data you get, the more you reduce the epistemic uncertainty of the model. In fact, this uncertainty is even more important to consider of when the dataset is small. For safety-critical applications, epistemic uncertainty is about understanding examples that rarely appear in the training set.

Aleatoric is the uncertainty in the data. This uncertainty cannot be reduced, even with a lot of data, but can be learned as an output of a model. Moreover, this uncertainty does not increase for situations far from the learning distribution, unlike Epistemic uncertainty.

Aleatoric uncertainty can be divided into two categories.

- **Heteroscedastic**: An input-dependent uncertainty where one input may contain more than another. Occlusion, sun glare or no visual patterns are considered heteroscedastic uncertainties.
- **Homoscedastic**: A constant noise in the input data, regardless of the content of the input. The quality of a camera or the quantization of an image are good examples of homoscedastic uncertainty.

This uncertainty is useful when we have a large amount of data, *i.e.*, when the model uncertainty is small. So far, this uncertainty is also lighter to compute than the other uncertainties as no Bayesian inferences are necessary.

Out-of-Distribution (OoD) captures the mismatch between the training distribution and the test distribution. More specifically, an OoD example is a data belonging to a class unknown to the perception system, *i.e.*, a class that is not defined or present in the training data. In the open world setting, many objects are not included in the offline training set, and thus fall into the OoD category. For example, training a model to distinguish dogs from cats and testing the model on an image of a fish. The model will fail to predict the correct class because it has never seen a fish during a train time.

1.1.3 *Image segmentation*

In the prelude, we briefly presented the functioning of a deep neural network for classification, and the basis for training it. In the context of our work, we focus on the task of dense image (pixel) classification: image segmentation. The objective of this task is to classify every single pixel within the image in order to obtain a segmented image. Historically, classical computer vision approaches - such as thresholding or image clustering - could work for basic tasks, but failed to segment complex scenes. With the advent of DNNs, the task has considerably improved. We show below a brief review of past and present methods to perform image segmentation with neural networks.

First, image segmentation can be applied under three different granularities, as shown in Figure 1.4:

- Semantic segmentation: it consists in associating each pixel of the image with a semantic class. In a driving scene context, all cars belong to the same class, as does each pedestrian or any other class.

- Instance segmentation: it segments only the entities in the image, but unlike semantic segmentation, instance segmentation dissociate each different instance when classifying the pixels in the scene. Two different pedestrians will be assigned to two different prediction masks.

- Panoptic segmentation: it is a combination of the two previous tasks, where we decompose the image into countable *things* (cars, pedestrians, etc.) and uncountable *stuff* (skies, buildings, etc.). Unlike semantic segmentation, each predicted *thing* is defined by its identification to distinguish different instance of the same class.

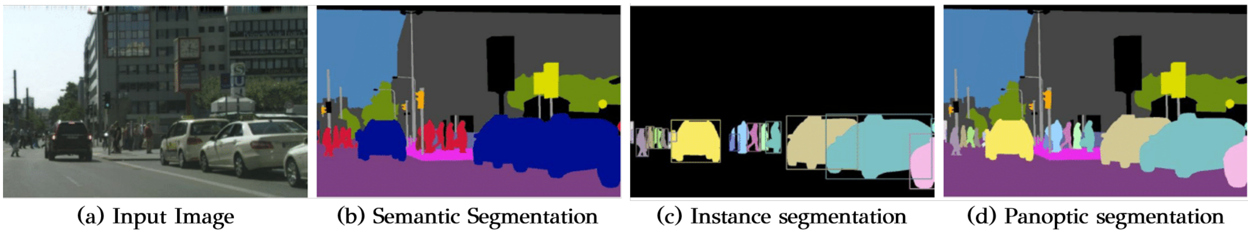

(a) Input Image (b) Semantic Segmentation (c) Instance segmentation (d) Panoptic segmentation

Figure 1.4: **Image Segmentation** (a) the input image, (b) the semantic segmentation where every pixels are classified, (c) the instance segmentation where each instance is localized and segmented, and finally, (d) the panoptic segmentation which fuse the two previous tasks. credit [CWL+20]

There are plenty of datasets for image segmentation. The most popular datasets are PASCAL VOC [EGW+10], MS COCO [LMB+14] or ADE20K [ZZP+17] for common object segmentation. For on-the-road real image segmentation, CityScapes [COR+16] with 5000 images annotated with fine grained annotation and 20000 images with coarse annotations; WoodScapes [YHH+19] with 10000 fisheye images from 4 point-of-view around the car; and two much smaller datasets: CamVid [BFC08] 600 and KITTI [GLU12] 400 images. We can also notice synthetic datasets extract from game engine. SYNTHIA [RSM+16] and GTA V [RVRK16] datasets, compose of, respectively, 9400 images and 24966 images. Medical image segmentation is also a growing area of computer vision thanks to DRIVE [SAN+04] or Ksavir-SEG [JSR+20]. In the following paragraphs, we introduce the most well known architectures to solve these task.

FCN [LSD15] Long et al. were the first to introduce fully convolutional networks (FCNs) for image segmentation, see Figure 1.5. The initial intention was to adapt classification networks such as AlexNet [KSH12], VGG [SZ14] or GoogleNet [SLJ+15] to dense prediction tasks. Their main contribution is to use a de-convolution layer (transpose convolution) for upsampling. In addition, they exploit the skip connection to combine deep and shallow semantic information. Despite the simplicity of the approach, the architecture has significantly improved the state of the art in 2015. However, this simple approach has limitations in two important areas:

- Inconsistent label prediction on large objects. For objects larger than the receptive field, the prediction of the segmentation network is splitted, and may be inconsistent.
- Loss of spatial information due to reductive convolution of the first layer. The model struggles to predict an accurate segmentation map for small objects.

DeconvNet [NHH15] Noh et al. take the concept of de-convolution a step further by constructing a complete decoder network consisting of de-convolution, un-pooling, and ReLU activation to obtain a more refined prediction. Similar to SegNet [BKC17], DeconvNet records the location of the maximum activation of the de-convolution layer to recover the pooling operation in the encoder part of the network. The encoder-decoder (Figure 1.6) architecture results in a smoother and denser prediction than FCN.

U-Net [RFB15] The U-net paper introduces a hierarchical decoder to retrieve different levels of features. The frameworks achieve outstanding results in 2015 for most semantic segmentation challenges. Like FCN, Ronneberg et al. also use skip connection to fuse spatial and semantic information. But unlike FCN, they use concatenation instead of addition to merge the feature, see Figure 1.7. In addition, they use boundary weighted loss to improve boundary segmentation.

PsPNet [ZSQ+17] Zhao et al. introduce the Pyramid Scene Parsing (PSP) network to retrieve the global prior context. They exploit the pyramid clustering module to deal with objects of different sizes. The main idea is to cluster the input feature map at N different scales and perform a 1x1 convolution. Finally, the resulting features are concatenated and merged by convolution (Figure 1.8). PSPNet uses a ResNet [HZRS16] architecture for the main part of the network.

DeepLab v1, v2, v3, v3+ [LCPK+15, CPK+16, CPSA17, CZP+18] Starting in 2015, Chen et al. note that most encoders in segmentation were from the classification network. But these architectures, iteratively downsample the image resolution to get a single value per image. Unlike classification, the problem of segmentation requires to keep as much spatial information as possible. Thus, DeepLab v1 [LCPK+15] introduced a fully connected Conditional Random Field at the top of the last layer (Figure 1.9) for finer results. Deeplab v2 [CPK+16] incorporates dilated convolution, also known as convolution a-trous, as does DilatedNet [YK16]. The idea is to modify the kernel of the convolution with stride. The remaining convolution increases the receptive field without decreasing the input resolution. Deeplab v2 exploits successive a-trous convolutions as a cascading

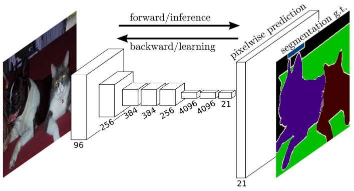


Figure 1.5: **Fully Convolutional Networks (FCN):** One of the first neural network architecture for efficient semantic segmentation.

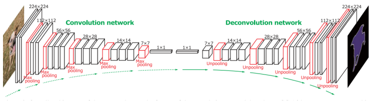


Figure 1.6: **DeconvNet**: Encoder-Decoder architecture with de-convolution layers for dense and smooth prediction.

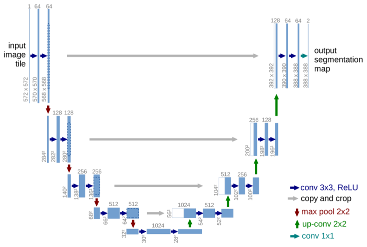


Figure 1.7: **U-Net**: The decoder re-uses the feature computed to symmetric layer of the encoder to reconstruct the different levels of shape details.

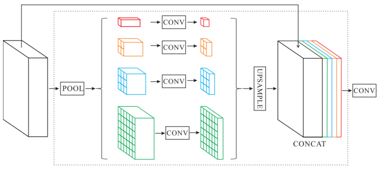


Figure 1.8: **Pyramid Pooling module**: illustration of the pooling at different scale.

context module. DeepLab v3 [CPSA17] reused PSPNet's Spatial Pyramid Pooling in addition to Deeplab v2. Finally, the latest version, *i.e.*, Deeplab v3+ [CZP+18] improves the decoder part with more layers, and residual connections from early and intermediate layers of the encoder. Deeplab is currently one of the most competitive architectures, while conceptually simple.

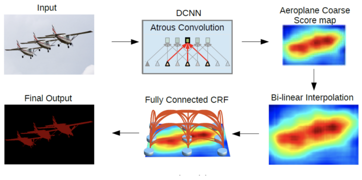


Figure 1.9: **Deeplab v1**: conditional random fields help to restore fine details after the last upsampling.

Transformer-based [SGLS21, XWY+21] Since [VSP+17], the Transformer architecture has broken through the field of natural language processing (NLP). In [DBK+21], authors show that this architecture could be transferred to the computer vision tasks as well. The attention mechanism allow the receptive field to be much more flexible and enable to gather information about objects adaptively from different corners of the image, beyond the rigid rectangular receptive field from convolutions. Modern image segmentation problems require robustness on object size. Quickly, segmentation research benefited from this framework. Transformer-based segmentation networks use [DBK+21, LLC+21, BDPW22] as the main part of the encoder. [SGLS21] extend ViT for segmentation. They use the output embedding of each patch in the image with a ViT, and feed then in a Vit decoder that processes the patch/token features computed by the encoder. The decoder is enriched with multiple class tokens corresponding to the classes to be segmented. [XWY+21] use a transformer architecture as encoder and a lightweight multilayer perception (MLP) for the decoder. Using an MLP, SegFormer [XWY+21] allows combining local and global attention for a better scene representation.

Instance segmentation Here we briefly discuss the main architecture dedicated to instance segmentation. The initial work is based on object detection, first we predict a bounding box around each object of interest, then we predict the class of the object, and finally we segment the instance inside the box. Similar to object detection, there are two types of methods, one step like Yolo [RDGF16, RF17, RF18][1], SSD [LAE+16] and others [CMS+20, ZSL+21], and two steps such as Faster R-CNN [Gir15, RHGS17][2]. The first one aims to predict directly from the image, while the second one predicts first the region proposal and then the class object. For instance segmentation, most methods are an adaptation of the corresponding architecture with an auxiliary branch to compute the segmentation mask of the predicted object [HGDG17, FSW+19, WKS+20, WZK+20][3].

Panoptic segmentation A new and more challenging task has emerged from semantic and instance segmentation: panoptic segmentation [KHG+19][4]. The paradigm is more comprehensive for scene understanding because it localizes the instance object as well as the associated category of objects. In the literature, we denote three different methods, illustrated in Figure 1.10:

- **Box-based**: fusion of the bounding box around each instance and

[1] [RDGF16] Joseph Redmon et al., *You only look once: Unified, real-time object detection*. In CVPR 2016.

[2] [Gir15] Ross Girshick. *Fast r-cnn*. In ICCV2015

[3] [HGDG17] Kaiming He et al., *Mask R-CNN*. In ICCV 2017.

[4] [KHG+19] lexander Kirillov et al., *Panoptic segmentation*. In CVPR 2019.

the segmentation map [QCY20].

- **Center-based**: regression of the center coordinate of each instance and an offset map to aggregate the segmentation map per instance [WZG+20].

- **Box/Center Free**: directly predict a mask for each instance and stuff category, where each stuff category are considered without id [WZA+21].

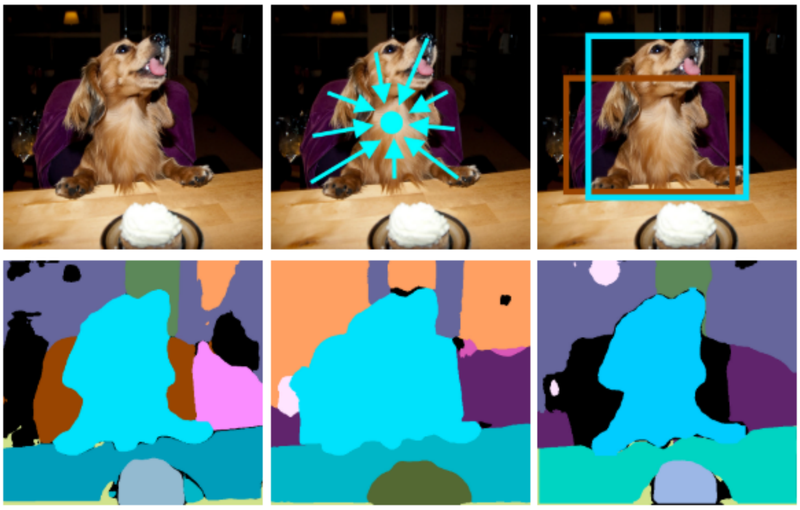


Figure 1.10: **Comparison of framework**: on the left, box/center free; at the middle, center-based; and on the right, box based. credit [WZA+21]

Early attempts to solve this task were to adapt or combine semantic segmentation and instance [KGHD19, CCZ+20] architectures. Naturally, most recent architectures integrate a transformer to boost the performances [CSK21, CMS+21].

1.2 SAFETY FOR AUTOMATED DRIVING

Synopsis In the late 1980s, General Motors developed the first electronic control unit (ECU) for fuel ignition. Since, ECU device's have multiplied in modern cars making the vehicles more and more autonomous. These devices can control the speed, and the steering wheel. The purpose of safety is to eliminate or mitigate the risks associated with automated systems. Traditional safety can deal with systematic failures or unexpected behavior of the car. But with the advent of machine learning, specifically deep learning, in controlling the car, traditional safety concepts, practices and standards must adapt to this specific type of algorithms. In this section, we will introduce four different safety standards to give context to them. From traditional safety such as ISO 26262 to more recent standards specifically addressing artificial intelligence and autonomous driving.

1.2.1 *ISO 26262: Systematic and random failure*

Road vehicles - Functional safety [ISO19] is one of the pillars of traditional safety in the automotive industry. The main objective is to reduce the risk due to hazards caused by malfunctioning electronic or electrical systems. For example, an airbag that inflates without a collision or an unwanted release of energy from the device that can lead to an explosion. The safety objectives of the ISO 26262 standard are therefore to put locks in place to prevent those risks. In this standard, the risks come from two main events. Systematic failures such as software bugs and random failures such as hardware aging. The ISO 26262 standard proposes to use a V-cycle (Figure 1.11), from the requirement to the validation of the system. It proposes a quantitative objective to be reached in order to reduce the risks and increase the safety of the vehicle.

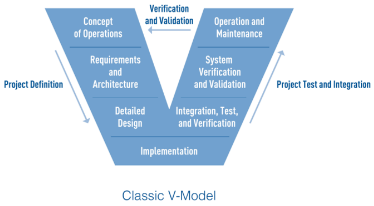

Verification and Validation
Concept of Operations
Operation and Maintenance
Requirements and Architecture
System Verification and Validation
Project Definition
Project Test and Integration
Detailed Design
Integration, Test, and Verification
Implementation
Classic V-Model

Figure 1.11: **ISO 26262 V-Cycle**: mainly focus on avoiding systematic failures.

1.2.2 *ISO 21448: SOTIF, Unexpected behavior*

The Safety Of The Intended Functionality (SOTIF) [21419] complements the ISO 26262 standard. It focuses on external events (outside the system) that are not properly handled by the system, *e.g.* weather conditions, road infrastructure, driving domains, user driving tasks, road users, car communication, car position, *etc.* SOTIF divides hazards into four distinct area:

- **Area 1 Known-safe**: the system knows that the driving scenario is safe for the vehicle and the environment.
- **Area 2 Known-unsafe**: the system knows that the vehicle is in a dangerous situation.
- **Area 3 Unknown-safe**: the system is not aware that the situation is safe for the car.
- **Area 4 Unknwon-unsafe**: the system is completely blind. The car does not know that the driving scenario is unsafe.

The objective of the SOTIF activities is to evaluate the potentially hazardous behaviors present in situations 3 and 4 and to demonstrate that the resulting residual risk is reasonable. It is expected that the residual risk will be reduced (*i.e.*, domains 2 and 3) and confidence in safety will be increased by the growth set of scenarios in domain 1. To do this, the [21419] proposes to collect as relevant data as possible to cover a sufficient number of situations.

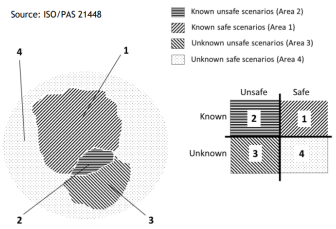

Source: ISO/PAS 21448
Known unsafe scenarios (Area 2)
Known safe scenarios (Area 1)
Unknown unsafe scenarios (Area 3)
Unknown safe scenarios (Area 4)
4
1
2
3
Unsafe
Safe
Known
Unknown
2
1
3
4

Figure 1.12: **SOTIF objective**: the main goal of the SOTIF is to reduce the Area 4: Unknown-unsafe scenario.

1.2.3 *ISO/TR 4804: SAFAD, automated car safety*

The Safety First for Automated Driving (SAFAD) [SAF19] report is the first attempt to address safety, autonomous driving and AI (Figure 1.13). Specifically, they raised four challenges, which need to be understood during implementation. First, deep neural networks cannot detect unknowns. Second, DNNs are overconfident about the input and output of the distribution. Third, DNNs do not necessarily base their decisions on semantically meaningful features. And last, DNN predictions are brittle and can flip under minor changes to the input distribution. In addition, the security requirements (reliability, robustness, temporal stability, criticality of certain types of errors, *etc.*) are not integrate on mainstream losses such as the cross-entropy. To remedy this, they made several propositions to monitor/analyze/verify the input data, (*i.e.*, Check for distributional changes in the operational domain, Search for a new object (OoD), Check for changes in the world, *etc.*) and on model behavior (*i.e.*, Use an observer to track unexpected behavior). This last proposal significantly influences our conception of the observer network in Chapter 2.

The two previous standards do not directly address to artificial intelligence (AI) and more specifically machine and deep learning. They do not explicitly consider the application of machine learning:

- They do not address data collection.
- No evaluation measures for machine learning applications.
- They do not address uncertainty.

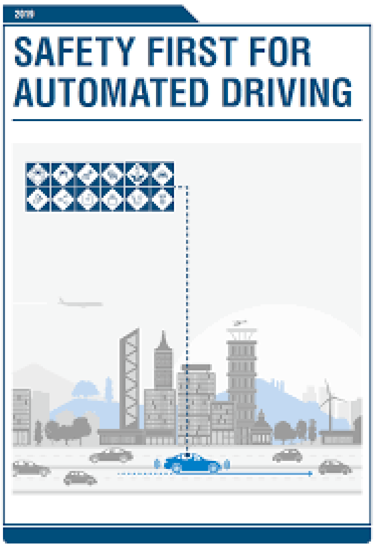


Figure 1.13: **SAFAD**: first attempt to standardise safety and artificial intelligence.

1.2.4 *Safety in machine learning*

In the machine learning and computer vision community, how to achieve trustworthy and robust neural networks is a major research question [RSG16]. When deploying a model for safety-critical application such as autonomous cars, we do not only consider the final accuracy of the model. We should also consider what are the minimal requirements for each car to satisfy. Moreover, as the goal is to spread these technologies for a large population, we must deal with the scalability to millions of cars [SSSS17, MGK^{+}17]. How to ensure that the autonomous system does not fail when the environment changes from the training distribution? Amodei et al. [AOS^{+}16] propose a list of five practical research problems to ensure to mitigate accident risk:

- Maintaining the surrounding environment as it is.
- Avoiding hacking the problem, by trivial sub optimal solution.
- Adaptation to scalability problem.
- Carefully explore the surrounding world.
- Safe behavior in unknown situations.

Our work focuses mainly on the last point, that is to say, how to construct models able to detect unknown situations for a safer behavior.

1.3 RELATED WORKS

Synopsis Deep Neural Networks (DNNs) are not always reliable. DNNs do not know that they do not known and are not as robust as *humans* are, for instance, these models seem to be biased over texture, regardless of the shape of the object [GRM+19][1]. The problem of data samples outside the original training distribution has been long studied for various applications before the deep learning era, under slightly different names and angles: outlier [BKNS00], novelty [SWS+00], anomaly [LTZ08] and, more recently, OoD detection [HG17, HMD18][2]. In the context of widespread DNN adoption this field has seen a fresh wave of approaches based on input reconstruction [SSW+17, BWAN18, LNFS19, XZL+20], predictive uncertainty [GG16, KG17, MG18a], ensembles [LPB17, FBA+20], adversarial attacks [LSL18, LLLS18], using a void or background class [RHGS15, LAE+16] or dataset [BKOŠ19, HMD18, MG18a], *etc*. In the following subsections, we dig into the main category of methods for Out-of-Distribution detection. We split related work into three categories: Architecture Based, Data Based and task specific.

1.3.1 *Architecture Based*

Here, we present methods that rely on specific training schemes or dedicated model architectures, independent of the data statistics.

Baseline *i.e.*, softmax In spite of the overconfidence pathological effect, using the maximum class probability from the softmax prediction can be used towards OoD detection [HG17, ORF20]. Temperature scaling [GPSW17, P+99][3] is a strong post-hoc calibration strategy of the softmax predictions using a dedicated validation set. If predictions are calibrated, OoD samples can be detected by thresholding scores. Pre-training with adversarial attacked images [HLM19] has also been shown to lead to better calibrated predictions and good OoD detection for image classification. We consider this framework as a simple baseline, because for advanced problems, accuracy and calibration on i.i.d does not transfer well on OoD detection, as explained in [OFR+19]. One can also mitigate overconfidence by modifying the cross-entropy loss [LGG+17]. In [AC20], Faruk et al. use an auxiliary task and use the consistency of prediction as OoD detector. In addition, we can moderate overconfidence with a modified softmax prediction [Neu18, MSK18] or modified cross entropy [RLF+19].

Bayesian Neural Networks (BNNs) Bayesian approaches [Gra11, Nea12, BCKW15][4] can capture predictive uncertainty from distributions learned over network weights, but do not scale well [DJW+20] and approximate solutions are preferred in practice. Variationnal Inference of BNNs can be improved [LW16, LW17], done deterministicly [WNM+19] in order to better scale to modern architec-

[1] [GRM+19] Robert Geirhos et al., *Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness*. In ICLR 2019.

[2] [HG17] Robert Geirhos et al., *Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness*. In ICLR 2019.

[3] [GPSW17] Dan Hendrycks et al., *A baseline for detecting misclassified and out-of-distribution examples in neural networks*. In ICLR 2017.

[4] [BCKW15] Charles Blundell et al., *Weight uncertainty in neural networks*. In ICML 2015.

tures and datasets [HLA15, ZSDG17, WT11]. Mixture Density Networks [CLLO18] can also learn to measure uncertainty without heavy sampling method as well as [HMK+20] for recurrent neural networks (GRUs) or free epistemic estimation [PFC+19].

Deep (Pseudo-)Ensemble Deep Ensemble (DE) [LPB17][1] is a highly effective, yet costly approach, that trains an ensemble of DNNs with different initialization seeds. Deep Ensembles work well in practice because they explore entirely different modes [FHL19]. Distillation for Deep Ensemble can reduce the run time required [MMG20] at inference. Pseudo-ensemble approaches [Gal16, MIG+19, MAG+20, KSW15] are a pragmatic alternative to DE that bypass training of multiple networks and generate predictions from different random subsets of neurons [Gal16, SHK+14] or from networks sampled from approximate weight distributions [MIG+19, FBA+20, MAG+20]. Swapout [SHF16] introduces stochastic training that can be used in practice for OoD detection as well as [TAS18] that leverage batch normalization for uncertainty measurement. However they all require multiple forward passes and/or storage of additional networks in memory.

Prior Networks In [MG18a][2], the authors decide to directly emulate an ensemble by parameterizing the output predictions of a model with a Dirichlet distribution in order to model uncertainty. They separate model uncertainty, data uncertainty and distributional uncertainty (OoD). A Deep Prior Networks (DPNs) is trained to predict a strong or flat a priori distribution on the expected sample by minimizing a KL objective. Later, [MG19] found that the reversed KL has a better property and helps to stabilize the training. Some researchers use normalizing flow networks [CZG20] or GANs [SKCS20] to simulate OoD data. But for more complete problems, Prior networks still need an explicit dataset to be train on, in order to predict a flat distribution.

1.3.2 *Data Based*

In this subsection, we describe methods that detect OoD from the statistics of the input data, or through internal activations of a pretrained network.

Reconstruction In semantic segmentation, anomalies can be detected by training a (usually variational) autoencoder [CM15, BWAN18, VPSM20, BBL+19][3] or generative model [SSW+17, LNFS19, XZL+20] on in-distribution data. OoD samples are expected to lead to erroneous and less reliable reconstructions as they contain unseen patterns during training. On high resolution and complex urban images, autoencoders under-perform while more sophisticated generative models require large amounts of data to reach robust reconstruction or rich pipelines with re-synthesis and comparison modules. Flows based

[1] [LPB17] Balaji Lakshminarayanan et al., *Simple and scalable predictive uncertainty estimation using deep ensembles*. In NeurIPS 2017.

[2] [MG18a] Andrey Malinin et al., *Predictive uncertainty estimation via prior networks*. In NeurIPS 2018.

[3] [BBL+19] Andrea Borghesi et al., *Anomaly detection using autoencoders in high performance computing systems*. In AAAI 2019.

approach uses normalizing flows [HCS+19, KD18] to compute the true distribution of the data $p(x)$ and use it to detect OoD. Nevertheless, these methods do not seem to be very effective in practice [NMT+19].

Adversarial attacks In ODIN, Liang et al. [LSL18][1] leverage temperature scaling and small adversarial perturbations on the input at test-time to predict in- and Out-of-Distribution samples. Lee et. al [LLLS18] extend this idea with a confidence score based on class-conditional Mahalanobis distance over hidden activation maps. Both approaches work best when train OoD data is available for tuning, yet this does not ensure generalization to other OoD datasets [SSL19]. Moreover, so far this method has not been shown effectively for structured output tasks where the test cost is more likely to explode, as adversarial perturbations are necessary for each pixel.

Features statistics Another approach is to seek into the activation of a pre-trained neural network [BMN21] statistics which differ when facing an Out-of-Distribution sample. In industrial defect, CutPaste [LSYP21] uses data augmentation of small patch at to detect anomalies during test. And PatchCore [RPZ+22][2] use a memory bank of patch, extracted from pre-trained network features to compute a distance between training samples and testing samples.

Learning errors Inspired by early approaches from model calibration literature [P+99, ZE01, ZE02, NCH15, NC16], a number of methods propose endowing the task network with an error prediction branch allowing self-assessment of predictive performance. This branch can be trained jointly with the main network [DT18, YK19], however better learning stability and results are achieved with two-stage sequential training [CTBH+19, HDVG18, BPB21, SvNB+19a][3].

1.3.3 *Task specific*

Most of the previous was applied for classification, but does not always transfer well for image segmentation or 2d object detection. We present here, methods that are dedicated for 2d object detection and image segmentation.

2D detection For 2D OoD detection, VOS [DWCL22][4] synthesises virtual outliers to regularize the boundary prediction of the detector. Some approaches try to adapt well-known frameworks to increase the reliability of them [CCKL19]. CertainNet [GHM+22] separate uncertainties for each output signal: objectness, class, location and size. Du et al. [DWGL22] use video in the wild to train an uncertainty branch and detect OoD.

Segmentation Specific to segmentation, [DBBSC21] use image reconstruction from the segmentation prediction and an additional

[1] [LSL18] Shiyu Liang et al., *Enhancing the reliability of out-of-distribution image detection in neural networks*. In ICLR 2018.

[2] [RPZ+22] Karsten Roth et al., *Towards total recall in industrial anomaly detection*. In CVPR 2022.

[3] [CTBH+19] Charles Corbière et al., *Addressing failure prediction by learning model confidence*. In NeurIPS 2019.

[4] [DWGL22] Xuefeng Du et al., *Towards unknown-aware learning with virtual outlier synthesis*. In ICLR 2022.

dissimilary network to detect OoD. Mukhoti et al. develop a bayesian DeepLab [MG18b] with dropout. In [SvNB+19b], the authors use a *gambling networks* to improve and robustifyed a segmentation networks. Chan et al. [CRG21a] leverage OoD dataset to train the segmentation network to maximize the entropy on these samples while minimize it on in-distribution examples. DenseHybrid [GBv22] fuses the data posterior and the data likelihood from discriminative network to segment OoD by learning from negative data.

1.4 DATASETS AND METRICS

1.4.1 *Datasets*

In the literature, early methods focused on classification where the target model was trained on a dataset and tested on a completely different OoD dataset. However, this framework could introduce hazards that come not only from the novelty of the class but also from the data registration process [AC20]. In this thesis, we focus on OoD detection for image segmentation. The task is more challenging and less studied. In image segmentation, an OoD object may cover only a small part of the image while the rest of the image remains in the distribution. In recent years, datasets have been constructed to leave out-of-field objects only on the test set. In semantic segmentation we can denote multiple datasets.

CamVid OoD: We design a custom version of CamVid [BFC08][1], where we blit random animals from [MATTEP08] in a random part of the image. This dataset contains 367 train and 233 test images. There are 19 different species of animals, and one animal in each test image. This setup is similar of Fishyscapes [BSN+19] but allow users to quickly test their methods without been stuck to specific deep learning framework[2].

StreetHazards: This is a synthetic dataset [HBM+19][3] from the Carla simulator. It is composed of 5125 train and 1500 test images, collected in six virtual towns. There are 250 different kinds of anomalies (like UFO, dinosaur, helicopter, etc.) with at least one anomaly per image.

BDD Anomaly: Composed of real images, this dataset is sourced from the BDD100K semantic segmentation dataset [YCW+20][4]. Here, motor-cycles and trains are selected as anomalous objects and all images containing these objects are removed from the training set. The remaining dataset contains 6688 images for training and 361 for testing.

SegmentMeIfYouCan: Appeared recently with a public score-boards, available to compare all OoD methods. The dataset [CLU+21][5] is decomposed on two different tracks: RoadAnomaly21 and Road-Obstacle21. The first one contains 100 anomalies on the road such as a plane on the highways or an elephant in front of the car, see Figure 1.14 and represents real situations. RoadObstacle21 contains

[1] [BFC08] Gabriel J. Brostow et al;, *Semantic object classes in video: A high-definition ground truth database*. In PRL 2008.

[2] TensorFlow models are mandatory to test on the server of the Fishyscapes dataset.

[3] [HBM+19] Dan Hendrycks et al., *A benchmark for anomaly segmentation*. ArXiv 2019.

[4] [YCW+20] Fisher Yu et al., *BDD100K: A diverse driving dataset for heterogeneous multitask learning*. In CVPR, 2020.

[5] [CLU+21] Robin Chan et al., *Segment-meifyoucan: A benchmark for anomaly segmentation*. In NeurIPS Datasets and Benchmarks 2021.

371 high resolution images with small objects on the road (*e.g.* stuffed toys, sleighs or tree stumps). No training sets are provided for these datasets, with the assumption to use solely CityScapes as the train set.

Lost&Found: It [PRG+16] contains 2071 images extracted from 112 videos in the wild. Each frame contains unexpected small objects lost from cargot, where most of the objects are toys or boxes.

Fishyscapes: this is a public benchmark [BSN+19] that exploits synthetic objects blit in the CityScapes [COR+16] test dataset. The images are on the server and are updated every three months with new objects.

MVTec AD: [BFSS19], a dataset of industrial anomaly inspection. It contains more than 5000 images and fifteen classes of objects and textures, such as toothbrushes, capsules or cables.

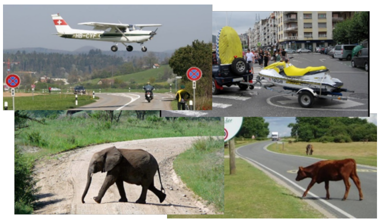


Figure 1.14: **Segment Me If You Can**: sample of the test set of the dataset for anomaly segmentation.

1.4.2 *Metrics*

Along the chapters, we compare our different methods on numerous metrics for OoD and error detection described in the following:

- **Fpr95Tpr** [LSL18]: It measures the false positive rate when the true positive rate is set to 95%. The aim is to obtain the lowest possible false positive rate while guaranteeing a given number of detected errors.

- **Area Under the Receiver Operating Characteristic curve (AuRoc)** [HG17]: This threshold free metric corresponds to the probability that a certain example has a higher value than an uncertain one.

- **Area under the Precision-Recall Curve (AuPR)** [HG17]: Also a threshold-independent metric. The AuPR is less sensitive to unbalanced datasets than AuRoc.

- **Recall at 95% Precision (R@P) [BPB21]**: We measure the recall when the precision is equal to 95% (e.g. R@P=0.95). Better uncertainty measures ought to achieve higher recall.

- **Trigger [BPB21]**: We propose a ”safety trigger rate metrics” which is the percentage of images in the dataset with a coverage of a certain prediction over a threshold. For instance ”Trigger 75%” is the percentage of images in the test set where the coverage of safe prediction is above 75%.

We also use the two following metrics for calibration:

- **Expected Calibration Error (ECE) [NCH15]**: This metric measures the expected difference between accuracy and predicted uncertainty.

- **Adaptive Calibration Error (ACE) [NDZ+19]**: Compared to standard calibration metrics where bins are fixed, ACE adapts the range of each the bin to focus more on the region where most of the predictions are made.

1.5 CONCLUSION

To sum up the above methods, we show in Table 1.1 a recap of the previous frameworks. We highlight pros and cons of each batch of methods and where the ideal methods should be fast, memory efficient, easy to train and accurate for OoD detection.

Type	Example	OoD accuracy	Fast Inference	Memory efficient	Training specification
Softmax	MCP [HG17]	-	✓	✓	No
Bayesian Learning	MCDropout [GG16]	✓	-	✓	Reduces IoU acc.
Reconstruction	GAN [XZL+20]	✓	✓	✓	Unstable training
Ensemble	DeepEnsemble [LPB17]	✓	-	-	Costly Training
Auxiliary Network	ConfidNet [CTBH+19]	-	✓	✓	Imbalanced train set
Test Time AA	ODIN [LSL18]	-*	-	✓	Extra OoD set
Prior Networks	Dirichlet [MG18a]	✓	✓	✓	Extra OoD set

Table 1.1: **Summary of various OoD detection approaches amenable to semantic segmentation.** For real-time safety, key requirements for an OoD detector are accuracy, speed, easy training and memory efficiency. *Not accurate for semantic segmentation

In the following chapters, we introduce the concept of observer network for efficient OoD detection in image segmentation. The observer network has several advantages in terms of accuracy and inference speed. First, in Chapter 2 we introduce the concept of ObsNet, then in Chapter 3 and Chapter 4 we present two different ways to train an observer. Then, in Chapter 5, we adapt the method for the segmentation of instances and panoptic objects. Finally, in Chapter 6, we show how to integrate an observer on a demonstration car and test it in real conditions.

Part II

LEARNING WITH OBSERVER NETWORKS

2

Observer Network

◂ Chapter 1 Chapter 3 ▸

Synopsis In this Chapter, we introduce the concept of observer networks (ObsNet). This class of method is useful for error, Out-of-Distribution (OoD) detection and uncertainty measurement. The main idea is to use an additional network in parallel with the target network. The auxiliary network, ObsNet, observes internal activations of the frozen, pre-trained target network. ObsNet is trained on the failures of the target network. In the following sections we present the motivation, the architecture and the challenge to train an ObsNet. We show the characteristics of this framework for classification on Cifar10 and Cifar100.

2.1 MOTIVATION

In the context of safety-critical applications such as autonomous driving, we do not only seek for the accuracy of the primary task. We also need to have a reliable prediction that can guarantee a safe behavior of the automated system. Moreover, the solution must be incorporated in a real-time embedding setting. To establish the ideal framework, we have to focus on the four following milestones:

- **Accuracy**: Primary task (*e.g.* classification, segmentation, detection) should stay as good as possible, for the appropriate functioning of the car.
- **Reliability**: Prediction of the primary task must provide a confidence measurement, to avoid unsafe situations.
- **Memory footprint**: Embedded applications have high memory constraints, thus the methods should limit the memory usage.
- **Run time efficiency**: Real-time solution must be fast, and preserve a high frame rate for real-time application.

2.1.1 *Trade-off between accuracy and trustworthy*

In the literature, methods that deal with uncertainty, OoD or error detection are either accurate or slow. Ensemble [LPB17][2] or pseudo-ensemble [GG16][3] are known to be accurate, they perform multiple forwards passes and use the entropy of the prediction as the uncertainty measurement. But these methods are memory and computational cumbersome, as most embedded hardware ar-

[2] [LPB17] Balaji Lakshminarayanan et al., *Simple and scalable predictive uncertainty estimation using deep ensembles*. In NeurIPS 2017.

[3] [GG16] Yarin Gal et al., *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*. In ICML 2016.

chitecture cannot handle multiple predictions. Test-time adversarial attacks [LSL18] also show some limitations. The backward pass to compute the adversarial example is very slow. Finally, deterministic approaches [HG17][1] are inaccurate and known to be overconfident for in and out-of-distribution samples.

2.1.2 *Observer networks*

Here, we introduce ObsNet. A new class of methods for efficient and fast errors/OoD/uncertainty detection. The main idea is to use a dedicated architecture that operates in parallel to the target network. The framework (detailed in Figure 2.1) is composed of an additional network mimicking the target network architecture. During ObsNet training, the target network is frozen. Thus we do not impact the primary task and then preserve its performance. During inference, we only perform two forward passes: one for the primary task and one for the reliability task. Then memory consumption and number of operations is *only* multiplied by two, which is small compared to ensemble and pseudoensemble. Given the input image and residual connection from intermediate activations of the target network, we give ObsNet the ability to *observe* the behavior of the target network.

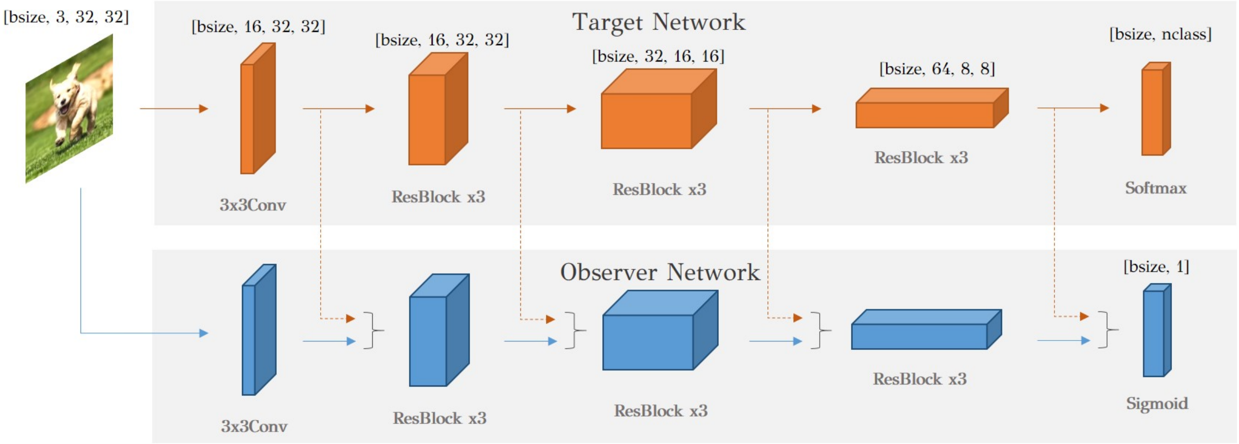


Figure 2.1: **Architecture of a vanilla ObsNet upon a ResNet18 for classification on Cifar10**. The Observer network takes as input the image and residual connection coming from intermediate activation of the Target Network. The observer is then dedicated to the error/uncertainty/OoD detection while the target network is unchanged.

To train our ObsNet, we use the failure mode of the target network. More formally, given a dataset $D = \{x, y\}$, we train ObsNet with a binary cross entropy:

$$\mathcal{L}_{Obs}(x,y) = (\mathbb{1}_{T(x)\neq y} - 1)\log(1 - Obs(x, T_r(x))) \\ - \mathbb{1}_{T(x)\neq y}\log Obs(x, T_r(x)), \quad (2.1)$$

with T the target network, T_r the residual connection from intermediate feature maps of T, Obs the Observer network and with $\mathbb{1}_{T(x)\neq y}$ the indicator function of $T(x) \neq y$. We train the auxiliary network to output 1 if the network predicts the wrong class and 0 otherwise. In other words, we give ObsNet the ability to output

[1] [HG17] Dan Hendricks et al., *A baseline for detecting misclassified and out-of-distribution examples in neural networks*. In ICLR 2017

the probability that the target network predicts the wrong class: $Obs(x, T_r(x)) \approx Pr[T(x) \neq y]$.

2.2 EXPERIMENTS

In this section, we analyze the behavior and the performance of a vanilla observer network. To better understand its characteristics, we show results and ablation for classification on Cifar10 and Cifar100. For all experiments we use a ResNet18 as the Target Network. In this chapter, we focus our work on OoD detection.

2.2.1 *Datasets, Metrics, and Compared Methods*

Many papers [LSL18, LPB17][1] evaluate Out-of-distribution on classification using two distinct datasets. Usually, the setting is not well defined, as two datasets can have hazards that do not come only from the known object but also from a different distribution or from a different recording setting. Here, we propose to leverage a different approach, similar to [AC20][2], we exclude some classes from the train set, but leave the test set unchanged. We use Cifar10 as our reference dataset. We train a ResNet18 on only 7 classes: airplane, automobile, bird, cat, deer, and dog. We then test our model on the 10 classes that include the additional: horse, ship and truck classes. These additional classes correspond to our OoD classes. This setting give us the guarantee that the process to record the data is the same for in and out of distribution. We also show results on Cifar100 where we keep the 70 first classes as in-distribution and leave the last 30 classes as the OoD.

To compare the performance of the observer network, we select four different metrics: **Fpr95Tpr** [LSL18], **Area Under the Receiver Operating Characteristic curve (AuRoc)** [HG17], **Area under the Precision-Recall Curve (AuPR)** [HG17] and **Adaptive Calibration Error (ACE)** [NDZ$^+$19].

We compare the performance of the observer network against four baselines from the related works:

- Deterministic: **MCP** [HG17]: Maximum Class Prediction. One minus the maximum of the prediction.
- Pseudo Ensemble: **MCDropout** [GG16]: The entropy of the mean softmax prediction with dropout. We use 10 forward passes for all the experiences.
- Adversarial Attack: **ODIN** [LSL18]: ODIN performs test-time adversarial attacks on the primary network. We seek the hyper-parameters *Temp* and ϵ to have the best performance on the validation set. The criterion is one minus the maximum prediction.
- Ensemble: **Deep ensemble** [LPB17]: a small ensemble of five networks. We use the entropy of the averaged forward passes.

[1] [LSL18] Shiyu Liang et al., *Enhancing the reliability of out-of-distribution image detection in neural networks*. In ICLR 2018.

[2] [AC20] Faruk Ahmed et al., *Detecting semantic anomalies*. In AAAI 2020.

2.2.2 *Results & ablation*

One strong assumption of our solution is to use residual connections from the target network. We show on 2.1 an ablation study on where to place these connections. The first interesting observation is that deeper layer seems to be more informative for ObsNet to disentangle in from out of distribution. Moreover, the best performance comes when the observer is able to watch the entire target network.

Input Layer	AuROC ↑	AuPR ↑	Fpr95Tpr ↓	ACE ↓
✓, -, -, -, -	71.69	77.93	85.92	0.2761
-, ✓, -, -, -	71.25	77.84	86.01	0.2678
-, -, ✓, -, -	75.96	80.51	81.57	0.2481
-, -, -, ✓, -	77.99	81.92	76.76	0.2331
-, -, -, -, ✓	78.19	81.03	75.61	**0.2084**
-, -, -, -, -	50.00	55.70	95.24	0.5216
-, ✓, ✓, ✓, -	78.45	82.56	77.78	0.2322
✓, ✓, ✓, ✓, ✓	**80.46**	**84.21**	**74.63**	0.2334

Table 2.1: **Ablation on activation on Cifar10.** The Target Network and the ObsNet are trained on Cifar7, leaving the horse, sheep and truck as the OoD classes. ObsNet performs best when it observes all the activations and the input image. Table also shows that deeper layers are more relevant for OoD detection than shallow ones.

Table 2.2 shows the running time and the memory consumption of our methods compared to others. Ensemble and Pseudo-ensemble are computationally demanding. Even if test-time adversarial attacks (*i.e.*, ODIN) only use few parameters, the method is slow due to the back-propagation. Finally, MCP is the fastest but inaccurate see Table 2.3. Our framework conserves good OoD detection, has a light memory footprint and is fast as they only need to compute two forwards passes.

	MCP	MCDropout	D.E.	ODIN	ObsNet
Run Time (ms)	0.01	0.107	0.045	0.038	0.019
Memory (MB)	1.046	1.046	5.230	1.046	2.089

Table 2.2: **Ablation on memory footprint and run time inference.** The ideal framework should be accurate, fast and with low additional storage. Our ObsNet is a good trade-off for these conditions.

In Table 2.3 and Table 2.4 we show results on Cifar10 and Cifar100. We can see that our ObsNet has strong OoD detection. The observer is better than baseline such as MCP or MCDropout, and close to SOTA methods like Ensemble or ODIN: 0.61% AuROC difference between ObsNet and Deep Ensemble (Table 2.3). We can also notice that our framework preserves the accuracy of the target network, which is not the case for MCDropout because dropout can hurt accuracy for convolutional networks.

2.3 CONCLUSION

We present ObsNet, a simple framework for error, Out-Of-Distribution detection or uncertainty measurement. We show that the observer can be both accurate and fast for classification on Cifar10 and Cifar100. Moreover, it does not change the architecture and the performance of the target network. We highlight that observers need all the intermediate activations of the target network to reach the best performance.

Methods	Accuracy ↑	AuROC ↑	AuPR ↑	Fpr95Tpr ↓	ACE ↓
MCP	55.89	75.76	79.90	77.47	0.2189
MCDropout	53.29	78.32	79.56	75.95	**0.2166**
D.E.	**57.48**	**81.07**	**85.22**	**69.26**	0.2193
ODIN	55.89	79.98	83.92	74.12	0.2528
ObsNet	55.89	80.46	84.21	74.63	0.2334

Table 2.3: **OoD detection evaluation on Cifar10.** The Target Network and the ObsNet are trained on Cifar7, leaving the horses, sheep and trucks as the OoD.

Methods	Accuracy ↑	AuROC ↑	AuPR ↑	Fpr95Tpr ↓	ACE ↓
MCP	50.17	80.14	82.23	76.40	0.1368
MCDropout	49.95	79.64	82.10	76.47	0.1425
D.E.	**54.25**	**84.3**	**85.30**	**71.29**	**0.1395**
ODIN	50.17	81.08	83.26	74.05	0.1361
ObsNet	50.17	82.03	83.70	74.00	0.1436

Table 2.4: **OoD detection evaluation on Cifar100.** The Target Network and the ObsNet are trained on Cifar70, leaving the last thirty classes as OoD.

However, training an observer network is not straightforward. The main point to consider is that ObsNet is trained on the errors made by the target network in the training set. But due to a natural overfitting, usually, the target network performs nearly perfectly on the training set. So, we are facing an unbalanced training set by feeding ObsNet only positive samples. In the above experiments, we perform early stopping of the target ResNet18 and leave 20.00% errors in the training set. These tricks allow the observer to gather negative examples, but could decrease the primary task performance. In the three next chapter Chapter 3, Chapter 4 and Chapter 5, we show different way to help stabilize and improve the training of ObsNet even when very few remaining errors are available.

3

Divergence Based Uncertainty

◂ Chapter 2 Chapter 4 ▸

Synopsis Chapter 2 introduces ObsNet, a simple framework dedicated to observing internal activations of a target network. Here, we aim at predicting uncertainty in semantic segmentation. We propose an efficient way to train ObsNet by learning divergence based uncertainty. We introduce a new uncertainty measure based on disagreeing predictions as measured by a dissimilarity function. Unlike distillation methods which train a student network to jointly perform the task and estimate uncertainty, we propose to estimate only this uncertainty by training the observer in parallel to the task-specific network. Using self-supervision, we allow ObsNet to predict both epistemic and aleatoric uncertainties. We show experimentally that our proposed approach is much less computationally intensive at inference time than competing methods (*e.g.*, MCDropout), while delivering better results on safety-oriented evaluation metrics on the CamVid dataset (Figure 3.1). This chapter is based on our paper "Learning Uncertainty for Safety-Oriented Semantic Segmentation in Autonomous Driving" in Proceedings of the IEEE International Conference on Image Processing (ICIP).

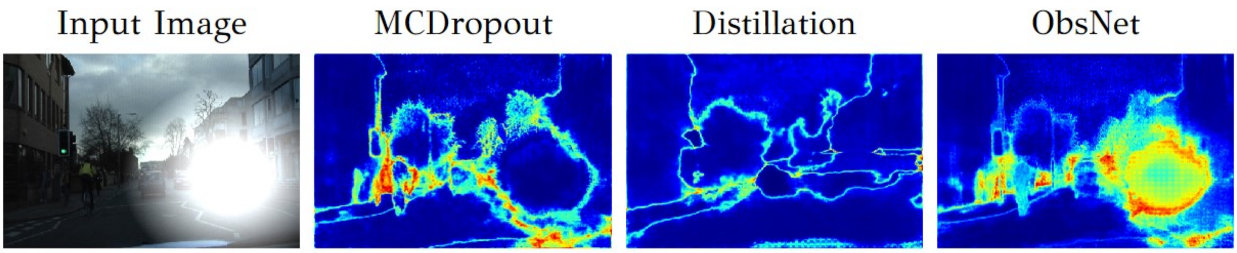


Figure 3.1: **Uncertainty maps for a noisy image.** From left to right: The input image with artificial sun glare. MCDropout with 50 samples detects epistemic uncertainty but not aleatoric uncertainty caused by the sun glare. Distillation is fast in inference but the uncertainty map is inaccurate and badly captures aleatoric uncertainty. Our proposed ObsNet detects epistemic uncertainty and aleatoric uncertainty and is as fast as distillation.

3.1 INTRODUCTION

3.1.1 *Safety in neural networks*

With the recent development of deep learning, neural networks have proven to match and even outperform human level performance at solving complex visual tasks necessary for autonomous driving such as object detection and image recognition [XGD+17, CPSA17, HGDG17]. However, when it comes to safety critical applications, the use of neural networks raises huge challenges still to be unlocked.

Building models able to outperform humans in dealing with unsafe situations or in detecting operating conditions in which the system is not designed to function remains an open field of research. For example, although Softmax outputs class conditional probabilities, it produces misleading high confidence outputs even in unclear situations [MSK18].

This is all the more dramatic in autonomous driving [MKG18] applications where the confidence of the prediction is safety critical. Demonstration of safety for AI components is a key challenge addressed by SOTIF [21419] which focuses on external events (external to the system) that are not correctly handled by the system (e.g. weather conditions, user driving tasks, road users, ...). In such systems, detecting uncertain predictions is a key trigger to produce a safe behavior by interrupting the current process and starting a fallback process (*e.g.*, human intervention) instead of risking a wrong behavior.

3.1.2 *Uncertainty learning with observer network*

Recently, many approaches have sought to compute the uncertainty associated with deep neural networks (epistemic uncertainty). We select Deep Ensemble [LPB17] and MCDropout [GG16][1] as they are representative for the respective categories (ensembles and pseudo-ensembles), conceptually simple and often used in literature as upper bounds or baselines. Both methods measure the variations between several predictions of the same image. Although these methods provide interesting results for epistemic uncertainty estimation, they increase the inference time during the test phase in a prohibitive way. To alleviate this problem, distillation is a very popular method [GBP18][2]. In this context, a student network tries to regress the average of the MCDropout realizations to capture the variance. In addition, the student is also trained to predict the original task output, which leads to a multi-task problem. While distillation solves the computational cost problem, one can ask whether trying to jointly solve the main task and the uncertainty estimation with the same head of a single network can lead to under performing results. In fact, we experimentally show in this chapter that distillation is very underwhelming in a safety critical context.

Moreover, distillation based methods fail to uncover aleatoric uncertainty caused by the absence of information in the image [Gal16, KD09] and even ensembles can sometimes fail in detecting aleatoric uncertainty. For example, both are not able to detect the uncertainty caused by the glare shown on Figure 3.1. This is likely due to the fact that the sun glare noise covers a large portion of the image and several forward passes with different dropout realizations are unlikely to change the output of the network. In this chapter, we propose to predict the uncertainty of a target deep neural network arising from both epistemic and aleatoric causes, without the cost of ensemble methods. We introduce a safety oriented context, where we evaluate

[1] [GG16] Yarin Gal et al., *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*. In ICML 2016.

[2] [GBP18] Corina Gurau et al., *Dropout distillation for efficiently estimating model confidence*. ArXiv 2018.

the effectiveness of an uncertainty measure to detect a safety critical prediction. Our main contribution is a safety oriented uncertainty estimation framework that consists of a deep neural network called ObsNet running in parallel to the target network. ObsNet is trained using self-supervised classes to output predictions that are similar to the target network when the target network is certain, and completely different outputs when the prediction is uncertain. The uncertainty is then measured as the dissimilarity between the target network output and ObsNet output. We empirically show it produces a good proxy for measuring the uncertainty in a safety oriented context. ObsNet has the following properties that allow it to be an improvement over previous methods:

- ObsNet computes an uncertainty measure that takes both epistemic and aleatoric causes into account;
- ObsNet is trained in a simple self-supervised fashion, meaning that no expensive annotations are required;
- ObsNet does not require retraining the target network and can work with any off-the-shelf network;
- ObsNet is fast. The processing of ObsNet happens in parallel to the target network, meaning it has a reduced overhead compared to other uncertainty estimation methods.
- ObsNet improves over other uncertainty estimation methods at detecting safety critical predictions.

3.2 METHOD

In this section, we describe how the observer network is built to predict an uncertainty map as illustrated in Figure 3.2.

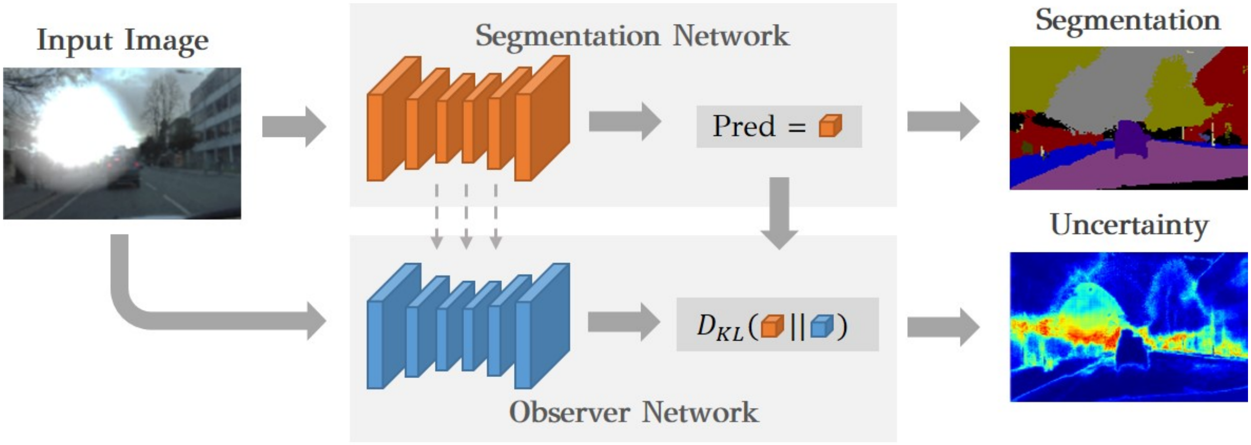


Figure 3.2: **Architecture of our framework** The uncertainty is computed as the Kullback-Leibler divergence between the segmentation network prediction and ObsNet output. Our framework allows the target network to be dedicated to the segmentation part and the ObsNet dedicated to uncertainty measurement.

3.2.1 *Certain set vs. uncertain set*

For safety purposes, it is often better to not make any decision rather than making a decision that cannot be guaranteed. Given a trigger that detects decisions that cannot be guaranteed, a safe system could then stop its current process and start a fallback process (*e.g.* human intervention). Such a trigger divides the predictions $\hat{y}$ of a given neural network into 2 classes: the class of *certain* predictions associated with the class $c = +1$, for which an average error rate can be guaranteed; and the *uncertain* class, associated with the label $c = -1$, for which no average guarantee can be obtained. More formally, we have the following property:

$$\mathbb{E}_{\hat{y}|c=+1}[l(\hat{y}, y)] \leqslant \epsilon, \tag{3.1}$$

with $\hat{y}|c = +1$ the distribution of *certain* predictions, y the ground truth and $l(.,.)$ the loss function of the target application.

We propose to use uncertainty estimation to obtain such safety trigger. Given a function $u(\hat{y})$ that estimates the uncertainty of a prediction $\hat{y}$, we define a safety threshold δ such that predictions under the threshold are in the *certain* set:

$$u(\hat{y}) \leq \delta \Rightarrow c = +1 \tag{3.2}$$

In practice, given an error rate threshold ϵ and its corresponding uncertainty threshold δ, we evaluate uncertainty functions by the recall, that is, the proportion of samples that are deemed certain see Figure 3.3.

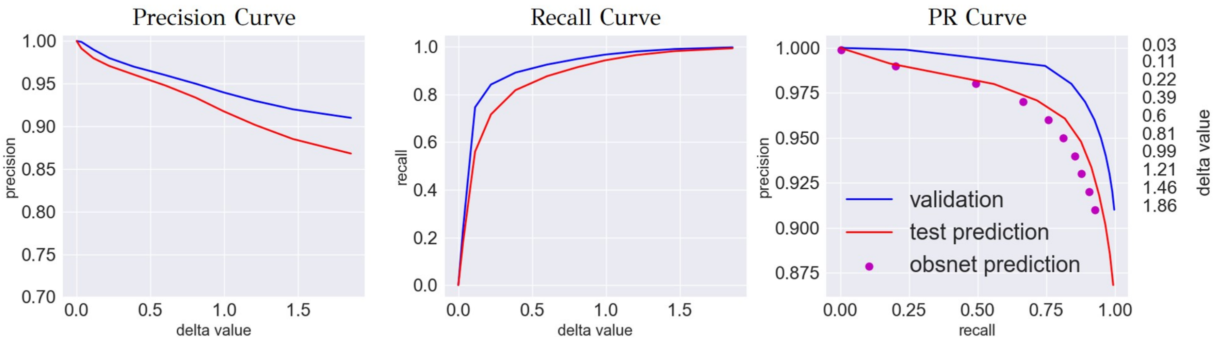


Figure 3.3: **Precision and Recall**. Curve depend on the uncertainty threshold δ for epistemic uncertainty. The hyper-parameter δ allows to select which examples are in the *certain* prediction set (*i.e.*, $c = +1$).

3.2.2 *Divergence based Learning*

To obtain $u(.)$, we propose to model the uncertainty as the dissimilarity between several predictions. Let $\hat{y}$ be the prediction of the neural network we are analyzing, and let y_a be an additional prediction without any assumption on how it is obtained (*e.g.* additional forward of a stochastic model, ensemble, oracle, *etc.*). We propose that measuring the dissimilarity between $\hat{y}$ and y_a gives us a sense of the uncertainty regarding prediction $\hat{y}$. In the case of a regression problem, the dissimilarity can be measured as the distance between the two predictions. In this paper, we focus on softmax classification which is used in semantic segmentation and is well studied in uncertainty estimation. It that case, $\hat{y}$ and y_a being the probability distribution over the different possible classes (softmax outputs), we propose to use the KL divergence to define the uncertainty function $u(.)$:

$$u(\hat{y}) = D_{KL}(\hat{y}||y_a). \tag{3.3}$$

In other words, if $\hat{y}$ and y_a produce similar outputs, the uncertainty of $\hat{y}$ is low, whereas if $\hat{y}$ and y_a produce dissimilar (or disagreeing) outputs, the uncertainty of $\hat{y}$ is high.

Unfortunately, it is clear that the additional prediction y_a may not be available at inference time. Our main contribution solves this problem by introducing a second predictor, the observer with output y_o which we use in place of y_a. Training y_o to precisely regress the predictions y_a can be a difficult learning problem. Moreover, in case of *uncertain* predictions, it is not required that y_o perfectly matches y_a. Instead, y_o only has to be sufficiently different from $\hat{y}$ to produce a large KL divergence, just as y_a would have done. Therefore, we propose to train y_o using a self-supervised classification problem distinguishing between *certain* and *uncertain* predictions.

In practice, given a training set of pairs $(\hat{y}, y_a)$, we use the safety threshold δ to split the pairs into *certain* $c = +1$ and *uncertain* $c = -1$ classes, and we train our predictor ObsNet to minimize $D_{KL}(\hat{y}||y_o)$ for $c = +1$ and maximize $D_{KL}(\hat{y}||y_o)$ for $c = -1$ by optimizing the following problem:

$$\min_{\theta} D_{KL}(\hat{y}||y_o)^c, \tag{3.4}$$

with θ the parameters of the observer. We argue that this objective is much easier to optimize than regressing y_a since the output y_o has many more degrees of freedom in the *uncertain* case. Instead of being forced to predict the exact same class as y_a, y_o can predict any class different from the one predicted by $\hat{y}$.

3.2.3 *Oracle Prediction y_a*

Usually, uncertainty is classified into two different classes: Epistemic and Aleatoric which we both propose to estimate using different additional predictions y_a:

Epistemic uncertainty is associated with the model uncertainty. It captures the lack of knowledge about the process that generated the data. To estimate epistemic uncertainty, we propose to use MCDropout as the additional prediction y_a. We compute the additional prediction y_a as the average of T forward passes with dropout. Note that our training setup is entirely self-supervised as the labels c are obtained using D_{KL} over forward passes of the target network only.

We show on Figure 3.3 what setting a specific threshold δ implies in terms of precision and recall for *certain* $(c = 1)$ predictions. As we can see, the observer is perfectly able to recover the specific operating points of the uncertainty obtained by the original MCDropout additional prediction.

Aleatoric uncertainty is associated with the natural randomness of the input signal [KG17][1]. More precisely, in this work, we focus on heteroscedastic uncertainty, which is the lack of visual features in the input data (e.g. sun glare, occlusion, ...). We propose to artificially create such cases by adding random glare noise to the input image. The prediction $\hat{y}$ is obtained by a single forward pass on the noisy image, while the additional prediction y_a is obtained by a single forward pass on the clean image.

The uncertainty function $u(\hat{y}) = D_{KL}(\hat{y}||y_a)$ then establishes whether the noise we added led to aleatoric uncertainty or not. Thanks to δ, we label the pair with either $c = +1$ or $c = -1$. As for epistemic uncertainty, we train our auxiliary network on the labeled pairs $(\hat{y}, y_a)$ using Equation 3.4.

3.3 EXPERIMENTS

We present our results in this section. We show the benefit of using divergence based uncertainty to train an ObsNet is a good choice for autonomous driving perception applications. We conduct evaluations on on CamVid, against others competitive methods.

3.3.1 *Dataset, Metrics and Compared Methods*

In this section, we evaluate our method on **CamVid** [BFC08][2], a dataset for driving scene image segmentation containing 11 classes including cyclist, road, sky, pedestrian etc. It contains 367 training images and 233 test images. Each image is resized to 360x480 pixels. Using CamVid, we evaluate both epistemic and aleatoric uncertainty:

Epistemic: We consider the natural epistemic uncertainty in the model as well as the one arising from Out-of-Distribution (OoD) samples. To simulate OoD samples, we blit animals unseen during training such as lions, horses, bears, etc.

Aleatoric: For data uncertainty, we added random artifact in the image: Glare in the image (*i.e.*, important increase of brightness in an ellipse), Rain (*i.e.*, grey line on the image) and patches (*i.e.*, rectangles of uniform color and random sizes). Each artifact varies in terms of size and coordinates on the image.

To compare the results, we adopt safety oriented metrics: **Trigger**, **Recall at 95% Precision**, **Area under the Precision-Recall Curve (AuPR)** [HG17] and **Expective Calibration Error (ECE)** [GPSW17].

We report our results on the following table, with all metrics above. We compare several methods:

- **MCP [HBM$^+$19]**: One minus the maximum of the softmax probability.
- **Void Class (VC)[BSN$^+$19]**: Void/unknown class prediction for segmentation.

[1] [KG17] Alex Kendall et al., *What uncertainties do we need in bayesian deep learning for computer vision?* In NeurIPS 2017

[2] [BFC08] Gabriel J. Brostow et al;, *Semantic object classes in video: A high-definition ground truth database*. In PRL 2008.

- **MCDropout [Gal16]**: We consider this as a baseline. We use $T = 50$ and $T = 2$ forward passes.
- **MCDA[AB18]**: Data augmentation such as geometric and color transformations added during inference time to capture aleatoric uncertainty.
- **Distillation [GBP18]**: We propose two variants of the distillation: supervised (*i.e.*, using Teacher outputs and cross-entropy with ground truth for training) and unsupervised (*i.e.*, the student only regress the teacher output).
- **Observer**: Our proposed method, with KL divergence based uncertainty training. We use two oracles: MCDropout (self-sup) and Ground Truth (GT). The Ground Truth variant uses ground truth labels as additional prediction y_a and is used to measure the influence of having a self supervised setup.

For all our segmentation experiments we use a Bayesian SegNet [BKC17][1], [KBC15] with dropout as the main network. Therefore, our ObsNet follows the same architecture as this SegNet. To train distillation for aleatoric uncertainty estimation, we change the training set up to be fair with our method. When the student gets a noisy image, it is trained to output the same prediction as the teacher given a de-noised image.

3.3.2 *Results: Epistemic uncertainty*

We first report results for epistemic uncertainty on Table 3.1 and Table 3.2. As we can see in Table 3.1, ObsNet GT performs best or similar to MCDropout in all the safety metrics. Our self-supervised ObsNet method is better than MCDropout at comparable computational cost. Distillation alone offers low performances, which is due to the complexity of regressing the exact MCDropout outputs and difficulty to capture both uncertainty and class prediction.

To validate that our method is able to capture epistemic uncertainty we test on OoD samples. We show in Table 3.2 results on the same dataset but with animals randomly blit in the image. Again our method perform the best with ObsNet GT and self-supervised is close to MCDropout. Distillation again performs badly.

We show qualitative uncertainty maps of Figure 3.5. MCDropout and ObsNet output very similar maps for epistemic uncertainty.

3.3.3 *Results: Aleatoric uncertainty*

We evaluate aleatoric uncertainty on Table 3.3 and Table 3.4. As we can see, MCDropout suffers dramatic failures and is unable to obtain high recall for both the sun glare and square patch. This is to be expected since MCDropout is not designed to capture aleatoric uncertainty. MCDA performs the best on glare among set-up without

[1] [BKC17] Vijay Badrinarayanan et al. *SegNet: A deep convolutional encoder-decoder architecture for image segmentation*. In PAMI 2017.

		R@P=0.95 ↑	AuPR ↑	Trigger 75% ↑	ECE ↓	Run Time
no-retrain	MCDropout T = 50	88.2	**97.9**	**81.5**	**0.025**	0.789
	MCDropout T = 2	85.9	97.5	76.4	**0.025**	0.036
	MCP	81.0	96.5	67.4	0.048	**0.018**
	void class	64.1	95.2	39.9	0.123	**0.018**
Distillation	w/ T supervised	65.9	95.3	31.8	0.060	0.370
	w/ S supervised	70.3	95.6	30.5	0.101	0.021
	w/ T unsupervised	68.6	96.0	34.3	0.045	0.370
	w/ S unsupervised	66.6	95.3	23.2	0.092	0.021
ObsNet	from MCDropout	87.3	97.7	76.8	0.046	0.038
	from GT	**89.3**	**97.9**	**81.5**	0.051	0.038

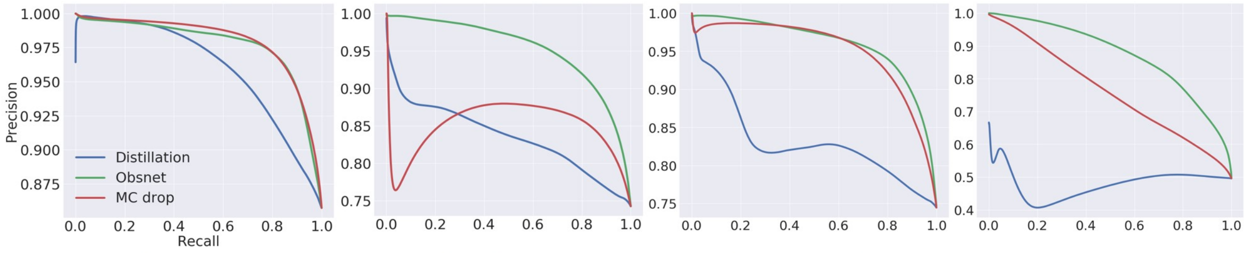
Table 3.1: **Evaluation of epistemic uncertainty.** ObsNet GT perform well, and the self-supervised version has the best trade-off between accuracy and inference time.

		R@P=0.95 ↑	AuPR ↑	Trigger 75% ↑	Ece ↓
no-retrain	MCDropout T = 50	85.1	**97.3**	67.3	**0.020**
	MCDropout T = 2	80.9	96.0	59.9	0.030
	MCP	65.9	95.2	39.1	0.071
	void class	56.1	94.2	20.2	0.140
Distillation	w/ T supervised	47.5	92.9	11.9	0.087
	w/ S supervised	63.5	94.8	23.1	0.098
	w/ T unsupervised	55.3	94.2	11.8	0.070
	w/ S unsupervised	61.6	94.7	19.0	0.086
ObsNet	from MCDropout	82.8	97.2	61.2	0.035
	from GT	**85.3**	**97.3**	**67.9**	0.070

Table 3.2: **Out-of-Distribution Evaluation.** ObsNet GT still performs the best. While our self-supervised method stays competitive.

additional network. As with epistemic uncertainty, distillation does not succeed in producing good results. In contrast, our ObsNet obtains significantly better results in all the considered cases[1].

We show Precision-Recall curves on Figure 3.4. As we can see, MCDropout produces overconfident predictions on noisy areas which leads to very low precision at low recalls. This is qualitatively visible on Figure 3.5 where noisy areas (e.g. glare on the third line) have very low uncertainty values. Overall, ObsNet significantly outperforms MCDropout and data-augmentation based methods for aleatoric uncertainty while being much more computationally efficient.



Classic Sun Glare Square Patch Rain

Figure 3.4: **Precision-Recall Curves.** We can see that Precision of Distillation and MCDropout drop significantly even for small recall when aleatoric uncertainty arises. ObsNet is much more robust to this uncertainty.

To evaluate the generalization capabilities of ObsNet, we train on different artifacts than the tested ones (see Table 3.3 and Table 3.4). As we can see, although training on a different noise decreases the performance compared to the full training, ObsNet still outperforms MCDropout, distillation and MCDA by a large margin. This shows the capacity of ObsNet to generalize the unseen noises.

Test Sun Glare	Method	Train	R@P=0.95 ↑	AuPR ↑	Trigger 75% ↑
no-retrain	MCDropout T=50		0.1	83.9	18.9
	MCDropout T=2		0.0	83.7	15.5
	MCP		1.0	90.5	17.2
	Void Class		0.3	82.7	8.2
	MCDA T=50		44.7	91.7	14.3
Distillation	w/ T supervised	patch	0.3	85.7	4.1
	w/ T supervised	rain	0.5	77.1	3.8
	w/ T supervised	glare	0.0	83.3	2.2
	w/ T supervised	all	0.0	81.9	1.7
ObsNet	self-supervised	patch	46.7	92.3	15.1
	self-supervised	rain	33.9	90.8	18.3
	self-supervised	glare	68.4	95.3	23.2
	self-supervised	all	**76.1**	**96.1**	**33.5**

Table 3.3: **Aleatoric uncertainty tested on Sun Glare.** Our ObsNet outperforms every other methods, while MCDropout suffers from dramatic failures.

3.4 CONCLUSION

In this chapter, we have presented a simple, yet effective method to estimate uncertainty using an observer network. We introduce a

[1] Full Table with all different aleatoric artifacts can be in the appendix.

Test Patch	Method	Train	R@P=0.95 ↑	AuPR ↑	Trigger 75% ↑
no-retrain	MCDropout T=50	-	63.9	94.9	19.6
	MCDropout T=2	-	23.4	93.5	18.5
	MCP	-	0.3	90.3	15.9
	Void Class	-	0.2	86.4	8.7
	MCDA T=50	-	32.0	90.4	12.4
Distillation	w/ T supervised	glare	0.0	85.2	3.1
	w/ T supervised	rain	0.0	54.0	0.0
	w/ T supervised	patch	1.4	79.6	4.4
	w/ T supervised	all	0.0	79.5	1.9
ObsNet	self-supervised	glare	46.5	92.6	15.2
	self-supervised	rain	2.7	70.2	0.9
	self-supervised	patch	72.5	95.6	23.6
	self-supervised	all	**77.5**	**96.1**	**30.5**

Table 3.4: **Aleatoric uncertainty tested on Square Patch.** Our ObsNet outperforms every other methods and is computationally more efficient compared to

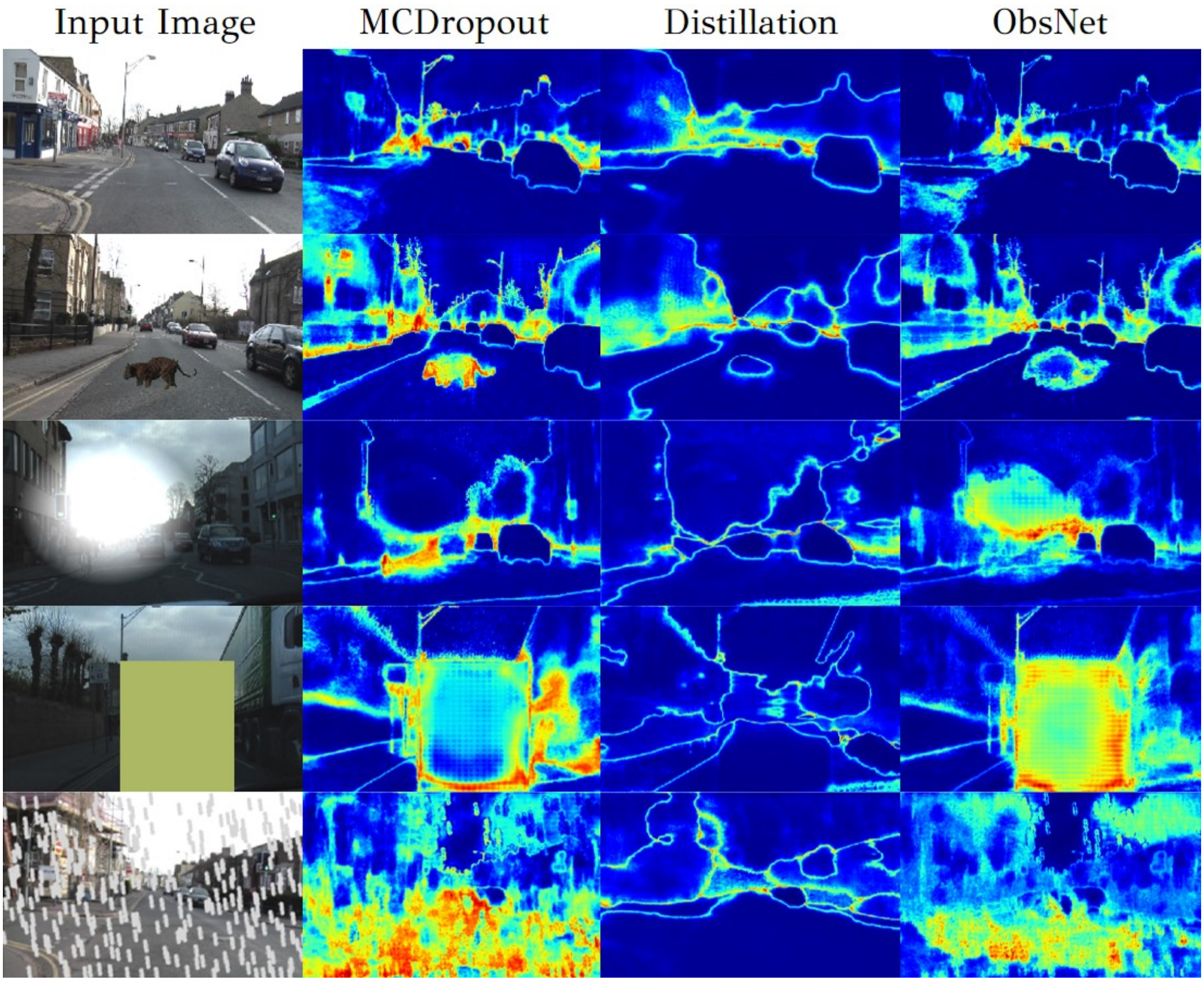


Figure 3.5: **Uncertainty map.** From column left to right: Input image, MC-Dropout uncertainty, distillation uncertainty, and ObsNet uncertainty. First line is with an OoD animal and the second one is with glare. ObsNet is not only close to MCDropout for epistemic detection, but is also capable of detecting aleatoric uncertainty.

safety oriented framework where the uncertainty is used to trigger a safety signal when a given error rate cannot be met. In that context, ObsNet is specifically trained such that its uncertainty predictions are usable as a safety indicator. Contrarily to ensemble methods, ObsNet requires a single forward pass making it computationally efficient. In contrast to distillation based methods, ObsNet relies on self-supervised classes and is much more effective.

Our observer uses an oracle to pre-process the dataset into certain and uncertainty set. However, ObsNet relies on the performance of this oracle and thus, the efficiency of the auxiliary network is upper bounded by how good the oracle is. In other words, the performance of the observer cannot be better than that of the oracle. We tackle this problem in the Chapter 4. We reuse the framework of the observer network, but we get rid of the oracle and leverage local adversarial attacks to train the observer instead.

4

Local Adversarial Attacks

◂ Chapter 3 Chapter 5 ▸

Synopsis In this chapter, we use the Observer Network as a dedicated Out-of-Distribution (OoD) detector for semantic segmentation. Our previous approach used an oracle prediction to guide the training of the ObsNet. The observer was trained to predict uncertainty thanks to a divergence based approach, but the final accuracy was bounded by the oracle performance. Here, we no longer rely on an external prediction to train the observer. We propose to mitigate the common shortcomings by following two principles: generating training data for the OoD detector by leveraging blind spots in the segmentation network and focusing the generated data on localized regions in the image to simulate OoD objects. Our main contribution is an ObsNet associated with a dedicated training scheme based on Local Adversarial Attacks (LAA) for OoD detection. We validate the soundness of our approach across numerous ablation studies. We also show it obtains top performances both in speed and accuracy when compared to ten recent methods of the literature on three different datasets. This chapter is based on our paper ”Triggering Failures: Out-Of-Distribution detection by learning from local adversarial attacks in Semantic Segmentation”, in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).

4.1 INTRODUCTION

In this work, we address OoD detection for semantic segmentation, an essential and common task for visual perception in autonomous vehicles. We propose to use an Observer Network trained with adversarial attacks as a proxy to detect pixels from objects not in the training distribution.

4.1.1 *OoD Detection & Lack of Available Errors*

Here, we consider ”Out-of-Distribution” pixels from a region that has no training labels associated with. This encompasses unseen objects, but also noise or image alterations. The most effective methods for OoD detection tasks stem from two major categories of approaches: ensembles and auxiliary error prediction modules. DeepEnsemble (DE) [LPB17][2] is a prominent and simple ensemble method that exposes potentially unreliable predictions by measuring the disagreement between individual DNNs. In spite of the outstanding

[2] [LPB17] Balaji Lakshminarayanan et al., *Simple and scalable predictive uncertainty estimation using deep ensembles*. In NeurIPS 2017.

performance, DE is computationally demanding for both training and testing and prohibitive for real-time on-vehicle usage. For the latter category, given a trained main task network, a simple model is trained in a second stage to detect its errors or estimate its confidence [CTBH+19, HDVG18, BPB21]. Such approaches are computationally lighter, yet, in the context of DNNs, an unexpected drawback is related to the lack of sufficient negative samples, *i.e.*, failures, to properly train the error detector [CTBH+19][1]. This is due to an accumulation of causes: reduced size of the training set for this module (essentially a mini validation set to withhold a sufficient amount for training the main predictor), few mistakes made by the main DNNs, hence few negatives.

We propose to revisit the two-stage approach with modern deep learning tools in a semantic segmentation context. Given the application context, *i.e.*, limited hardware and high performance requirements, we aim for reliable OoD detection (see Figure 4.1) without compromising on predictive accuracy and computational time. To that end we introduce two design principles aimed at mitigating the most common pitfalls of training an observer network:

- Training an OoD detector requires additional data that can be generated by leveraging blind spots in the segmentation network.
- Generated data should focus on localized regions in the image to mimic unknown objects that are OoD.

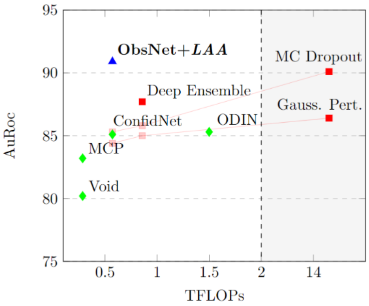


Figure 4.1: **Evaluation of precision vs. test-time computational cost on CamVid OoD.** Existing methods for OoD detection in semantic segmentation are either accurate but slow or fast but inaccurate. In contrast, our method ObsNet+*LAA* is both accurate and fast.

4.1.2 *Adversarial Attacks*

Meanwhile, it is well known that deep neural networks are vulnerable against adversarial attacks. Adversarial examples are structured perturbations added in the input image that fools the network's prediction. Fast Gradient Sign Method (FGSM) [GSS15][2] is a simple and popular attack that computes an adversarial example as:

$$\tilde{x} = x + \epsilon sign(\bigtriangledown_x \mathcal{L}(\theta, x, y)). \tag{4.1}$$

with x the image, y the ground truth, ϵ the step, and $L(.)$ the cross entropy. In other words, FGSM will add a noise in x to maximize the error.

With FGSM, we manage to generate as many negative samples as we perform attacks. While it is hard to formalize OoD for natural image distribution, we show that adversarial attack plays the role of a *proxy* between in and out-of-distribution.

4.1.3 *Strengths*

By combining the observer network paradigm with the training scheme of *LAA*, our method highlights several benefits:

✓ It can be used with any pre-trained segmentation network without

[1] [CTBH+19] Charles Corbière et al., *Addressing failure prediction by learning model confidence*. In NeurIPS 2019.

[2] [GSS15] Ian J. Goodfellow et al., *Explaining and harnessing adversarial examples*. In ICLR 2015.

altering their performances and without fine-tuning them (we train only the auxiliary module).

✓ It is fast since only one extra forward pass is required.

✓ It is very effective since we show it performs best compared to 10 very diverse methods from the literature on three different datasets.

4.2 METHOD

Training an OoD detector without OoD data is difficult, but can be done nonetheless by generating training data with carefully designed adversarial attacks. This aspect requires careful design to work effectively, which we detail in the following. We validate them experimentally in §4.3.

4.2.1 *ObsNet for Error Detection*

Modifying the segmentation network to account for OoD is expected to impact its accuracy as we show in the experiments. Furthermore, it prevents from using off-the-shelf pretrained segmentation networks that have excellent segmentation accuracy. As such, we follow a two-stage approach where an additional predictor tackles the OoD detection while the segmentation network remains untouched.

In the literature, two-stage approaches are usually related to calibration [P+99, ZE01, ZE02, NCH15, NC16] where the outputs of the segmentation network are mapped to normalized scores. However this is not well adapted for segmentation since it does not use the spatial information contained in nearby predictions. We show in the experiments that using only the output of the segmentation network is not enough to obtain accurate OoD detection.

We thus reuse the observer network framework introduced in chapter 2. This auxiliary network has a similar architecture to that of the segmentation network and use the input, the output and intermediate feature maps of the segmentation network as shown on Figure 4.2. We show experimentally that these design choices lead to increased OoD detection accuracy.

More formally, the observer network (denoted Obs) is trained to predict the probability that the segmentation network (denoted Seg) output is not aligned with the correct class y:

$$Obs(x, Seg_r(x)) \approx Pr[Seg(x) \neq y], \tag{4.2}$$

where x is the input image and Seg_r the skip connections from intermediate feature maps of Seg.

To that end, we train the ObsNet to minimize a binary cross-entropy

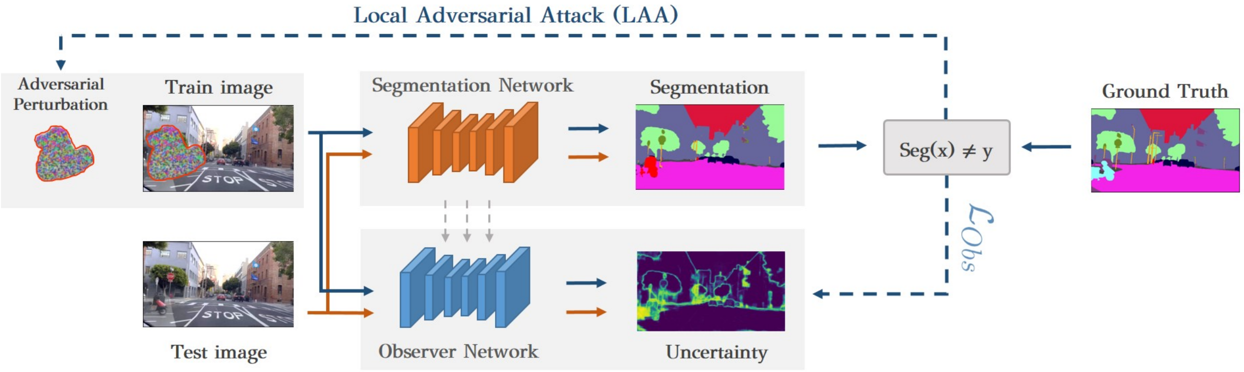


Figure 4.2: **Detailed Overview of our method. Training (blue arrow)** The *Segmentation Network* is frozen. The input image is perturbed by a local adversarial attack. Then the *Observer Network* is trained to predict *Segmentation Network*'s errors, given the images and some additional skip connections. **Testing (red arrow)** No augmentation is performed. The *Observer Network* highlights the out-of-distribution sample, here a motor-cycle. To compute the uncertainty map, the *Observer Network* requires only one additional forward pass compared to the standard segmentation prediction.

loss function:

$$\mathcal{L}_{Obs}(x,y) = (\mathbb{1}_{Seg(x)\neq y} - 1)\log(1 - Obs(x, Seg_r(x))) \\ - \mathbb{1}_{Seg(x)\neq y}\log Obs(x, Seg_r(x)) \quad (4.3)$$

with $\mathbb{1}_{Seg(x)\neq y}$ the indicator function of $Seg(x) \neq y$.

We emphasize an advantage of our approach w.r.t. previous methods that are related to the low computational complexity, as we only have to make a single forward pass through the segmentation network and the observer network. Experimentally, ObsNet is 21 times faster than MCDropout with 50 forward passes on a GeForce RTX 2080 Ti, while outperforming it (see §4.3). Moreover, our method can be readily used on state of the art pre-trained networks without requiring retraining or even fine-tuning them.

Without a dedicated training set of labeled OoD samples, one could argue that ObsNet is an error detector (similarly to [CTBH+19]) rather than an OoD detector and that it is furthermore very difficult to train since pre-trained segmentation networks are likely to make few errors. We propose to solve both of these issues by following two design principles:

- The lack of training data should be tackled by generating training samples that trigger failures of the segmentation network, which we can obtain using adversarial attacks.
- Adversarial attacks should be localized in space since OoD detection in a segmentation context corresponds to unknown objects.

4.2.2 *Local Adversarial Attacks*

We propose to generate the additional data required to train our ObsNet architecture by performing Local Adversarial Attacks (*LAA*) on the input image. In practice, we select a region in the image by using a random shape and we perform a FSGM [GSS15] attack such that it is incorrectly classified by the segmentation network:

$$\tilde{x} = x + LAA(Seg, x) \quad (4.4)$$

$$LAA(Seg, x) = \epsilon\,\text{sign}(\nabla_x \mathcal{L}(Seg(x), y))\Omega(x) \quad (4.5)$$

with step ϵ, $\mathcal{L}(\cdot)$ the categorical cross entropy and $\Omega(x)$ the binary mask of the random shape. We show *LAA* examples in Figure 4.3 and schematize the training process in Figure 4.2.

The reasoning behind *LAA* is two-fold. First, by controlling the shape of the attack, we can make sure that the generated example does not accidentally belong to the distribution of the training set. Second, leveraging adversarial attacks allows us to focus the training just beyond the boundaries of the predicted classes which tend to be far from the training data due to the high capacity and overconfidence of DNNs, like OoD objects would be.

We show in the experiments that *LAA* produces a good training set for learning to detect OoD examples. In practice, we found that generating random shapes is essential to obtain good performances in contrast to non-local adversarial attacks. These random shapes coupled with *LAA* may mimic unknown objects or objects parts, exposing abnormal behavior patterns in the segmentation network when facing them. We validate our approach in an ablation study in §4.3.2.

Discussion. We point out that by triggering failures using *LAA*, we address the problem of the low error rates of the segmentation network. We can in fact generate as many OoD-like examples as needed to balance the positive (*i.e.*, correct predictions) and negative (*i.e.*, erroneous predictions) terms in Equation 4.3 for training the observer network. Thus, even if the segmentation network attains nearly perfect performances on the training set, we are still able to train the ObsNet to detect where the predictions of the segmentation network are unreliable.

One could ask why not using *LAA* for training a more robust and reliable segmentation network in the first place, as done in previous works [GSS15, MMKI18, HLM19], instead of adding and training the observer network. Training with adversarial examples improves the robustness of the segmentation network at the cost of its accuracy (See §4.3.2), but it will not make it infallible as there will still be numerous blind-spots in the multi-million dimensional parameter space of the network. It also prevents using pre-trained state-of-the-art segmentation networks. Here, we are rather interested in capturing the main failure modes of the segmentation network to enable ObsNet to learn and to recognize them later on OoD objects.

Finally, one could ask why not perform adversarial attacks at test time as it is done in ODIN [LSL18][1]. Performing test time attacks has two major drawbacks. First it is computationally intensive at test time since it requires numerous backward passes, *i.e.*, one attack per pixel. Second, it is not well adapted to segmentation as perturbations of a single pixel can have effect on a large area (e.g., one pixel attacks) thus hindering the detection accuracy of perfectly valid predictions. We show in §4.3 that our training scheme is better performing both in accuracy and speed when compared to test time attacks.

[1] [LSL18] Shiyu Liang et al., *Enhancing the reliability of out-of-distribution image detection in neural networks*. In ICLR 2018.

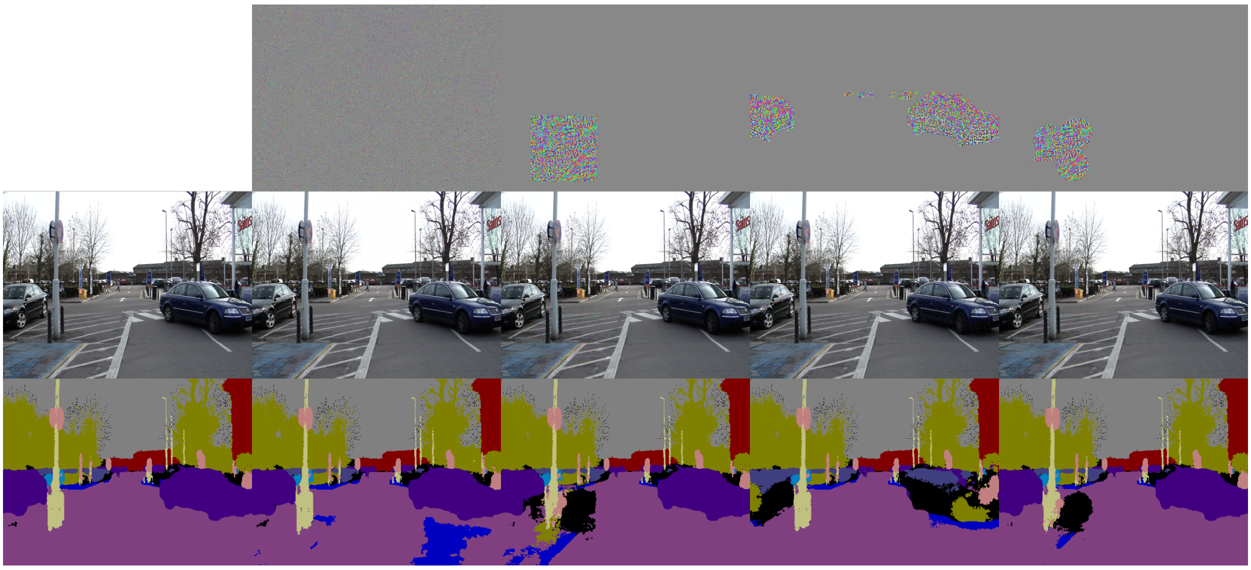


Figure 4.3: **Adversarial attack examples.** *Top*: Perturbations magnified 25×; *middle*: Input image with attacks; *bottom*: SegNet prediction. Adversarial Attacks are not visible in the image, but the accuracy of the SegNet drops significantly. Our ObsNet is trained on these patterns.

4.3 EXPERIMENTS

In this section, we present extensive experiments to validate that our proposed observer network combined with local adversarial attacks outperforms a large set of very different methods on three different benchmarks.

4.3.1 *Datasets, Metrics and Compared Methods*

To highlight our results, we select three datasets for Semantic Segmentation of urban streets scenes with anomalies in the test set. Anomalies correspond to out-of-distribution objects, not seen during train time. We select CamVid OoD, StreetHazards [HBM+19] and BDD Anomaly [HBM+19]. To evaluate each method on these datasets, we select three metrics for detecting misclassified and out-of-distribution examples and one metric for calibration: **Fpr95Tpr** [LSL18], **Area Under the Receiver Operating Characteristic curve (AuRoc)** [HG17], **Area under the Precision-Recall Curve (AuPR)** [HG17] and **Adaptive Calibration Error (ACE)** [NDZ+19].

We report results on Table 4.5, Table 4.6 and Table 4.7, with all the metrics detailed above. We compare several methods:

- **MCP** [HG17]: Maximum Class Prediction. One minus the maximum of the prediction.
- **AE** [HG17]: An autoencoder baseline. The reconstruction error is the uncertainty measurement.
- **Void** [BSN+19]: Void/background class prediction of the segmentation network.
- **MCDA** [AB18]: Data augmentation such as geometric and color transformations is added during inference time. We use the

entropy of 25 forward passes.

- **MCDropout** [GG16]: The entropy of the mean softmax prediction with dropout. We use 50 forward passes for all the experiences.
- **Gaussian Perturbation Ensemble** [FBA+20, MAG+20]: We take a pre-trained network and perturb its weights with a random Normal distribution. This results in an ensemble of networks centered around the pre-trained model.
- **ConfidNet** [CTBH+19]: ConfidNet is an observer network that is trained to predict the true class score. We use the code available online and modify the data loader to test ConfidNet on our experimental setup.
- **Temperature Scaling** [GPSW17]: We chose the hyper-parameters *Temp* to have the best calibration on the validation set. Then, like MCP, we use one minus the maximum of the scaled prediction.
- **ODIN** [LSL18]: ODIN performs test-time adversarial attacks on the primary network. We seek the hyper-parameters *Temp* and ϵ to have the best performance on the validation set. The criterion is one minus the maximum prediction.
- **Deep ensemble** [LPB17]: a small ensemble of 3 networks. We take the average of the predictions from all networks, and compute the entropy of it.

For all our segmentation experiments we use a Bayesian SegNet [BKC17][1], [KBC15] as the main network. Therefore, our ObsNet follows the same architecture as this SegNet.

4.3.2 *Results: OoD Detection*

First, to validate that the Local Adversarial Attacks contribute to improving the observer network, we show on Table 4.1 the performance gap for each metric on each dataset. This validates the use of *LAA* to train the observer network.

Dataset	Adv	Fpr95Tpr ↓	AuPR ↑	AuRoc ↑
CamVid OoD	-	54.2	97.1	89.1
	✓	**44.6**	**97.6**	**90.9**
StreetHazards	-	50.1	98.3	89.7
	✓	**44.7**	**98.9**	**92.7**
BDD Anomaly	-	62.4	95.9	81.7
	✓	**60.3**	**96.2**	**82.8**

Table 4.1: **Evaluation of the Local Adversarial Attack.** Local Adversarial Attacks improves ObsNet accuracy on each dataset.

The *LAA* can be seen as a data augmentation performed during ObsNet training. We emphasize that this type of data augmentation is not beneficial for the main network training, which is known as *robust training* [MMS+18][2] during ObsNet training, thus, the class prediction and the accuracy remain unchanged.

[1] [BKC17] Vijay Badrinarayanan et al. *SegNet: A deep convolutional encoder-decoder architecture for image segmentation*. In PAMI 2017.

[2] [MMS+18] Aleksander Madry et al., *Towards deep learning models resistant to adversarial attacks*. In ICLR 2018.

Dataset	Robust	Mean IoU ↑	Global Acc ↑
Camvid ODD	✗	**49.6**	**81.8**
	✓	41.6	73.9
StreetHazards	✗	**44.3**	**87.9**
	✓	37.8	85.1
Bdd Anomaly	✗	**42.9**	**87.0**
	✓	41.5	85.9

Table 4.2: **Impact of robust training on accuracy.** Robust training implies a drop of accuracy for the segmentation network.

In Table 4.3, we show ablations on *LAA* by varying the type of noise (varying between attacking all pixels, random pixels, pixels from a specific class, pixels inside a square shape and pixels inside a random shape, see Figure 4.3). We conclude that local attacks on random shaped regions produce the best proxies for OoD detection.

Type	Fpr95Tpr ↓	AuPR ↑	AuRoc ↑
All pixels	51.9	97.1	89.6
Sparse pixels	54.2	97.2	89.6
Class pixels	46.8	97.2	89.9
Square patch	45.5	**97.4**	90.5
Random shape	**44.6**	**97.4**	**90.6**

Table 4.3: ***LAA* ablation study by varying the attacked region.** Attacking the segmentation network in a randomly located region performs the best.

In Table 4.4, we conduct several ablation studies on the architecture of ObsNet. The main takeaway is that mimicking the architecture of the primary network and adding skip connections from several intermediate feature maps is essential to obtain the best performances. Surprisingly, the observer network is able to have low Fpr95Tpr even without the input image. Moreover, re-use the weight from the segmentation network help to increase Fpr95Tpr by up to 1.5%. For the smaller architecture, instead of keeping the same architecture as the segmentation network, we design a smaller variant: a convolutional network with three convolutional layers and a fully connected layer. This architecture mimicks the one used by ConfidNet [CTBH+19].

Method	Fpr95Tpr ↓	AuPR ↑	AuRoc ↑	ACE ↓
Smaller architecture	60.3	95.8	85.3	0.476
ObsNet w/o skip	81.3	92.0	74.4	0.551
ObsNet w/o input img	57.0	96.9	88.2	0.455
ObsNet w/o pretrain	55.7	96.9	88.7	0.419
ObsNet	**54.2**	**97.1**	**89.1**	**0.396**

Table 4.4: **ObsNet architecture ablation study.** This table shows the huge benefit to have residual connections at different stages of the segmentation network. This residual connection is used by ObsNet to observe the internal behavior of the main network.

In Table 4.5, Table 4.6 and Table 4.7 we show results on datasets that all contain OoD objects. We can see that ObsNet significantly outperforms all other methods on detection metrics on all three datasets. We highlight that our methods is also much faster than other high-ranked methods. Our methods is more than 20 times faster than MCDropout, the second best method on these tables. Furthermore, ACE also shows that we succeed in having a good calibration value.

Table 4.5: **Evaluation on CamVidOoD.** Best method in bold, second best underlined.

Method	Fpr95Tpr↓	AuPR ↑	AuRoc ↑	ACE ↓
Softmax [HG17]	65.4	94.9	83.2	0.510
Void [BSN+19]	66.6	93.9	80.2	0.532
AE [HG17]	93.0	87.1	59.3	0.745
MCDA [AB18]	66.5	94.6	82.1	0.477
Temp. Scale [GPSW17]	63.8	94.9	83.7	**0.356**
ODIN [LSL18]	60.0	95.4	85.3	0.500
ConfidNet [CTBH+19]	60.9	96.2	85.1	0.450
Gauss P [MAG+20]	59.2	96.0	86.4	0.520
Deep Ensemble [LPB17]	56.2	96.6	87.7	0.459
MCDropout [GG16]	49.3	97.3	90.1	0.463
ObsNet + *LAA*	**44.6**	**97.6**	**90.9**	0.446

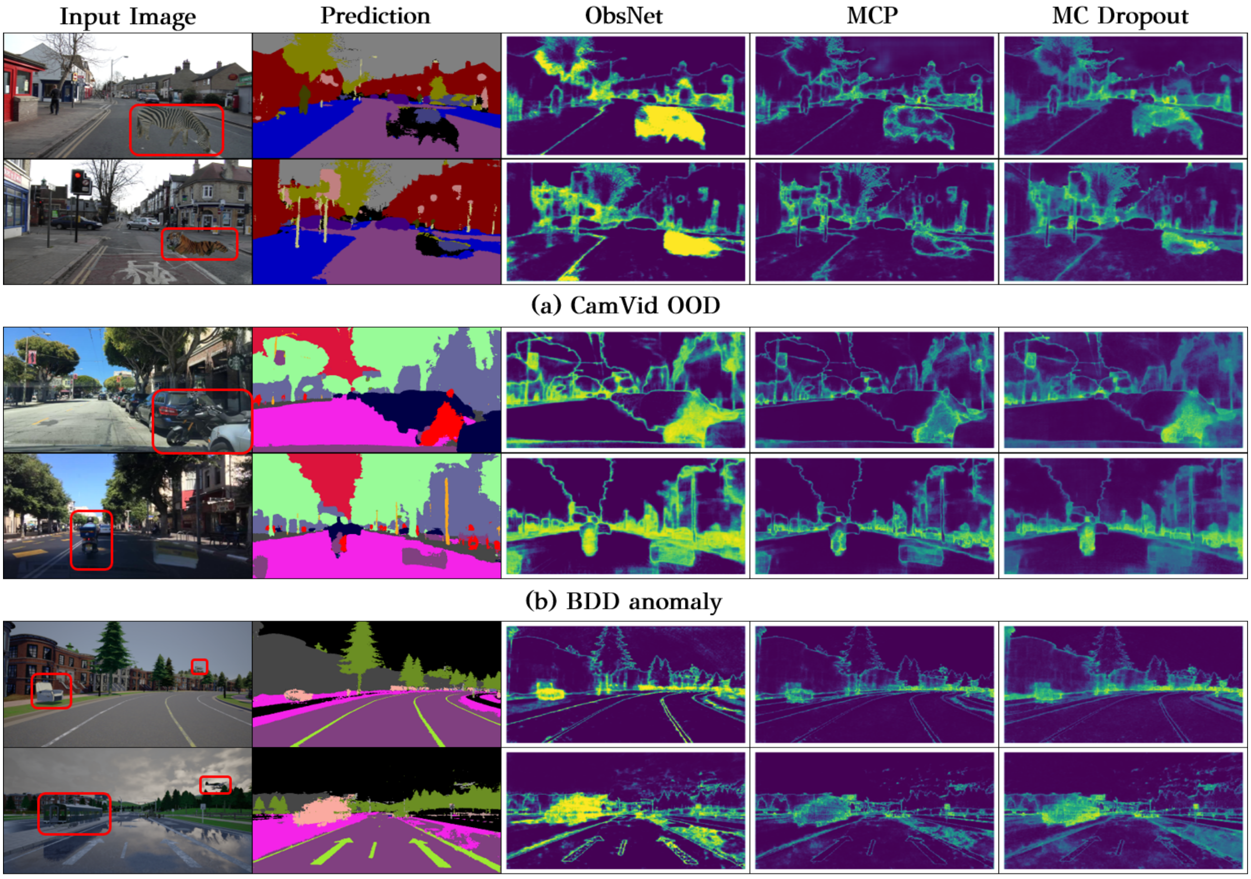
Table 4.6: **Evaluation on StreetHazard.** Best method in bold, second best underlined.

Method	Fpr95Tpr ↓	AuPR ↑	AuRoc ↑	ACE ↓
Softmax [HG17]	65.5	94.7	80.8	0.463
Void [BSN+19]	69.3	93.6	73.5	0.492
AE [HG17]	84.6	92.7	67.3	0.712
MCDA [AB18]	69.9	97.1	82.7	0.409
Temp. Scale [GPSW17]	65.3	94.9	81.6	**0.323**
ODIN [LSL18]	61.3	95.0	82.3	0.414
ConfidNet [CTBH+19]	60.1	98.1	90.3	0.399
Gauss P [MAG+20]	48.7	98.5	90.7	0.449
Deep Ensemble [LPB17]	51.7	98.3	88.9	0.437
MCDropout [GG16]	45.7	98.8	92.2	0.429
ObsNet + *LAA*	**44.7**	**98.9**	**92.7**	0.383

Table 4.7: **Evaluation on Bdd Anomaly.** Best method in bold, second best underlined.

Method	Fpr95Tpr ↓	AuPR ↑	AuRoc ↑	ACE ↓
Softmax [HG17]	63.5	95.4	80.1	0.633
Void [BSN+19]	68.1	92.4	75.3	0.499
AE [HG17]	92.1	88.0	53.1	0.832
MCDA [AB18]	61.9	95.8	82.0	0.411
Temp. Scale [GPSW17]	61.8	95.8	81.9	**0.287**
ODIN [LSL18]	60.6	95.7	81.7	0.353
ConfidNet [CTBH+19]	61.6	95.9	81.9	0.367
Gauss P [MAG+20]	61.3	96.0	82.5	0.384
Deep Ensemble [LPB17]	**60.3**	96.1	82.3	0.375
MCDropout [GG16]	61.1	96.0	82.6	0.394
ObsNet + *LAA*	**60.3**	**96.2**	**82.8**	0.345

To show where the uncertainty is localized, we outline the uncertainty map on the test set (see Figure 4.4). We can see that our method is not only able to correctly detect OoD objects, but also to highlight areas where the predictions are wrong (edges, small and far objects, etc).



(a) CamVid OOD

(b) BDD anomaly

(c) StreetHazards

Figure 4.4: **Error map visualization. 1st column**: We highlight the ground truth locations of the OoD objects to help visualize them (red bounding box). **2nd column**: Segmentation map of the SegNet. **3rd to 5th columns**: Uncertainty Map highlighted in yellow. Our method produces stronger responses on OoD regions compared to other methods, while being as strong on regular error regions, e.g., boundaries.

4.3.3 *Results: Robustness to Adversarial Attacks*

In safety-critical applications like autonomous driving, we know that the perception system has to be robust to adversarial attacks. Nevertheless, training a robust network is costly and robustness comes with a certain trade-off to make between accuracy and run time. Moreover, the task to only detect the adversarial attack could be sufficient as we can rely on other sensors (LiDAR, Radar, etc.). Although this work does not focus on Adversarial Robustness, empirically we note that ObsNet can detect an attack. To some extent this is expected as we explicitly train the observer to detect adversarial attacks, thanks to the LAA. In Table 4.8, we evaluate the adversarial attack detection of several methods. We apply a FGSM attack in a local square patch on each testing image. Once again, we can see that our observer is the best method to capture the perturbed area.

Method	Fpr95Tpr ↓	AuPR ↑	AuRoc ↑	ACE ↓
Softmax [HG17]	67.5	94.7	82.5	0.529
ConfidNet [CTBH+19]	58.4	96.4	86.8	0.462
Gauss P [MAG+20]	61.8	95.8	85.7	0.473
Deep Ensemble [LPB17]	63.9	96.5	86.4	0.468
MCDropout [GG16]	52.8	97.2	88.5	0.483
ObsNet + *LAA*	**42.1**	**97.7**	**91.4**	**0.423**

Table 4.8: **ObsNet Robustness on CamVid.** Error detection evaluation with random square attacks (best method in bold). This table shows the natural robustness of our Observer Network.

4.4 RESULTS: PUBLIC LEADERBOARD

To show the strength of our method, we submit our results on the SegmentMeIfYouCan [CLU+21][1] dataset. The dataset is solely composed of a test set of 100 images, containing OoD objects such as animal, carriage or caravan. We use the Deeplab v3+ with a Wide-ResNet 38 encoder [ZK16] pre-trained on CityScapes [COR+16] provided by the authors. We train an ObsNet for 20 epochs also on CityScapes dataset as explained in the method section.

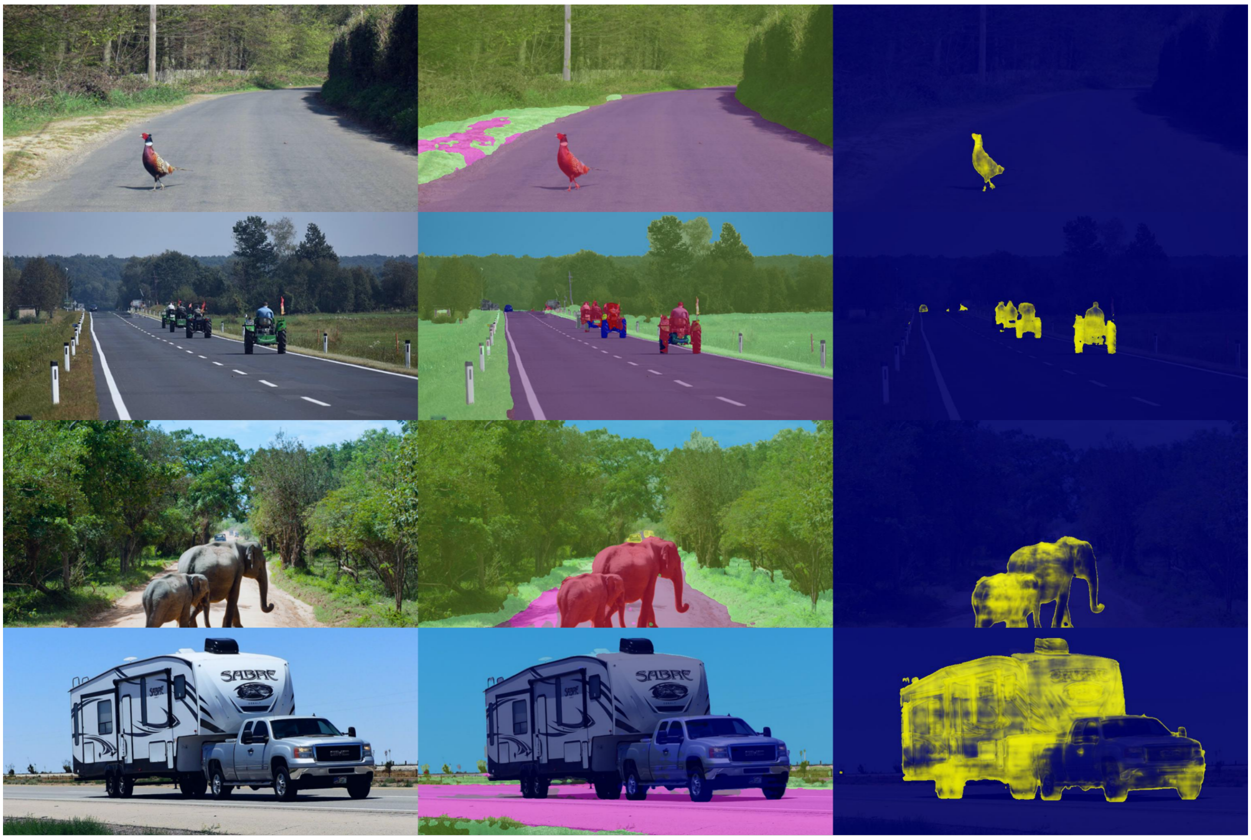
To obtain State-of-the-Art performance, we post-process the output from the observer. We apply a filter on the error map, to decrease the value on non object prediction. For every pixel different to person, rider, car, truck, bus, train, motorcycle, bicycle, we divide the value by a factor of 10. In analogy with panoptic segmentation, pixels corresponding to *things*, comes with higher priority than *stuff* classes, as shown on Figure 4.5. In Table 4.9, we show that our methods perform the best compared to methods that do not use OoD data in train by a large margin (+18.52% for AUPR). Moreover, we can see that our methods perform very well on an instance-wise metric. Without using any OoD sample in train, our methods get SOTA results on PPV and Meant metrics, by increasing the best results by +13.05% and +14.% respectively. (see Appendix A for details of the metrics).

4.5 CONCLUSION & LIMITATION

In this chapter, we proposed to use an observer network to address OoD detection in semantic segmentation, by learning from triggered failures. We use Local Adversarial Attacks to induce failures in the segmentation network and train the observer network on these samples. We show on three different segmentation datasets that our strategy combining an observer network with local adversarial attacks is fast, accurate and is able to detect unknown objects.

Nevertheless, error maps generated by the observer are not practical for downstream tasks. Indeed, pixel wise prediction shows limitation to localize precisely if there is an unknown object in the image. Prediction with high error on average could correspond to a crowd of in-distribution small objects. We show in the next chapter how to leverage an additional instance prediction to filter and aggregate the

[1] [CLU+21] Robin Chan et al., *Segmentmeifyoucan: A benchmark for anomaly segmentation*. In NeurIPS Datasets and Benchmarks 2021.



Input Image

Semantic Segmentation

Error Map

Figure 4.5: **Errors map.** We show our performance on the *Segment Me If You Can* dataset where we get SOTA performance among methods that do not use OoD data for training.

		Pixels-Wise Metrics		Instance-Wise Metrics		
Model name	Train w/ OoD	AUPR ↑	Fpr95Tpr ↓	sIoU gt ↑	PPV ↑	mean F1 ↑
ObsNet + *LAA*	✗	**75.44**	26.69	**44.22**	**52.56**	**45.08**
NFlowJS [GBKS21]	✗	56.92	34.71	36.94	18.01	14.89
JSRNet [VvA$^+$21]	✗	33.64	43.85	20.20	29.27	13.66
Image Resynthesis [LNFS19]	✗	52.28	**25.93**	39.68	10.95	12.51
Embedding Density [BSN$^+$19]	✗	37.52	70.76	33.86	20.54	7.90
Maximum Softmax [HG17]	✗	27.97	72.05	15.48	15.29	5.37
ODIN [LSL18]	✗	33.06	71.68	19.53	17.88	5.15
MCDropout [GG16]	✗	28.87	69.47	20.49	17.26	4.26
Ensemble [LPB17]	✗	17.66	91.06	16.44	20.77	3.39
Mahalanobis [LLLS18]	✗	20.04	86.99	14.82	10.22	2.68
DenseHybrid [GBv22]	✓	77.96	9.81	54.17	24.13	31.08
Maximized Entropy [CRG21b]	✓	85.47	15.00	49.21	39.51	28.72
Void Classifier [BSN$^+$19]	✓	36.61	63.49	21.14	22.13	6.49
SynBoost [DBBSC21]	✓	56.44	61.86	34.68	17.81	9.99

Table 4.9: **ObsNet performance on SegmentMeIfYouCan:** best methods on bold among the one trained without OoD

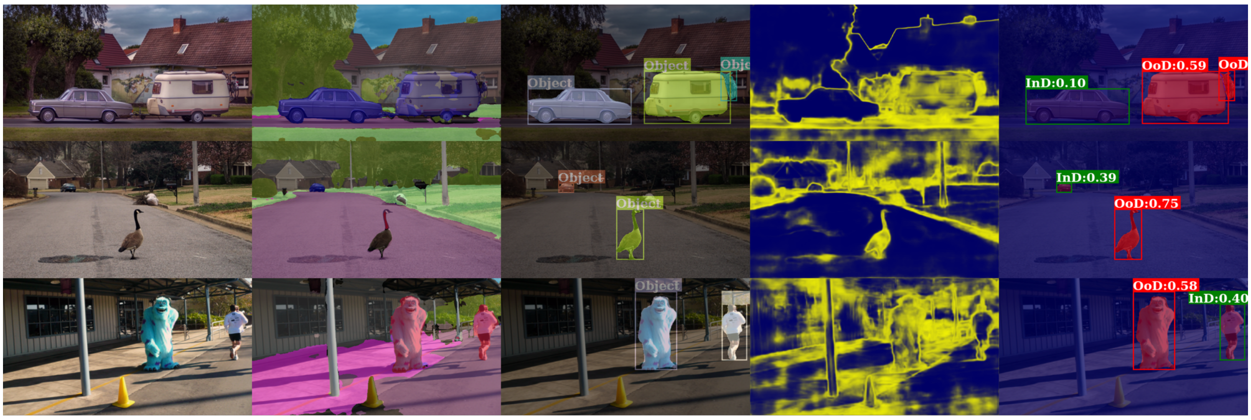
ObsNet output for better OoD object detection.

5

Instance-Aware Observer

◂ Chapter 4 Chapter 6 ▸

Synopsis In Chapter 4, we presented a powerful way to train the observer network with local adversarial attacks. We have shown promising results on Out-Of-Distribution (OoD) detection for semantic segmentation. However, these methods struggle to precisely locate the point of interest in the image, *i.e.,* the anomaly. This limitation is due to the difficulty of fine-grained prediction at the pixel level. To address this issue, we build upon the vanilla ObsNet approach by providing object instance knowledge to the observer. We extend ObsNet by harnessing an instance-wise mask prediction. We use an additional, class agnostic, object detector to filter and aggregate observer predictions. Finally, we predict an unique anomaly score for each instance in the image. We show that our proposed method accurately disentangles in-distribution objects from OoD objects on three datasets, as presented in Figure 5.1.



(a) Input Image (b) Semantic Seg (c) Instance Seg (d) Pixels-wise Error (e) Instance wise Error

Figure 5.1: **Qualitative results overview on SegmentMeIfYouCan [CLU+21].** Our framework uses the input image **(a)** to predict panoptic segmentation **(b)** and pixels-wise error map **(c)**. We then fuse **(b)** and **(c)** to produce the instance-wise error prediction **(d)**. We show on the right that we succeed to predict high anomaly score for OoD objects (*i.e.*, caravan, goose, and *monster*) and low anomaly score for in-distribution (*i.e.*, cars and pedestrians). We also filter high error background pixels.

5.1 Introduction

In this work, we aim at detecting OoD objects for 2D object segmentation. In this context, we consider as OoD the objects belonging to a class that is unknown by the perception system, *i.e.*, a class that is not defined nor present in the training data.

Recent works inspired by practices from system validation and monitoring, advance two-stage strategies to detect anomalies in semantic segmentation [BPB21, BBPB21][2] An Observer Network is trained

[2] [BBPB21] Victor Besnier et al., *Triggering Failures: Out-Of-Distribution detection by learning from local adversarial attacks in Semantic Segmentation*. In ICCV 2021.

to analyze and predict the confidence of a main perception network. Observer-based approaches have been shown to find a good balance between accuracy and computational efficiency [BPB21, BBPB21]. In this work we build on top of observer-based approaches to leverage their properties.

5.1.1 *From pixel-wise to instance-wise*

We argue that the pixel-wise error map (as shown in [BPB21, BBPB21, KG17, CTBH+19]) by itself is sub-optimal for anomaly detection in segmentation because these maps lack clarity. Due to the difficulty of fine-grained prediction, most boundaries between two classes or boundary of small or distant objects are considered uncertain. Therefore, the focus of interest in the image, *i.e.*, the OoD object, is drowned into this noise. The resulting error map does not provide precisely delimited spatial information: we know there is an error on the image but we struggle to accurately locate the corresponding OoD object. In other words, while we can get uncertainty estimates and predictions at pixel-level, extending them to objects is not obvious and we cannot automatically find objects far from the training distribution. As an example, an image depicting a crowd of pedestrians with a lot of boundaries has, on average, higher anomaly score than an image with only one OoD object in the middle of a road.

In this chapter, we propose to reduce the granularity of the task in order to improve the relevance of the error map. To this end, we use a class agnostic, instance segmentation network. With this additional prediction, we first filter background errors, and then aggregate uncertainty in an instance-aware manner Figure 5.2. Ultimately, we only want to highlight object instances with high errors. With this pragmatic and practical solution we can sort objects by anomaly score and then discard all objects close to the training distribution and keep those that are far from the training distribution.

5.2 METHOD

5.2.1 *Observer Networks*

Our work builds upon ObsNet [BPB21, BBPB21] that we briefly describe here. Observer networks are a two-stage method to detect pixels-wise errors and OoD. [BBPB21] designed two principles to train efficiently an auxiliary network. They improve the architecture by decoupling the OoD detector from the segmentation branch and by observing the whole network via residual connections. Secondly, they generate blind spots in the segmentation network with *local adversarial attacks* (LAA) at a random location of the image, mimicking an OoD object. ObsNet (Obs) outputs a pixels-wise error map corresponding to the probability that the semantic segmentation network (Seg) fails

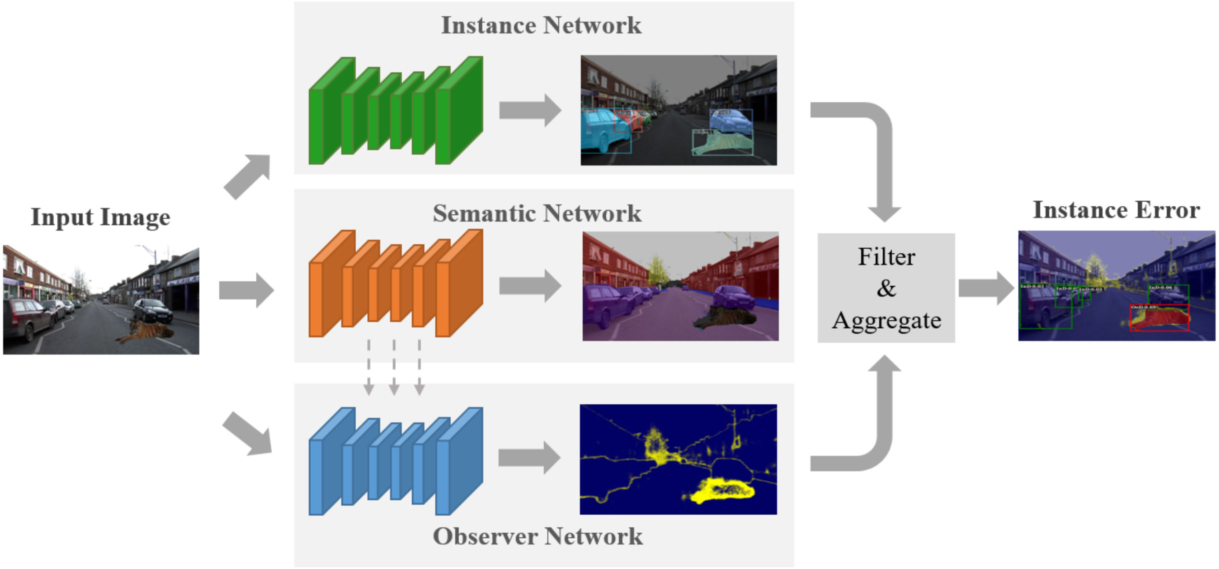


Figure 5.2: **Overview of our instance aware pipeline.** The image is fed into the instance, the semantic and the observer network (from top to bottom). In the middle, the ObsNet prediction is filtered by the class-agnostic instance prediction and the remaining error is then aggregate object-wise. Finally, in the right of the figure, we show in red, the objects far from the training distribution.

to predict the correct class y:

$$Obs(x, Seg_r(x)) \approx Pr[Seg(x) \neq y], \tag{5.1}$$

where x is the input image and Seg_r the skip connections from intermediate feature maps of segmentation network Seg.

5.2.2 *Instance Anomaly Detection*

To this end, we upgrade the semantic segmentation framework with instance hints. We use an additional class agnostic instance segmentation prediction. This detector (*Det*) produces a binary mask by mapping each object in the image.

Then, the idea is to separate the observer's prediction map into two categories. The background (classes of *stuff*) and the instance (classes of *things*) in the same way as in panoptic segmentation. Background errors correspond to global ambiguities in the scene at different scales: error at the decision boundary between two classes, prediction error between the road and the sidewalk or complexity of the leaves of a tree. In contrast, an instance error corresponds to an object far from the train distribution.

5.2.3 *Error Aggregation and Filtering*

In order to obtain a unique error score for each instance (similar to the well-known objectness score in object detection), we aggregate the per-pixel uncertainty within the predicted object mask to a unique value. In practice, given an image $x \in \mathbb{R}^{3 \times H \times W}$, we predict for each detected object o_i an anomaly score $a_i \in \mathbb{R}$:

$$a_i = \frac{1}{M} \sum_{h=0}^{H} \sum_{w=0}^{W} u^{(h,w)} \odot m_i^{(h,w)}, \tag{5.2}$$

where $u = Obs(x, Seg_r(x)) \in \mathbb{R}^{H\times W}$ is the pixel-wise error map of ObsNet; $m_i \in \mathbb{R}^{H\times W}$ is the binary mask of an instance o_i in the set of the detector prediction $Det(x) = \{m_i\}$; $M = \sum_{h,w=0}^{H\times W} m_i$ the area of the instance o_i; and $\odot$ is the element-wise product. We also filter predicted instance masks m_i by size, in order to remove very small detected objects ($< 16^2$ pixels) in the image.

This strategy shows several benefits. We can discover instances in the dataset that do not match with the training distribution, useful for active learning or object discovery. We can also localize where the anomaly is in the image, which is a primary requirement for safety-critical applications such as autonomous driving. In Figure 5.3, we show that our framework is able to detect several instances in the images, and the ObsNet succeeds in discriminating in-distribution objects from out-of-distribution ones.

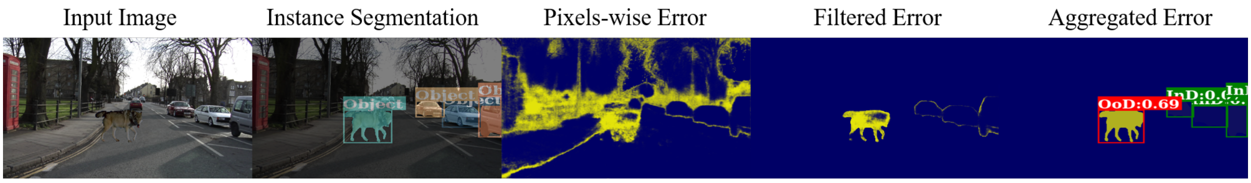


Figure 5.3: **Flows of the image processing.** From the input image (left), we compute the pixels-wise uncertainty and the object detection masks. Then we filter the uncertainty in the area of the object only, and finally aggregate the score in an instance aware manner. We can see that the OOD object is well detected while in-distribution objects with low anomaly score and background errors are erased (right).

5.3 EXPERIMENTS

We assess experimentally the effectiveness of our observer network coupled with an class-agnostic instance detector and compare it against several baselines.

5.3.1 *Datasets, Metrics & Compared Methods*

We conducts experiments on the **CamVid OoD** [BPB21], **StreetHazards** [HBM+19] and **BDD Anomaly** [YCW+20] datasets of urban streets scenes with anomalies in the test set. Anomalies correspond to OoD objects, not seen during training.

To evaluate each method on these datasets, we select four metrics to detect misclassified and out-of-distribution examples: **Fpr95Tpr** [LSL18], **Area Under the Receiver Operating Characteristic curve (AuRoc)** [HG17], **Area under the Precision-Recall Curve (AuPR)** [HG17] and **Mean Average Prediction (mAP_δ)**. We compute the latter metric where we discard object smaller than δ^2 pixels. For example, mAP_{32} in Table 5.2, mean that we do not take into account any object smaller than 32^2 (in distribution or OoD) in the dataset.

For each metric, we report the result where an object is considered well detected if the predicted mask has $IoU > .5$ with the ground truth. We assign to each detected object the anomaly score computed as Equation 5.2. We use a Bayesian SegNet [BKC17], [KBC15] as the main network for CamVid and a DeepLabv3+ [CZP+18][1] for BDD Anomaly and StreetHazards. The ObsNet follows the same architecture as the corresponding segmentation network.

For our instance segmentation module, we select two Mask R-CNN variants [HGDG17][2]: one trained on CityScapes [COR+16], reported as In-Distribution Detector, and one trained on MS-COCO [LMB+14], reported as Pan-Distribution. We do not leverage the class predicted but only the instance mask. Moreover, we use an additional oracle: we take every connected region of the same class in the annotation as one instance of an object, and we report this *detector* as GT-detector. The latter detector is a **perfect** detector. Indeed, it will detect every OoD object thanks to the GT. Then, the Observer will be evaluate on every object in the scene. Obviously, this detector is not available in practice, but is used, here, as a upper bound.

We compare our method against two other methods. **MCP** [HG17]: Maximum Class Prediction; one minus the maximum of the prediction. And **MCDropout** [GG16]: The entropy of the mean softmax prediction with dropout; we use 50 forward passes for all the experiences.

5.3.2 *Benefit of the instance module*

To validate the benefit of the instance detector, we first check that filtering the pixel-wise error map with the instance detector helps for pixel OoD detection, see Table 5.1. Using an instance detector significantly improves the performance of ObsNet. The GT-Detector show that our ObsNet is nearly perfect to detect OoD only, with a Fpr95Tpr a 1%. Moreover, this experiment shows that keeping a raw error map is sub-optimal because many pixels with high anomaly score do not correspond to an OoD object but actually belong to the background of the images, whereas they can easily be filtered out by our instance scheme.

Method	Fpr95Tpr↓	AuPR_error ↑	AuRoc ↑
Softmax [HG17]	70.0	11.45	76.7
ObsNet	40.5	22.72	87.9
ObsNet + *in-detector*	31.3	49.4	92.3
ObsNet + *pan-detector*	8.7	70.1	97.3
ObsNet + *gt-detector*	**1.0**	**90.5**	**99.7**

Table 5.1: **Pixel-wise evaluation on CamVid OoD**. We consider OoD pixels only as the positive class.

5.3.3 *Instance-Wise Results*

Here, we compare our methods for object detection on Table 5.2, Table 5.3 and Table 5.4. We can observe that for each dataset the results are quite different. This is due to the scale of the anomalies and the number of them in each dataset. In Table 5.2, all anomalies are above 64^2 pixels, which can explain why the metrics drastically

[1] [CZP+18] Liang-Chieh Chen et al., *Encoder-decoder with atrous separable convolution for semantic image segmentation*. In ECCV 2018.

[2] [HGDG17] Kaiming He et al., *Mask R-CNN*. In ICCV 2017.

improve as we discard smaller detected objects. In Table 5.4, most of the objects are in fact anomalies, which is why mAP is high, even for the object below 32^2 pixels. Finally, even if on average *pan-detector* outperforms *in-detector*, this is not always the case in Table 5.3. Indeed, *pan-detector* can detect more objects, and among them smaller in-distribution objects, that can hurt performances. Overall, ObsNet outperforms baseline methods, regardless of the detector.

Det	Method	Fpr95$_{32}$ ↓	Roc$_{32}$ ↑	mAP$_0$	mAP$_{32}$ ↑
In	Softmax [HG17]	*	56.7	7.7	55.2
	MCDropout [GG16]	*	58.3	8.4	58.5
	ObsNet	*	**60.5**	**9.9**	**63.7**
Pan	Softmax [HG17]	57.2	79.4	4.8	62.1
	MCDropout [GG16]	52.8	84.6	6.9	70.8
	ObsNet	**46.4**	**89.6**	**11.3**	**81.4**
GT	Softmax [HG17]	43.3	80.4	10.8	72.1
	MCDropout [GG16]	32.1	85.5	13.5	79.2
	ObsNet	**27.2**	**94.3**	**22.3**	**92.0**

Table 5.2: **Instance-wise evaluation on CamVid OoD**. We consider OoD examples only as the positive class. *not enough OoD objects have been detected by the detector to compute the metrics

Det	Method	Fpr95$_{32}$ ↓	Roc$_{32}$ ↑	mAP$_0$ ↑	mAP$_{32}$ ↑
In	Softmax [HG17]	*	52.5	9.5	13.3
	MCDropout [GG16]	*	52.6	9.5	13.3
	ObsNet	*	**55.8**	**9.9**	**16.8**
Pan	Softmax [HG17]	*	63.0	5.9	16.8
	MCDropout [GG16]	*	62.2	6.0	17.0
	ObsNet	*	**64.5**	**6.9**	**20.1**
GT	Softmax [HG17]	65.6	81.9	22.7	37.8
	MCDropout [GG16]	61.9	82.5	23.0	39.1
	ObsNet	**53.3**	**86.8**	**27.1**	**50.7**

Table 5.3: **Instance-wise evaluation on BDD Anomaly**. We consider OoD examples only as the positive class.

Det	Method	Fpr95$_{48}$ ↓	Roc$_{48}$ ↑	mAP$_{16}$	mAP$_{48}$ ↑
In	Softmax [HG17]	*	**50.4**	80.0	**81.9**
	MCDropout [GG16]	*	50.3	80.0	81.2
	ObsNet	*	**50.4**	**80.1**	**81.9**
Pan	Softmax [HG17]	*	53.7	57.6	77.7
	MCDropout [GG16]	*	53.6	**57.7**	77.6
	ObsNet	*	**54.1**	56.9	**77.8**
GT	Softmax [HG17]	80.0	85.2	88.9	99.0
	MCDropout [GG16]	74.5	86.0	86.9	99.0
	ObsNet	**72.6**	**87.5**	**89.0**	**99.2**

Table 5.4: **Instance-wise evaluation on StreetHazards**. We consider OoD examples only as the positive class.

In Figure 5.4, we report the histogram of objects detected by our detector, ranked by our framework. We can well disentangle in-distribution objects as cars, bicycles, or pedestrians, from OoD objects (animals).

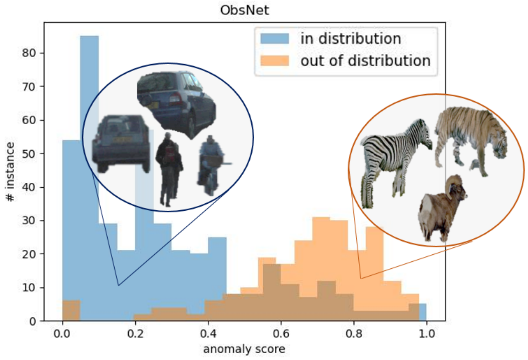


Figure 5.4: **Histogram on CamVid OoD**. Anomaly score from obsnet and detection from mask RcNN trained on the pan-distribution. We show here some examples of well-detected objects predicted as in distribution in blue (left). While objects detected with high anomaly scores (right) are considered as OoD in orange.

We illustrate a few qualitative results in Figure 5.5. ObsNet emphasizes OoD objects with higher anomaly scores compared to in-distribution objects. Its predicted error maps are generally clearer and more accurate.

5.3.4 *Object size*

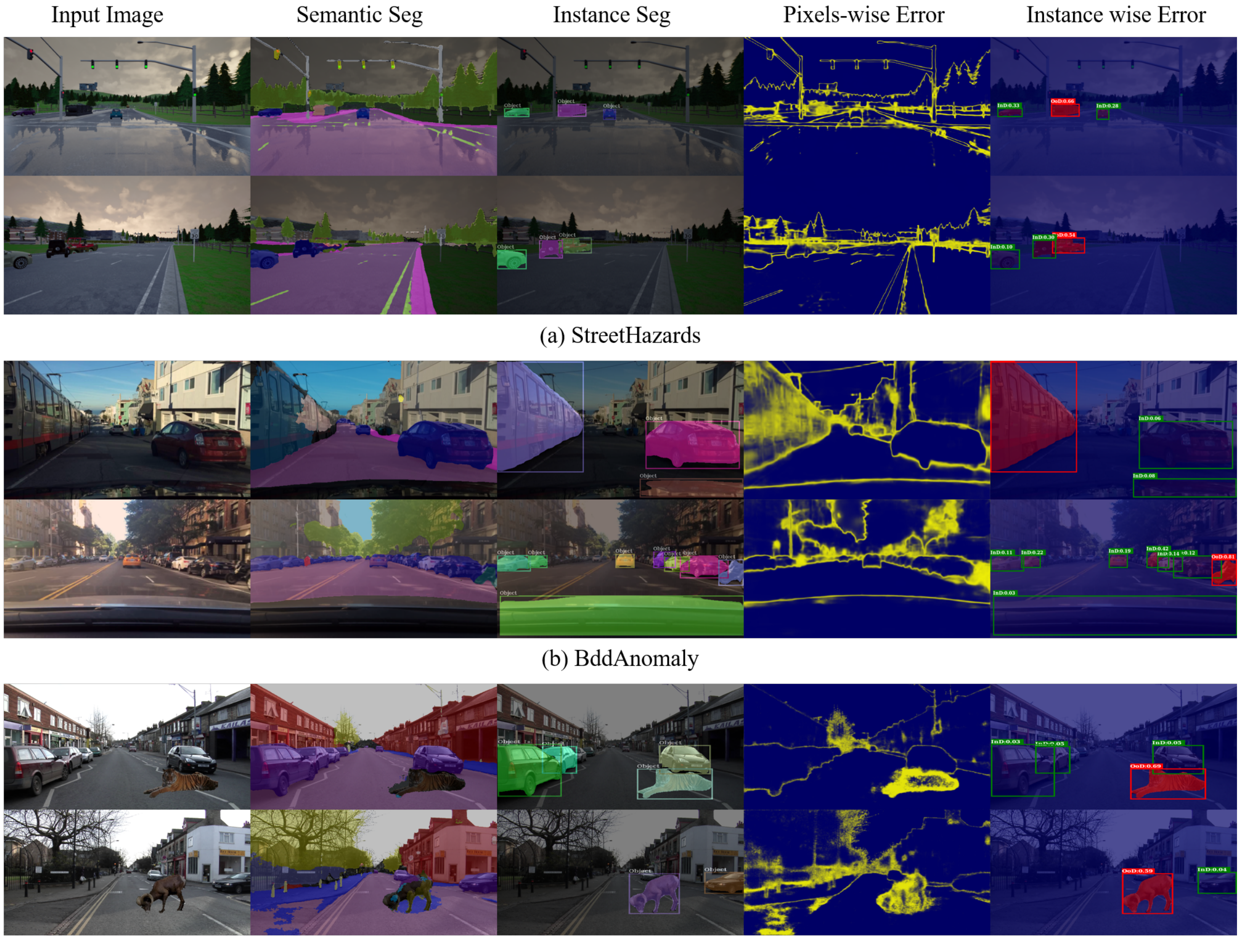
One drawback of the method remains the size of the anomaly object. Interestingly, small or far objects are often considered as an error, whereas only part of them are real anomalies. This can be explained because the contours of each object are often considered as errors at the pixel level and thus most of these small objects are flagged as anomalies. In Figure 5.6, we show the bounding boxes detected and classified by our framework. We see that most of the small objects, far from the camera, have high uncertainty. When we filter objects below a threshold, these false positives tend to disappear.

This is of course a limitation for datasets that contain only small anomalies like LostAndFound [PRG+16], where anomalies are small obstacles on the road. In our setting, we cannot discard objects below a threshold on the size because all real anomalies would be mostly below that threshold and thus indistinguishable from small objects with uncertain contours.

5.4 CONCLUSION

In this chapter, we propose to use an additional, class-agnostic, object detector to filter and aggregate an anomaly score from ObsNet pixel-wise error map. Our strategy helps to better disentangle in-distribution from out-of-distribution objects.

One limitation is the size of the instance detected, as small and far object are usually detected as OoD by our framework. One idea could be to normalize on the size of the object (*i.e.*, divide by the size of the bounding box). But we let this for future works. In the next chapter, we show how to integrate and observer network and a segmentation network in a DemoCar.



(a) StreetHazards

(b) BddAnomaly

(c) CamVid OoD

Figure 5.5: **Qualitative results on the StreetHazards (top row), BDD Anomaly (mid) and CamVid OoD (bot).** We can see that our method is able to detect numerous objects and disentangle in-distribution objects ($a_i < .5$ in green) from out-of-distribution objects ($a_i > .5$ in red).

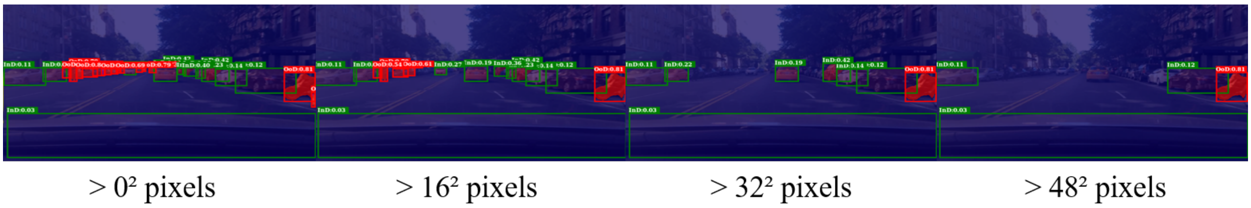


Figure 5.6: **Visualization of the objects detected on Bdd Anomaly.** All motorcycles are OoD. We filter each detection by the size, from left with no filtering to right where we filter every object below 48^2 pixels. In this example, we see that at 32^2 we succeed detecting many objects and classify only the motorcycle as OoD.

Part III

LEARNING CAR

6

Learning Car Project

◂ Chapter 5 Chapter 7 ▸

Synopsis Through this chapter, we present how we integrate an ObsNet in a demo car. We implement a proof-of-concept for our previous works on observer networks to show that our research helps to build safer applications. We show step by step how to embed neural networks in a vehicle. We use a Nvidia Jetson Xavier for network inference, and RTMaps for orchestrating every component, from the image capture to the results display. We evaluate our solution in a real world setting, driving in Créteil, France, with a Demo Car (cockpit point of view Figure 6.1 to show that our ObsNet is able to detect hazards on the road. Without extensive effort on network optimization, we show that our methods run at 8 FPS to perform both image semantic segmentation and pixel-level error detection. The solution allows to safely detect anomalies at slow speed such as parking or dense urban scenarios.

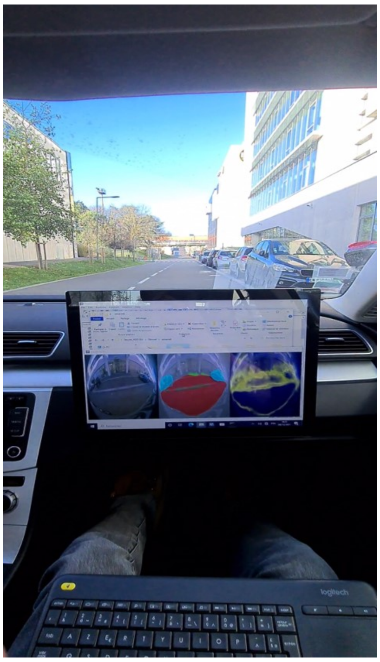


Figure 6.1: **Learning Car**: View from inside the vehicle with real-time segmentation and anomaly prediction.

6.1 SAFETY CRITICAL APPLICATION

6.1.1 *Automated Car*

Many industrial actors aim to develop cars able to drive from a location A to a destination B without human intervention. We can split the principal leaders into two different sectors: historical automakers such as Mercedes, Toyota, Volkswagen or Tesla, and service providers like Waymo, Uber or Cruise.

One important thing to understand is the different levels of driving automation. Jointly with the Organization for Standardization (ISO), the Society of Automotive Engineers (SAE) developed a standard with 6 levels of automation. Level 0 does not provide any autonomous features but only warning or momentary assistance. Level 1 proposes help for steering wheel or speed control, while level 2 can monitor both features (*i.e.*, speed and trajectory assistance). For the first three SAE levels, the driver must supervise the features of the car. Level 3 proposes advanced autonomous driving assistance, where the driver only needs to overtake the control when the system requests it, like a traffic jam chauffeur. Level 4 does not need any driver but can operate only in a constrained environment. Finally, level 5 of driving automation provides a full autonomous driving system able to drive in all conditions.

To better understand how to build an autonomous car using a

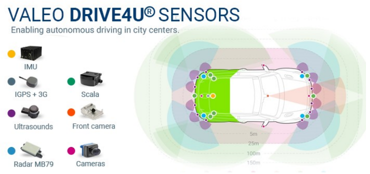


Figure 6.2: **Car sensors**: Valeo's Drive4U sensor, allowing the vehicle to *see* the world.

perception system, we show in Figure 6.3 the different steps to consider. The system is split under 7 steps:

1. Sensors: composed of different captors such as cameras, LiDAR, radar, etc. They aim at capturing raw data of the environment.
2. Localization: gather coordinate of the car.
3. Perception: understand what objects are around the car.
4. Scene Modeling: depict the scene around the car, *i.e.*, where are the other cars, pedestrians, where can I ride.
5. Prediction: Anticipate next actions of other vehicles, *i.e.*, where are other cars going, does this pedestrian cross the road.
6. Planning: develop an action plan to move from a location to another.
7. Control: finally, operate the physical actions to drive.

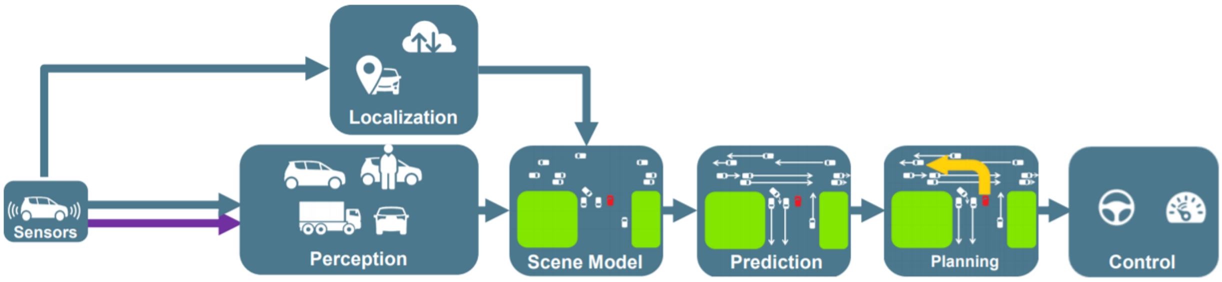


Figure 6.3: **End-to-End Autonomous Driving**: From the sensor to the final control of the car. We show here every step to consider to have a good representation of the environment.

Our research mainly covers the Perception and Scene Modeling parts. One common way to complete this tasks is to use sensor data (radar, LiDAR or images) and neural networks for scene understanding (Image segmentation [MFW16], depth estimation [GMFB19] or object detection [GLW$^+$21]). On our side we solely rely on images. We want to monitor a segmentation network with our ObsNet. The semantic segmentation network perceives objects and models the scene around the car. While our ObsNet helps to better trust the decision process of the segmentation network. Errors in the modeling of the scene can impact the prediction step and the planning step, resulting in wrong control of the car. In the literature, semantic segmentation as been widely used for autonomous driving

6.1.2 *Proof-of-concept for autonomous Driving*

Here we dig in more details on a few projects that use neural networks for controlling the car. One of the first projects involving deep neural networks and autonomous driving was Alvinn [Pom88][1]. The goal was to control the angle of the steering while given the camera and laser input. They used a shallow neural network composed of 3

[1] [Pom88] Dean A. Pomerleau. *Alvinn: An autonomous land vehicle in a neural network*. In NeurIPS 1988

hidden layers to navigate, (see Figure 6.4). The neural network localizes the free space where the car can drive and directly predicts the steering angle to keep the car in the middle of the road.

More recently, Nvidia developed the DAVE-2 project [BTD+16][1] in 2016 (following, four years later with PiloNet project [BCD+20]) with the same objective: End to End Learning for Self-Driving Cars. They use a neural network much deeper than Alvinn one, functioning at 30FPS to learn the trajectory to follow. They propose a complete pipeline on data collection, training and off-line simulation. Finally they show impressive results on different weather, speed and traffic conditions.

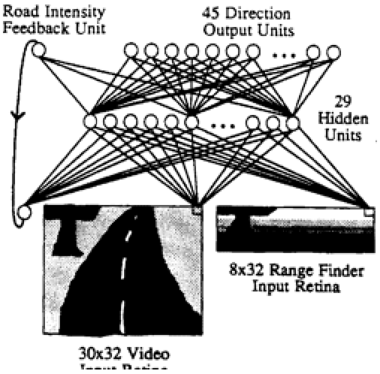


Figure 6.4: **ALVINN**: architecture of the model that can predict the angle of the steering wheel.

Inspired by previous work on deep learning, computer vision and autonomous cars, we decided to evaluate ourselves our ObsNet on a real situation. Contrary to previous works on end to end control of the car, we adopt a setup described in Figure 6.3 where our networks focus on the scene modeling and the perception. Thus our networks do not control the car, but show a representation of the context around it. This thesis is a collaboration between industry (Valeo) and academia (ENPC and ENSEA), thus we had the opportunity to make the proof of concept of our works on a demo car. Thanks to ENSEA, we get a Nvidia Jetson Xavier to embed our networks. Valeo provided us with a car equipped with cameras.

In the following section, we describe how we integrate an observer network and a segmentation target network presented in previous chapters on a demonstrator car.

6.2 OBSNET TRAINING

6.2.1 *Dataset*

To begin with, we describe here the dataset used to train our models off line. We use a Valeo internal dataset, sample shown on Figure 6.5. The dataset contains fisheye driving scene images in the wild from four different views around the car (rear, left, right, front), allowing you to get a 360° view. The images have been collected in different cities in Europe and America and contain numerous complex situations.

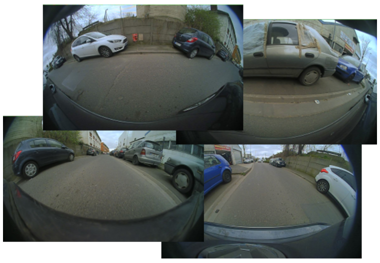


Figure 6.5: **CEA Dataset**: Fisheye images from our internal dataset.

The dataset is partially annotated for 3D object detection, depth estimation and semantic segmentation. Here, we solely use the segmented part. Eleven classes are available: road, side-walk, object, ground marking, etc. The training set is composed of 388 videos and the testing contains 112 sequences. Each sequence has 150 frames, thus, we gather a total of 75000 images: 58200 images for training, and 16800 for testing.

6.2.2 *Networks*

Our networks will be integrated in a limited resources embedded computing platform, we explain what change we made to fit our mod-

[1] [BTD+16] Mariusz Bojarski et al., *End to end learning for self-driving cars*. ArXiv, 2016.

els to the targeted hardware execution constraints. State-of-the-art architectures (Transformer) usually need a large quantity of memory and compute power that is not available for embedded systems. Compared to Chapter 4, we reduce the architecture complexity of the ResNet encoder and decrease of the image resolution.

Regarding the segmentation part, we train a Deeplabv3+ with a ResNet50 encoder[1] for 50 epochs; we used SGD with momentum as the optimizer. We used a batch-size of 8 images, resized to (512x1024), on four GPUs GeForce RTX 2080 Ti. Our ObsNet follows the same architecture, trained for 20 epochs, SGD momentum with batch-size of 4 images. The observer is trained as described in Chapter 4. During inference, we use half precision (float 16) for faster inference on both networks[2].

6.2.3 *Qualitative Results*

To evaluate the primary results, we gather additional data containing un-seen objects or objects without associated class for the segmentation network such as construction cones or wooden pallets. We show on Figure 6.6 that our solution is able to raise uncertain predictions (in yellow, bottom row). We also show an example where the segmentation misclassifies the side-walk. This step helps to validate that our models perform accurately on corner case examples.

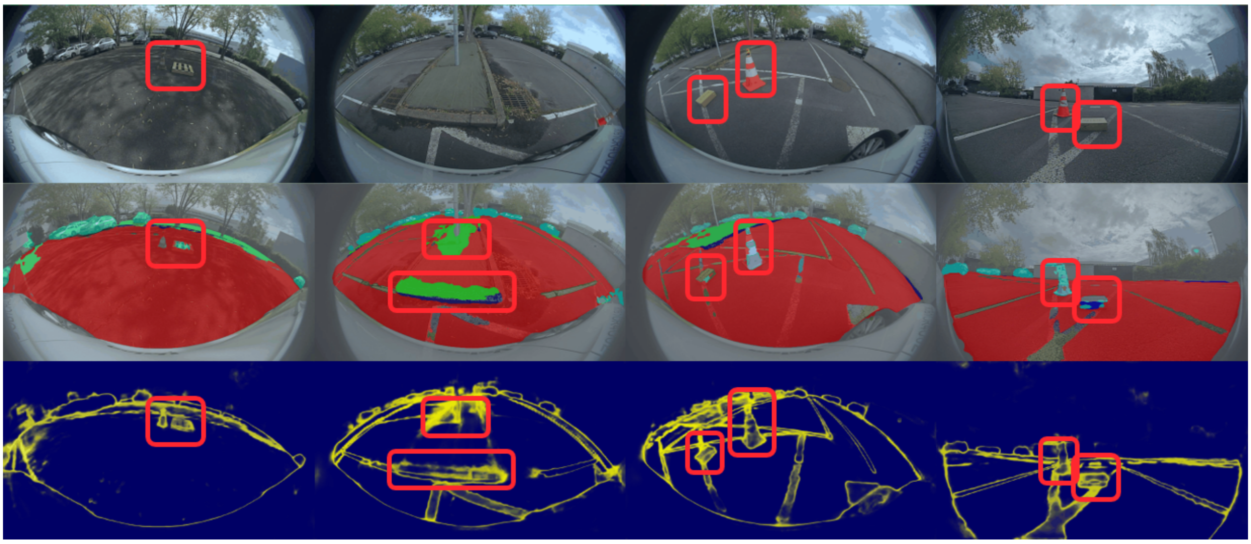


Figure 6.6: **Off-line validation**: Our ObsNet is able to detect (third row) building plot, wooden pallet and errors of the segmentation networks (second row).

6.3 EXPERIMENTAL SETUP

6.3.1 *Demo Car*

To embedded neural networks for autonomous vehicles, the first concern is the car. For the project, we used a Volkswagen Passat. The car is equipped with multiple sensors in order to *see* the world

[1] 2 times lighter than in Chapter 4

[2] default precision is float 32 in Pytorch models

around it. Long-range fisheye cameras, LiDAR, front, side and rear cameras and radar. These sensors allow the car to have a 360° field of view. Here, we leverage on the fisheye cameras, for semantic segmentation.

For application monitoring, the car is equipped with a computer running on Windows stored in the trunk. Moreover, in the passenger seat, a screen is available to display application results, as show in Figure 6.7.

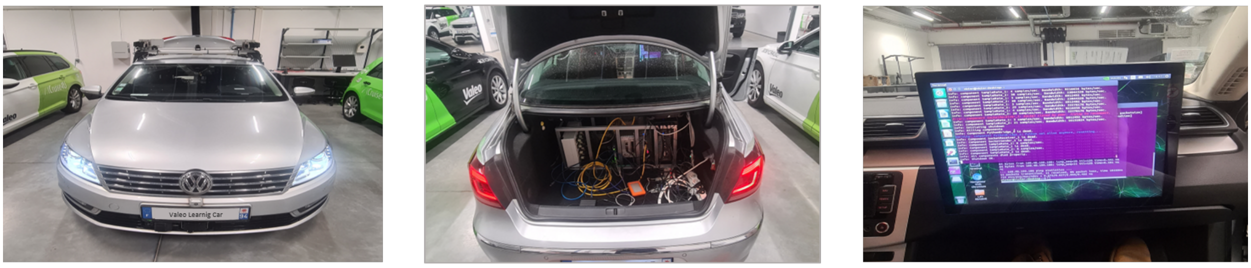


Figure 6.7: **Car overview**: On left the front view of the car, on the middle the back of the car with the Host PC and the Nvidia Jetson, and on the right the indoor monitor.

6.3.2 *Nvidia Jetson Xavier*

The second step to consider is the hardware that will run the model at inference. We gather a Nvidia Jetson Xavier (Figure 6.8), which is an embedded platform for ML acceleration running on Linux distribution. The card is a complete System on Module (SOM) that includes a GPU and a CPU which boost neural networks execution to increase inference throughput.

The device has two main advantages: first, it uses low power consumption. The autonomous brick of the car should use as less power as possible as the primary objective is to ride and each device that takes power of the battery diminishes the capability to ride further. The second is the physical size of the module, the Nvidia Jetson can be easily integrated in many embedded systems, specifically cars.

On the performance side, our Xavier is equipped with 512-core Volta GPU with Tensor Cores and 8-core ARM CPU. And has a power consumption under 30W. It is under these constraints that we build our model architecture in order to get the best throughput (Frame Per Seconds (FPS)) at inference time.

Figure 6.8: **Nvidia Jetson Xavier**: Hardware for accelerating ML applications.

6.3.3 *Orchestra conductor: RTMaps*

Once we have integrated pre-trained neural networks in an embedded system, such as a Nvidia Jetson. We seek to have a complete system where we gather all images from the camera, feeding them to the model and display the results. To orchestrate this workflow, we use a specific framework: RTMaps Figure 6.9. The software is a component based software. The main goal of RTMaps is to connect different bricks. Where each block is responsible for one action.

We split the solution into a Slave and and Host. Where the Host is a PC that manages all the system flows and the Slave executes what the Host command. Host and Slave could connect through RTMaps. In our setting, our slave is the Nvidia Jetson Xavier that will perform only the models inferences, and the host is a regular PC that will collect images, synchronize them, send them to the slave, and display the results.

Sequentially, we have the following execution, illustrated in Figure 6.10:

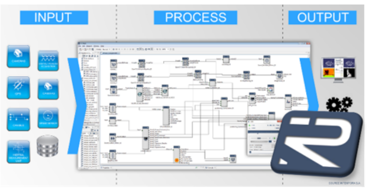


Figure 6.9: **RTMaps**: component synchronization.

1. **(Host)** Get raw image input for all four sensors, and synchronize them.
2. **(Host)** Pre-process the images: resized, formatted to the right format.
3. **(Host)** Send the image to the Nvidia Jetson.
4. **(Slave)** Python brick to execute the model inference.
5. **(Slave)** Send the results to host.
6. **(Host)** Display results
7. **(Host)** Record results

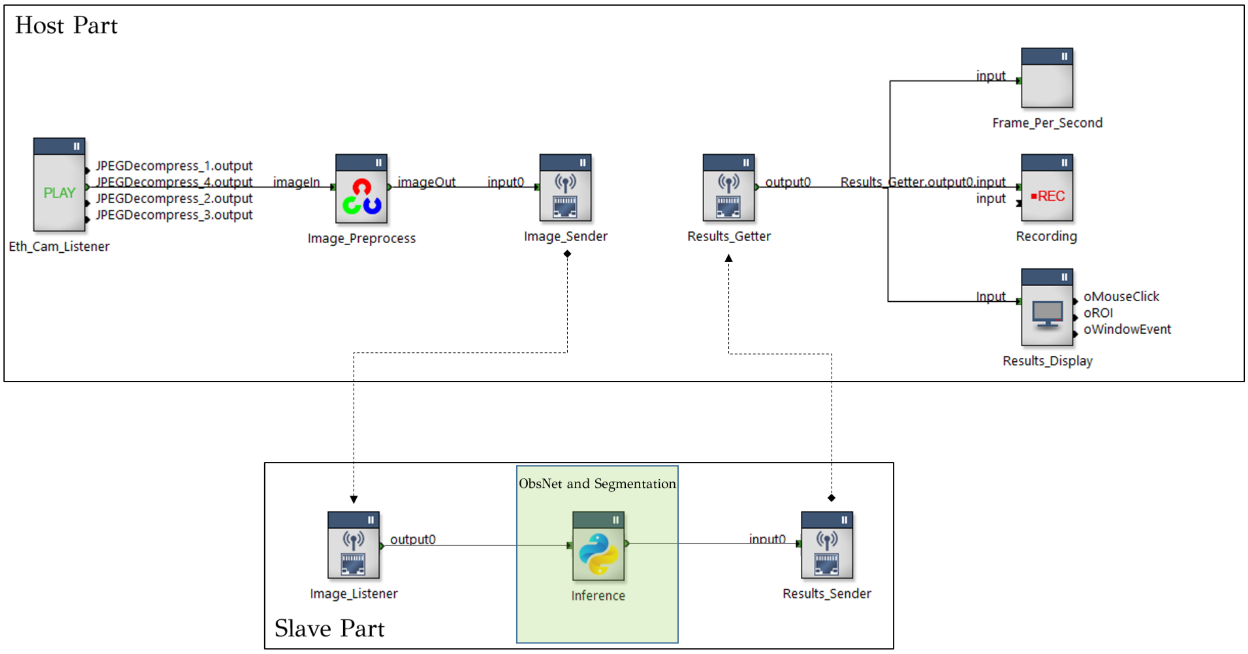


Figure 6.10: **RTMaps Final Diagram**: Full process of an image, from image gatherer to the recording and the display.

Before trying a solution on road, we simulate off-line a scenario with pre-recorded images. This step helps to detect bugs, and evaluate the performance of the model (fps, accuracy, etc). We show an example of the setting in Figure 6.11.

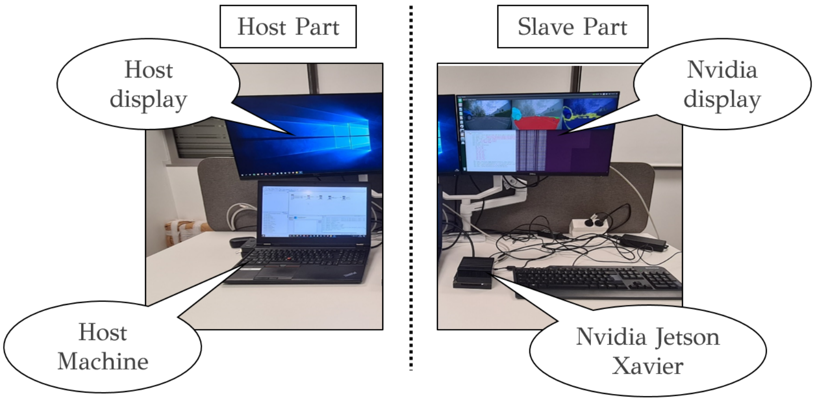


Figure 6.11: **Off-line simulation**: Before jumping into the car, we evaluate the framework off-line with pre-recorded video.

6.4 REAL-WORLD TESTING

6.4.1 *Road Evaluation*

We tested our model close to the Valeo building, in Créteil, France, near Paris see Figure 6.12. The city presents numerous challenging situations: parking exit, traffic circle and traffic sign, or construction site. Moreover, we drove in broad daylight, with many pedestrians and other cars, buses and trucks driving around us. The session lasts around 3min to complete.

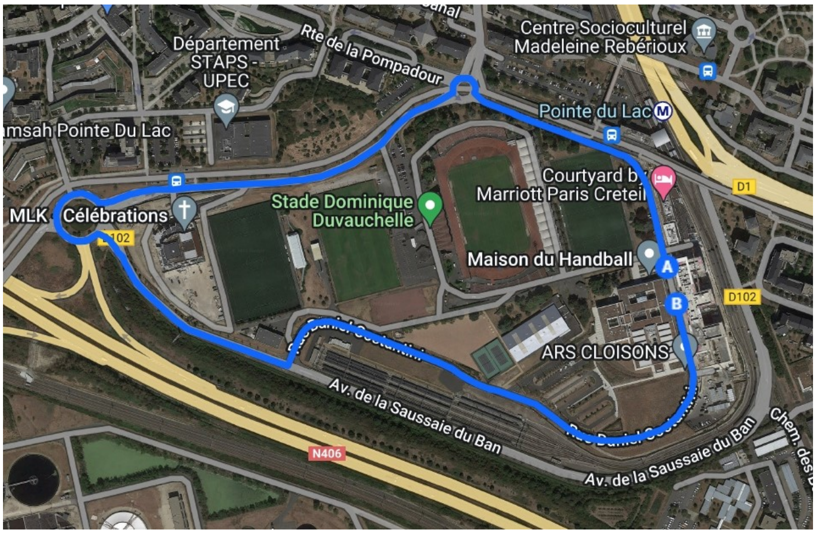

Département STAPS - UPEC
Rte de la Pompadour
Centre Socioculturel Madeleine Rebérioux
Pointe du Lac
msah Pointe Du Lac
Courtyard by Marriott Paris Creteil
MLK
Célébrations
Stade Dominique Duvauchelle
Maison du Handball
A
B
D1
D102
ARS CLOISONS
Av. de la Saussaie du Ban
N406
Av. de la Saussaie du Ban

Figure 6.12: **Créteil**: Area where we test our ObsNet framework covering numerous challenging situations.

In Figure 6.13, we show how our observer network and our segmentation network perform. ObsNet succeed in detecting regular errors, such as misclassification of a public trash, or stain on the road. We can notice that our ObsNet output overreacts on lane marking. Our solution runs at 8 FPS which is acceptable for low speed like parking areas or dense urban scenes. Even if our solution could show some limitations on high speed situations, we argue that other methods like ensemble are much more time consuming. Indeed, better hardware capacity will ultimately increase the throughput.

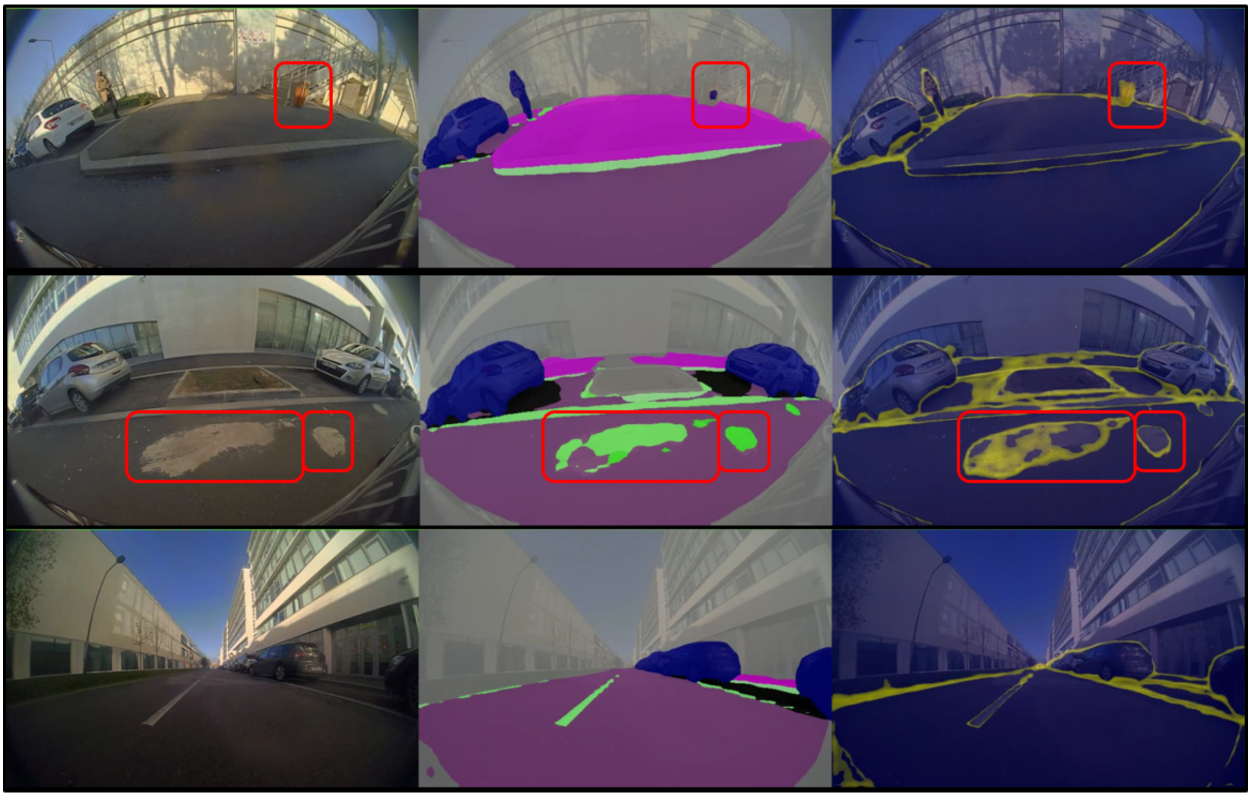


Figure 6.13: **Online test**: Example of classical errors our ObsNet is able to detect, from wrong text prediction on the road to trash detected as an object like a pedestrian.

We show in Figure 6.14 that our observer is also able to detect mispredictions on the car due to sun glare. Indeed, the prediction is confident where the car is well segmented, but as soon as the car becomes segmented as background, the observed output has high error score in the area.

6.4.2 *Loop Improvement*

Testing an application is an incremental process, there is still room for improvement. For the project we did three rounds. Each time improving either the quality of the prediction or the frame rate. Here are a few updates on what we did.

We know that our ObsNet is likely to predict error on the boundary of two classes. Thus we merged all ground marking classes to reduce the number of class boundaries and we retrain both networks. For the second round, we also used half precision, which doubles the frame rate (from 4 FPS to 8 FPS).

For the third and last iteration, we tried a vanilla domain adaptation technique to improve prediction accuracy. Indeed, we noticed that our dataset does not overlap with the operating domain, *i.e.*, Créteil area. Thus, thanks to the previous recording sessions, we updated the batchnorm statistics [BJB+20][1] (running mean and variance of each normalization layers) to match the targeted distribution.

[1] [BJB+20] Victor Besnier et al., *This dataset does not exist: Training models from generated images*. In ICASSP 2020

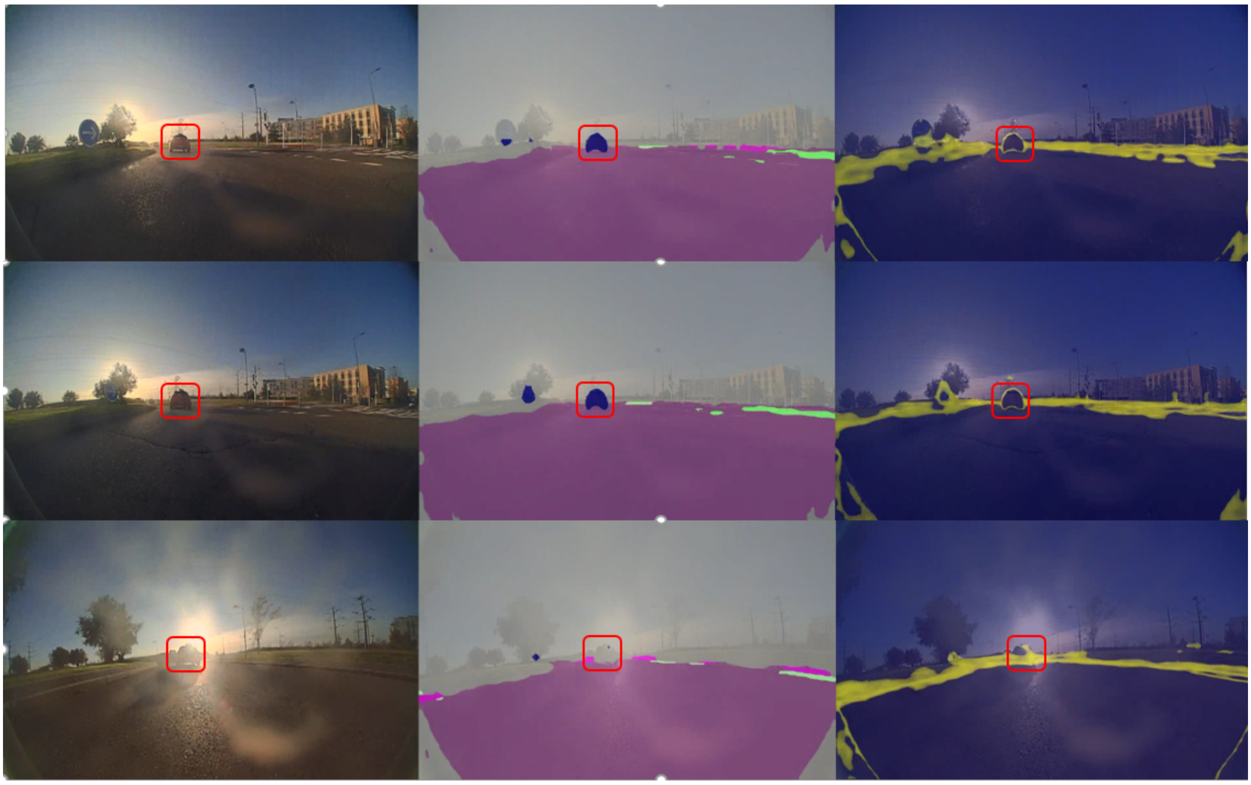


Figure 6.14: **Online test**: In this sequence, we show that our observer is able to detect a wrong prediction due to sun glare, in a live demo.

Visually, the results look much more smooth and accurate than the previous record.

Finally, we designed new scenarios to test our ObsNet on real anomalous samples. We placed different objects in front of the car, like fences, construction cones or cardboard. As shown on Figure 6.15 we succeed in detecting these objects with high anomaly scores.

6.5 LIMITATION AND FUTURE IMPROVEMENT

In this chapter, we show that our previous research is exploitable for industry. We make a proof of concept that we can detect anomalies on the road thanks to our ObsNet pre-trained on a custom dataset. By using a Demo Car, a Nvidia Jetson and RTmaps, our solution works at 8 FPS which allows the car to ride at slow speed like in dense urban scenes or for parking areas.

However, our solution shows limitations when the car speed increases. The faster the car is, the more distance between every frame is traveled. Thus our integrated solution is not fitted yet for a high speed like highway. But we argue in Chapter 4 that our method is still much more time efficient than others' works. Indeed, we could consider model compression techniques like pruning [FC19][1], knowledge distillation [HVD15], quantization [FLF+18] or Neural Architecture Search (NAS) [HLL+18] to increase both model efficiency and inference time, but we let this engineer magic for future works.

[1] Jonathan Frankle et al., *The lottery ticket hypothesis: Finding sparse, trainable neural networks*. In ICML 2019

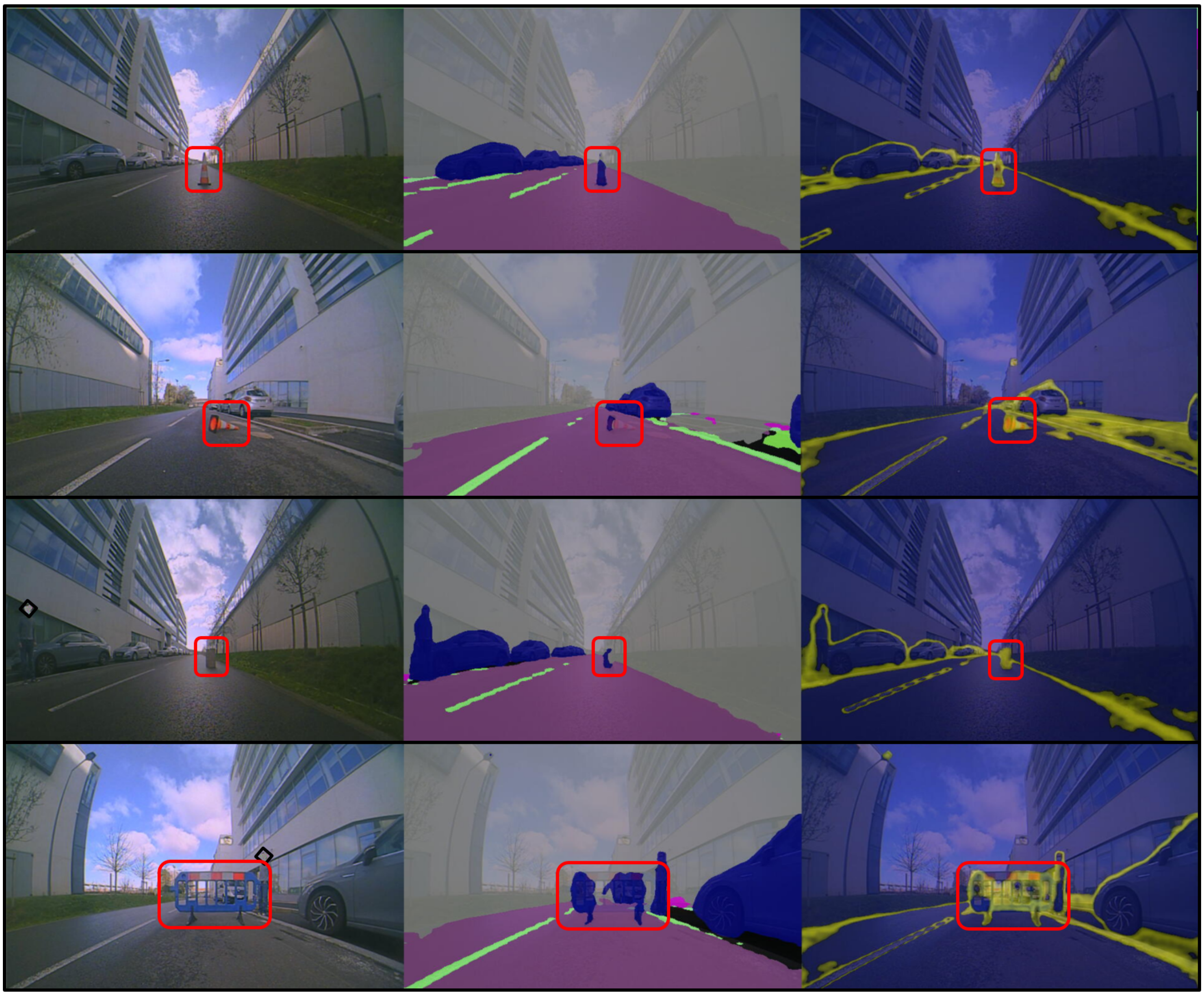


Figure 6.15: **Online test**: We constructed a custom scenario to show that our ObsNet is able to detect road obstacles in front of the car.

Part IV

EPILOGUE

7

Conclusion

◂ Chapter 6 Appendix A ▸

7.1 SUMMARY

Along this manuscript, we tackled safety of automated driving by using Observer Networks in order to improve Deep Neural Networks (DNNs) trustworthiness. We introduced the concept of observer network: a framework that seeks inside of a target network's intermediate activations. We showed how to train observer networks when a few meaningful data are available to train the observer. In Chapter 3, we leverages an oracle prediction to learn a divergence based uncertainty measurement. We succeeded in learning both epistemic and aleatoric uncertainty, and predict uncertainty map as accurate as other competitive methods while being much faster. However, this framework is bound by the oracle performance. In Chapter 4, we dropped the oracle supervision and uncertainty measurement. We leveraged Localized Adversarial Attacks (LAA) of OoD detection. LAA increases the number of errors available at train time to better detect Out-of-Distribution samples. ObsNet + LAA outperforms every other methods, and gets State-Of-The-Art results in a public challenging benchmark[2]. Nevertheless, observer networks produce pixel-wise error maps, which do not enable automatic detection of object instances in the scene. This concern is tackled in Chapter 5, by improving the concept with instance segmentation. We used class agnostic instance masks to filter and aggregate the errors map of ObsNet. We showed with this method that we can better predict the position and size of the object. The two first contributions have been published in two major conferences (ICIP and ICCV) and the last one is still under review.

To validate the soundness of our research works, we also showed in Chapter 6 how to integrate an observer network as well as a segmentation network on a Demo Car as a proof of concept. We built a complete workflow inside of a Volkswagen car equipped with four fisheye cameras, running with an Nvidia Jetson Xavier and orchestrated by RTMaps. Our solution succeeded in detecting abnormal situations on the road at low speed. We believe that our contributions will help to make the decision process of neural networks safer, faster than previous methods, and use few additional storage compared to literature.

[2] Segment Me If You Can dataset leaderboard is updated regularly and available at https://segmentmeifyou-can.com/leaderboard

7.2 FUTURE WORKS

Future directions can be split into two different paths. The first option is to rely even more on the instance information to detect Out-of-Distribution samples. The second one is to use boosting of observers in order to build a stronger model anomaly detection. Below, we explore in more detail our ideas.

Panoptic Errors detection In this first idea, we want to adapt ObsNet for panoptic segmentation. In fact, we discover that current approaches to detect OoD mostly consider every pixel as either in distribution or Out-of-Distribution independently. We believe that pixel level prediction is suboptimal for the goal of OoD sample detection. These methods produce noisy error maps, inaccurate and not adapted for many tasks. Even worse, these methods can be uncertain in the boundary of an object while being overconfident inside of it. We believe that panoptic anomaly detection can mitigate this problem. Indeed, allowing the observer to have instance aware privilege can help to better localize the object of interest. Moreover, we believe that it could predict more clear and homogeneous predictions.

We propose to use a Max-DeepLab [WZA+21] as the target network. This network segments the image into a fixed-size set of N class-labeled masks $\{m_i, p_i(c)\}_{i=0}^N$ where m_i is the segmentation mask and $p_i(c)$ is the probability to assign the class c on the mask m. To improve the reliability of this target network, we could use an observer networks, to predict an additional vector $u_i \in \mathbb{R}^N$. Each value of this vector is associated to its corresponding mask and correspond to the probability that the target network predict the wrong class, *i.e.*, $u_i \approx Pr[p_i(c) \neq \hat{y}_i]$ with $\hat{y}_i$ the ground truth for mask m_i. Indeed, the observer could take intermediate attention blocks of the target network. To train the observer with adversarial attacks, we could attack a specific mask m_i to hallucinate a new shape or a different class.

Boosting Neural networks The second path is more exploratory. In classical machine learning, boosting is a general framework that combines multiple weak classifiers to form a stronger one. Each weak classifier should slightly correct the previous one. This class of methods performs particularly well for tabular data [FS97, CG16][1] But intuitively, these methods should not work properly for Deep Learning. In fact, modern Deep Neural Networks are prone to overfitting, and thus cannot be considered as weak classifiers because they perform nearly perfectly on the train set. [WW16] is one of the few attempts to use boosting and Deep Neural Networks. They iteratively add shallow neural networks, and step by step correct the errors made by the previous ones.

On our side, we can also view our observer network as a boosting of two networks: the target network makes a prediction, and the observer network tries to detect either it is right or wrong. But we

[1] [CG16] Tianqi Chen et al., *Xgboost: A scalable tree boosting system*. In KDD 2016.

could scale the framework by having a stack of observer networks. To counter the problem of overfitting, we could use Local Adversarial Attacks. If we perform enough LAA, we can, in theory, have as many errors in the training set as we want. Thus, this technique could be used to train a set of observers that try to correct previous observers under more and more LAA. Indeed, increasing the number of ObsNet will automatically increase the number of parameters and the number of operations needed. But if we consider a smaller architecture for each observer and dedicate each ObsNet to one unique activation of the target network as input, we can mitigate the overall computation.

Artwork 2: A safe place to drive in the future. DreamStudio, image generated from Stable Diffusion [RBL+21] model.

References

[21419] ISO/PAS 21448. Road vehicles — safety of the intended functionality. *ISO/PAS*, 2019.

[AB18] Murat Seçkin Ayhan and Philipp Berens. Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks. *Medical Imaging with Deep Learning*, 2018.

[AC20] Faruk Ahmed and Aaron Courville. Detecting semantic anomalies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[AOS$^+$16] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[BBL$^+$19] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. Anomaly detection using autoencoders in high performance computing systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[BBPB21] Victor Besnier, Andrei Bursuc, David Picard, and Alexandre Briot. Triggering failures: Out-of-distribution detection by learning from local adversarial attacks in semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15701–15710, October 2021.

[BBPB22] Victor Besnier, Andrei Bursuc, David Picard, and Alexandre Briot. Instance-aware observer network for out-of-distribution object segmentation. In *ArXiv*, 2022.

[BCD$^+$20] Mariusz Bojarski, Chenyi Chen, Joyjit Daw, Alperen Degirmenci, Joya Deri, Bernhard Firner, Beat Flepp, Sachin Gogri, Jesse Hong, Lawrence D. Jackel, Zhenhua Jia, B. J. Lee, Bo Liu, Fei Liu, Urs Muller, Samuel Payne, Nischal Kota Nagendra Prasad, Artem Provodin, John Roach, Timur Rvachov, Neha Tadimeti, Jesper E. van Engelen, Haiguang Wen, Eric Yang, and Zongyi Yang. The NVIDIA pilotnet experiments. *ArXiv*, 2020.

[BCKW15] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *ICML*, 2015.

[BDPW22] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEit: BERT pre-training of image transformers. In *International Conference on Learning Representations*, 2022.

[BFC08] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2008.

[BFSS19] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9592–9600, 2019.

[BJB$^+$20] Victor Besnier, Himalaya Jain, Andrei Bursuc, Matthieu Cord, and Patrick Pérez. This dataset does not exist: Training models from generated images. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[BKC17] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. PAMI*, 2017.

[BKNS00] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[BKOŠ19] Petra Bevandić, Ivan Krešo, Marin Oršić, and Siniša Šegvić. Simultaneous semantic segmentation and outlier detection in presence of domain shift. In *GCPR*, 2019.

[BMN21] Robert Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep learning through the lens of example difficulty. *Advances in Neural Information Processing Systems*, 34:10876–10889, 2021.

[BPB21] Victor Besnier, David Picard, and Alexandre Briot. Learning uncertainty for safety-oriented semantic segmentation in autonomous driving. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3353–3357, September 2021.

[BSN+19] Hermann Blum, Paul-Edouard Sarlin, Juan Nieto, Roland Siegwart, and Cesar Cadena. The fishyscapes benchmark: Measuring blind spots in semantic segmentation. *arXiv*, 2019.

[BTD+16] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *ArXiv*, 2016.

[BWAN18] Christoph Baur, Benedikt Wiestler, Shadi Albarqouni, and Nassir Navab. Deep autoencoding models for unsupervised anomaly segmentation in brain MR images. In *MICCAI Workshops*, 2018.

[CCKL19] Jiwoong Choi, Dayoung Chun, Hyun Kim, and Hyuk-Jae Lee. Gaussian YOLOv3: An accurate and fast object detector using localization uncertainty for autonomous driving. In *ICCV*, 2019.

[CCZ+20] Bowen Cheng, Maxwell D. Collins, Yukun Zhu, Ting Liu, Thomas S. Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[CLLO18] Sungjoon Choi, Kyungjae Lee, Sungbin Lim, and Songhwai Oh. Uncertainty-aware learning from demonstration using mixture density networks with sampling-free variance modeling. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[CLU+21] Robin Chan, Krzysztof Lis, Svenja Uhlemeyer, Hermann Blum, Sina Honari, Roland Y. Siegwart, Mathieu Salzmann, P. Fua, and Matthias Rottmann. Segmentmeifyoucan: A benchmark for anomaly segmentation. In *NeurIPS Datasets and Benchmarks*, 2021.

[CM15] Clement Creusot and Asim Munawar. Real-time small obstacle detection on highways using compressive rbm road reconstruction. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 162–167. IEEE, 2015.

[CMS+20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, 2020.

[CMS+21] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. *arXiv*, 2021.

[COR+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[CPK+16] 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. *arXiv*, 2016.

[CPSA17] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv*, 2017.

[CRG21a] Robin Chan, Matthias Rottmann, and Hanno Gottschalk. Entropy maximization and meta classification for out-of-distribution detection in semantic segmentation. In *Proceedings of the ieee/cvf international conference on computer vision*, pages 5128–5137, 2021.

[CRG21b] Robin Chan, Matthias Rottmann, and Hanno Gottschalk. Entropy maximization and meta classification for out-of-distribution detection in semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5128–5137, October 2021.

[CSK21] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *NeurIPS*, 2021.

[CTBH+19] Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. In *NeurIPS*, 2019.

[CWL+20] Changhao Chen, Bing Wang, Chris Xiaoxuan Lu, Niki Trigoni, and Andrew Markham. A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence, 2020.

[CZG20] Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts. *Advances in Neural Information Processing Systems*, 33:1356–1367, 2020.

[CZP+18] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[DBBSC21] Giancarlo Di Biase, Hermann Blum, Roland Siegwart, and Cesar Cadena. Pixel-wise anomaly detection in complex driving scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16918–16927, June 2021.

[DBK+21] 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. In *ICLR*, 2021.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[DJW+20] Michael W. Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable bayesian neural nets with rank-1 factors. In *ICML*, 2020.

[DKD09] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.

[DT18] Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv*, 2018.

[DWCL22] Xuefeng Du, Zhaoning Wang, Mu Cai, and Sharon Li. Towards unknown-aware learning with virtual outlier synthesis. In *ICLR*, 2022.

[DWGL22] Xuefeng Du, Xin Wang, Gabriel Gozum, and Yixuan Li. Unknown-aware object detection: Learning what you don't know from videos in the wild. In *CVPR*, 2022.

[EGW+10] Mark Everingham, Luc Van Gool, C. K. I. Williams, J. Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge, 2010.

[FBA+20] Gianni Franchi, Andrei Bursuc, Emanuel Aldea, Séverine Dubuisson, and Isabelle Bloch. TRADI: Tracking deep neural network weight distributions. In *ECCV*, 2020.

[FC19] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[FHL19] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

[FLF+18] Hongxiang Fan, Shuanglong Liu, Martin Ferianc, Ho-Cheung Ng, Zhiqiang Que, Shen Liu, Xinyu Niu, and Wayne Luk. A real-time object detection accelerator with compressed ssdlite on fpga. In *2018 International Conference on Field-Programmable Technology (FPT)*, 2018.

[FS97] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 1997.

[FSW+19] Hao-Shu Fang, Jianhua Sun, Runzhong Wang, Minghao Gou, Yong-Lu Li, and Cewu Lu. Instaboost: Boosting instance segmentation via probability map guided copy-pasting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 682–691, 2019.

[Gal16] Yarin Gal. Uncertainty in Deep Learning. *PhD*, 2016.

[GBKS21] Matej Grcic, Petra Bevandic, Zoran Kalafatic, and Sinisa Segvic. Dense anomaly detection by robust learning on synthetic negative data. *arXiv*, 2021.

[GBP18] Corina Gurau, Alex Bewley, and Ingmar Posner. Dropout distillation for efficiently estimating model confidence. *arXiv*, 2018.

[GBv22] Matej Grcić, Petra Bevandić, and Siniša Šegvić. Densehybrid: Hybrid anomaly detection for dense open-set recognition. In *Computer Vision - ECCV 2022 - 17th European Conference*, 2022.

[GG16] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

[GHM$^+$22] Stefano Gasperini, Jan Haug, Mohammad-Ali Nikouei Mahani, Alvaro Marcos-Ramiro, Nassir Navab, Benjamin Busam, and Federico Tombari. Certainnet: Sampling-free uncertainty estimation for object detection. *RAL*, 2022.

[Gir15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2015.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[GLW$^+$21] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[GMFB19] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth prediction. In *The International Conference on Computer Vision (ICCV)*, 2019.

[GPSW17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. *ICML*, 2017.

[Gra11] Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

[GRM$^+$19] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015.

[HBM$^+$19] Dan Hendrycks, Steven Basart, Mantas Mazeika, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. A benchmark for anomaly segmentation. *ArXiv*, 2019.

[HCS$^+$19] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730. PMLR, 2019.

[HDVG18] Simon Hecker, Dengxin Dai, and Luc Van Gool. Failure prediction for autonomous driving. In *IV*, 2018.

[HG17] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

[HLA15] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*, pages 1861–1869. PMLR, 2015.

[HLL$^+$18] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[HLM19] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *ICML*, 2019.

[HMD18] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *ICLR*, 2018.

[HMK$^+$20] Seong Jae Hwang, Ronak R Mehta, Hyunwoo J Kim, Sterling C Johnson, and Vikas Singh. Sampling-free uncertainty estimation in gated recurrent units with applications to normative modeling in neuroimaging. In *Uncertainty in Artificial Intelligence*, pages 809–819. PMLR, 2020.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[ISO19] ISO. Iso26262, road vehicles - functional safety. *ISO*, 2019.

[JSR$^+$20] Debesh Jha, Pia H. Smedsrud, Michael A. Riegler, Pål Halvorsen, Thomas de Lange, Dag Johansen, and Håvard D. Johansen. Kvasir-seg: A segmented polyp dataset. In Yong Man Ro, Wen-Huang Cheng, Junmo Kim, Wei-Ta Chu, Peng Cui, Jung-Woo Choi, Min-Chun Hu, and Wesley De Neve, editors, *MultiMedia Modeling*, 2020.

[KBC15] Alex Kendall, Vijay Badrinarayanan, , and Roberto Cipolla. Bayesian SegNet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv*, 2015.

[KD09] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic ? does it matter ? *Structural Safety*, 2009.

[KD18] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[KG17] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NeurIPS*, 2017.

[KGHD19] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollar. Panoptic feature pyramid networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[KHG$^+$19] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollar. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Neurips*, 2012.

[KSW15] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015.

[LAE$^+$16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[LCPK$^+$15] Chen Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. In *ICLR*, May 2015.

[LGG$^+$17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[LLC$^+$21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[LLLS18] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*, 2018.

[LMB$^+$14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[LNFS19] Krzysztof Lis, Krishna Nakka, Pascal Fua, and Mathieu Salzmann. Detecting the unexpected via image resynthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[LPB17] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, June 2015.

[LSL18] Shiyu Liang, R. Srikant, and Yixuan Li. Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR*, 2018.

[LSYP21] Chun-Liang Li, Kihyuk Sohn, Jinsung Yoon, and Tomas Pfister. Cutpaste: Self-supervised learning for anomaly detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9664–9674, 2021.

[LTZ08] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *KDD*, 2008.

[LW16] Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix Gaussian posteriors. *33rd International Conference on Machine Learning, ICML 2016*, 2016.

[LW17] Christos Louizos and Max Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. *34rd International Conference on Machine Learning, ICML 2017*, 2017.

[MAG$^+$20] Alireza Mehrtash, Purang Abolmaesumi, Polina Golland, Tina Kapur, Demian Wassermann, and William M Wells III. Pep: Parameter ensembling by perturbation. In *NeurIPS*, 2020.

[MATTEP08] Heydar Maboudi Afkham, Alireza Tavakoli Targhi, Jan-Olof Eklundh, and Andrzej Pronobis. Joint visual vocabulary for animal classification. In *ICPR*, 2008.

[MFW16] Caio César Teodoro Mendes, Vincent Frémont, and Denis Fernando Wolf. Exploiting fully convolutional neural networks for fast road detection. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[MG18a] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *NeurIPS*, 2018.

[MG18b] Jishnu Mukhoti and Yarin Gal. Evaluating bayesian deep learning methods for semantic segmentation. *arXiv preprint arXiv:1811.12709*, 2018.

[MG19] Andrey Malinin and Mark Gales. Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness. In *Advances in Neural Information Processing Systems*, 2019.

[MGK$^+$17] Rowan McAllister, Yarin Gal, Alex Kendall, Mark Van Der Wilk, Amar Shah, Roberto Cipolla, and Adrian Weller. Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. In *IJCAI*, 2017.

[MIG$^+$19] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *NeurIPS*, 2019.

[MKG18] Rhiannon Michelmore, Marta Z. Kwiatkowska, and Yarin Gal. Evaluating uncertainty quantification in end-to-end autonomous driving control. *ArXiv*, abs/1811.06817, 2018.

[MMG20] Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. Ensemble distribution distillation. In *International Conference on Learning Representations*, 2020.

[MMKI18] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Trans. PAMI*, 2018.

[MMS$^+$18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[MSK18] Marcin Możejko, Mateusz Susik, and Rafał Karczewski. Inhibited softmax for uncertainty estimation in neural networks. *arXiv*, 2018.

[NC16] Mahdi Pakdaman Naeini and Gregory F Cooper. Binary classifier calibration using an ensemble of near isotonic regression models. In *KDD*, 2016.

[NCH15] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, 2015.

[NDZ$^+$19] Jeremy Nixon, Michael W. Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *CVPRW*, 2019.

[Nea12] Radford M Neal. Bayesian learning for neural networks. *arXiv*, 2012.

[Neu18] Lukas Neumann. Relaxed Softmax : Efficient Confidence Auto-Calibration for Safe Pedestrian Detection. *NIPS Machine Learning for Intelligent Transportation Systems Workshop*, 2018.

[NHH15] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[NMT$^+$19] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2019.

[OFR$^+$19] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

[ORF20] Philipp Oberdiek, Matthias Rottmann, and Gernot A. Fink. Detection and retrieval of out-of-distribution objects in semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[P+99] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *ALMC*, 1999.

[PFC+19] Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab, and Federico Tombari. Sampling-free epistemic uncertainty estimation using approximated variance propagation. In *ICCV*, 2019.

[Pom88] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

[PRG+16] Peter Pinggera, Sebastian Ramos, Stefan Gehrig, Uwe Franke, Carsten Rother, and Rudolf Mester. Lost and found: detecting small road hazards for self-driving vehicles. In *iros*, 2016.

[QCY20] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. *arXiv preprint arXiv:2006.02334*, 2020.

[RBL+21] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[RF17] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[RF18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[RHGS17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jun 2017.

[RLF+19] Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. *Advances in neural information processing systems*, 32, 2019.

[RPZ+22] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14318–14328, 2022.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[RSM$^+$16] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[RVRK16] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016.

[SAF19] SAFAD. Safety first for automated driving. *SAFAD*, 2019.

[SAN$^+$04] J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever, and B. van Ginneken. Ridge-based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 2004.

[SGLS21] 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*, pages 7262–7272, 2021.

[SHF16] Saurabh Singh, Derek Hoiem, and David Forsyth. Swapout: Learning an ensemble of deep architectures. *Advances in neural information processing systems*, 29, 2016.

[SHK$^+$14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014.

[SKCS20] Murat Sensoy, Lance Kaplan, Federico Cerutti, and Maryam Saleki. Uncertainty-aware deep classifiers using generative models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[SLJ$^+$15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[SSL19] Alireza Shafaei, Mark Schmidt, and James J Little. A less biased evaluation of out-of-distribution sample detectors. In *BMVC*, 2019.

[SSSS17] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

[SSW$^+$17] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *IPMI*, 2017.

[SvNB$^+$19a] Laurens Samson, Nanne van Noord, Olaf Booij, Michael Hofmann, Efstratios Gavves, and Mohsen Ghafoorian. I bet you are wrong: Gambling adversarial networks for structured semantic segmentation. In *ICCV Workshops*, 2019.

[SvNB$^+$19b] Laurens Samson, Nanne van Noord, Olaf Booij, Michael Hofmann, Efstratios Gavves, and Mohsen Ghafoorian. I bet you are wrong: Gambling adversarial networks for structured semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[SWS$^+$00] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *NeurIPS*, 2000.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[TAS18] Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. *arXiv*, 2018.

[VPSM20] Shashanka Venkataramanan, Kuan-Chuan Peng, Rajat Vikram Singh, and Abhijit Mahalanobis. Attention guided anomaly localization in images. In *ECCV*, 2020.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[VvA+21] Tomas Vojir, Tomáš Šipka, Rahaf Aljundi, Nikolay Chumerin, Daniel Olmeda Reino, and Jiri Matas. Road anomaly detection by partial image reconstruction with segmentation coupling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15651–15660, October 2021.

[WKS+20] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. SOLO: Segmenting objects by locations. In *Proc. Eur. Conf. Computer Vision (ECCV)*, 2020.

[WNM+19] Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E. Turner, Jose Miguel Hernandez-Lobato, and Alexander L. Gaunt. Deterministic variational inference for robust bayesian neural networks. In *International Conference on Learning Representations*, 2019.

[WT11] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

[WW16] Elad Walach and Lior Wolf. Learning to count with cnn boosting. In *European conference on computer vision*, pages 660–676. Springer, 2016.

[WZA+21] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. MaX-DeepLab: End-to-end panoptic segmentation with mask transformers. In *CVPR*, 2021.

[WZG+20] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, 2020.

[WZK+20] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[XGD+17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[XWY+21] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *arXiv preprint arXiv:2105.15203*, 2021.

[XZL+20] Yingda Xia, Yi Zhang, Fengze Liu, Wei Shen, and Alan Yuille. Synthesize then compare: Detecting failures and anomalies for semantic segmentation. In *ECCV*, 2020.

[YCW$^+$20] 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. In *CVPR*, 2020.

[YHH$^+$19] Senthil Yogamani, Ciarán Hughes, Jonathan Horgan, Ganesh Sistu, Padraig Varley, Derek O'Dea, Michal Uricár, Stefan Milz, Martin Simon, Karl Amende, et al. Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9308–9318, 2019.

[YK16] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[YK19] Donggeun Yoo and In So Kweon. Learning loss for active learning. In *CVPR*, 2019.

[ZE01] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *ICML*, 2001.

[ZE02] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *KDD*, 2002.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.

[ZSDG17] Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference. *arXiv preprint arXiv:1712.02390*, 2017.

[ZSL$^+$21] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021.

[ZSQ$^+$17] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[ZZP$^+$17] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Part V

GENERAL APPENDIX

A

Additional Results and Training details

◂ Chapter 7

A.1 LEARNING DIVERGENCE BASED UNCERTAINTY

In this we complete Chapter 3 with additional results and visualization to better understand and validate the soundness of our methods.

A.1.1 *Additional qualitative results*

In Figure A.1 we show results on every noise. We test: rain (*i.e.*, gray line on the image) and patches (*i.e.*, rectangles of uniform color and random sizes). We can see that, just as glare, ObsNet is able to detect both aleatoric and epistemic uncertainty. Distillation fails to detect such uncertainty as it is trained to predict the same output as the one of the teacher.

Such behavior can be seen on Figure A.2. Here, we fix a precision threshold of 95%, we rank each pixel of the image by uncertainty, then we try to make the best prediction without falling under the threshold. We can see that ObsNet is able to make good predictions in nearly all images, where MCDropout and distillation can not.

A.1.2 *Aleatoric additional results*

In Table A.1, Table A.2 and Table A.3, we show the complete results for aleatoric uncertainty. We can see that even if ObsNet is trained on another noise, it is able to achieve good results. Moreover, training ObsNet with several noises significantly improves the performances for all noises. We can then use much more noise to be more robust to any kind of situation for autonomous driving (e.g., snow, stain, or every low light).

A.1.3 *Uncertainty Label c*

We show in Figure A.3 the self-supervised label c from epistemic uncertainty. The red pixels on the most right images are the pixels where the uncertainty is above the safety parameters δ. It shows where we learn to maximize the KL divergence (red pixel) and where we learn to minimize it (in blue). For aleatoric uncertainty estimation, we see in red the pixels where the noise shifts the output distribution above the safety parameter δ. Thus, leading to aleatoric uncertainty

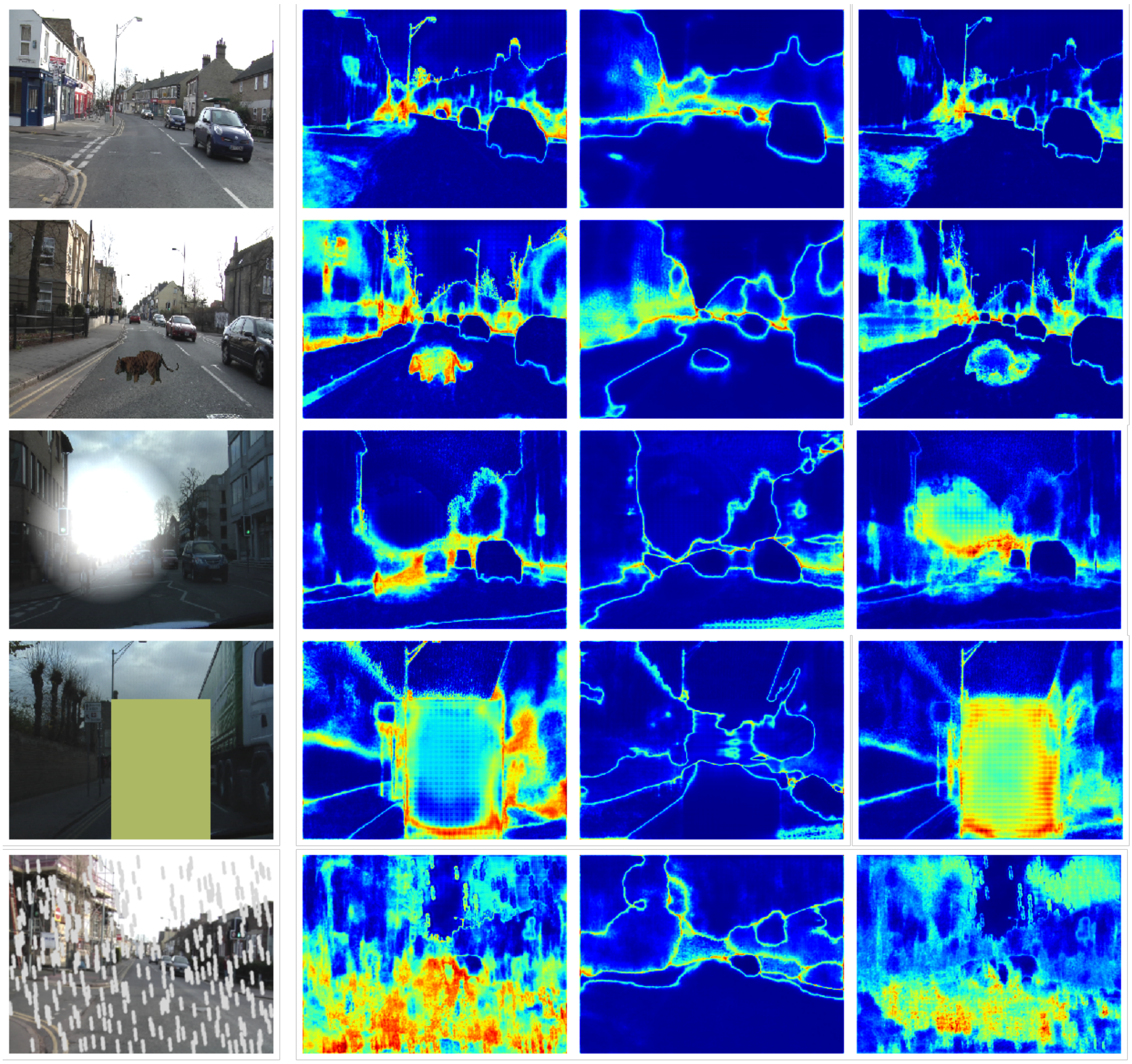


Figure A.1: **Uncertainty map**. From column left to right: Input image, MC-Dropout uncertainty, distillation uncertainty, and ObsNet uncertainty. We add two noises which are rain and patch.

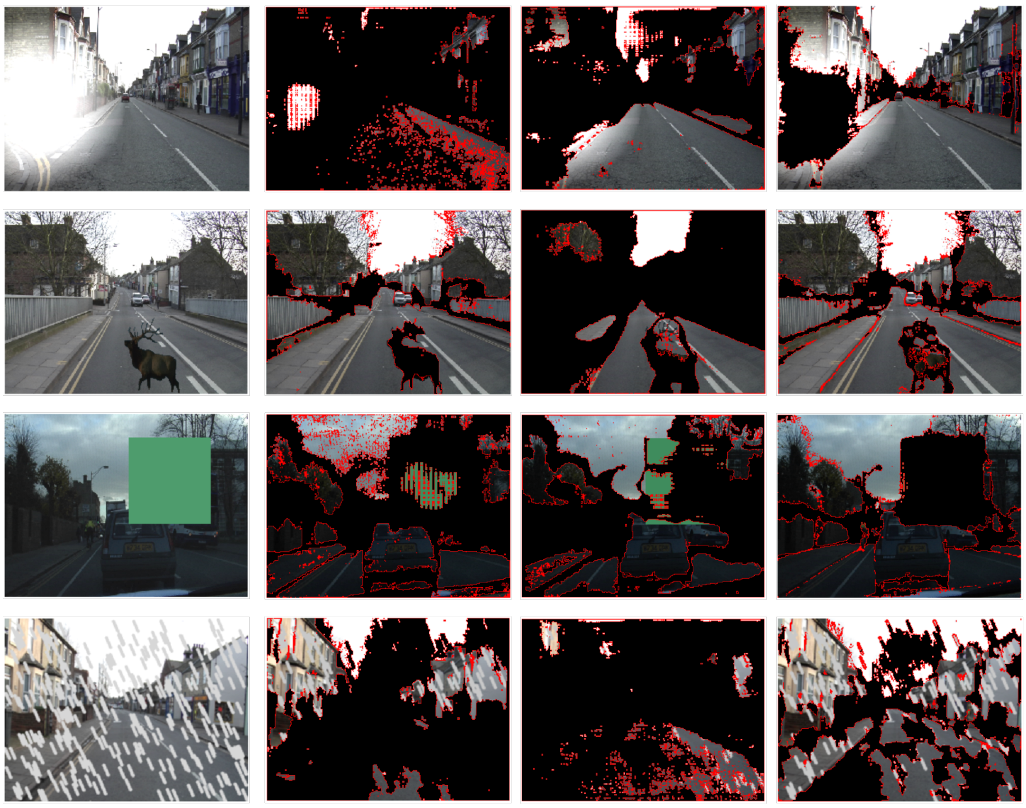


Figure A.2: **Coverage of the safe predictions.** In black and red, where the uncertainty leads the overall precision to fall under 95%. From left to right: original image, MCDropout, distillation, and ObsNet.

Test Glare	Method	Train	R@P=0.95	AuPR	Trigger 75%
no-retrain	MCDropout T=50	-	0.1 ± 0.1	83.9 ± 0.1	18.9 ± 1.5
	MCDropout T=2	-	0.0 ± 0.0	83.7 ± 0.2	15.5 ± 1.6
	Softmax	-	1.0 ± 0.3	90.5 ± 0.8	17.2 ± 2.7
	Void Class	-	0.3 ± 0.1	82.7 ± 0.1	8.2 ± 0.9
	MCDA T=50	-	44.7 ± 0.5	91.7 ± 0.2	14.3 ± 2.0
Distillation	w/ T supervised	patch	0.3 ± 0.1	85.7 ± 0.5	4.1 ± 0.1
	w/ T unsupervised	patch	1.9 ± 0.5	85.5 ± 0.7	4.3 ± 0.9
	w/ T supervised	rain	0.5 ± 0.0	77.1 ± 0.9	3.8 ± 0.8
	w/ T unsupervised	rain	0.5 ± 0.1	85.0 ± 0.5	4.2 ± 0.8
	w/ T supervised	glare	0.0 ± 0.3	83.3 ± 0.2	2.2 ± 0.8
	w/ T unsupervised	glare	1.6 ± 0.1	84.1 ± 0.3	3.0 ± 0.9
	w/ T supervised	all	0.0 ± 0.0	81.9 ± 0.3	1.7 ± 0.8
	w/ T unsupervised	all	1.1 ± 0.3	85.9 ± 0.2	3.1 ± 1.7
ObsNet (OURS)	from MCDropout	patch	46.7 ± 1.3	92.3 ± 0.3	15.1 ± 2.1
	from MCDropout	rain	33.9 ± 1.1	90.8 ± 0.3	18.3 ± 1.0
	from MCDropout	glare	68.4 ± 0.9	95.3 ± 0.1	23.2 ± 1.5
	from MCDropout	all	**76.1 ± 0.5**	**96.1 ± 0.1**	**33.5 ± 1.8**

Table A.1: **Sun Glare Evaluation.** Evaluation of aleatoric uncertainty, tested on glare.

Test Rain	Method	Train	R@P=0.95	AuPR	Trigger 33%
no-retrain	MCDropout T=50	-	10.3 ± 0.8	74.9 ± 0.5	5.6 ± 0.6
	MCDropout T=2	-	7.3 ± 0.7	73.3 ± 0.5	1.5 ± 0.3
	Softmax	-	6.1 ± 2.0	72.6 ± 0.3	1.2 ± 0.3
	Void Class	-	7.2 ± 3.5	73.0 ± 0.1	1.3 ± 0.3
	MCDA T=50	-	6.2 ± 0.4	69.6 ± 0.3	0.2 ± 0.2
Distillation	w/ T supervised	glare	0.0 ± 0.0	55.0 ± 0.6	0.0 ± 0.0
	w/ T unsupervised	glare	0.1 ± 0.1	54.1 ± 0.7	0.0 ± 0.0
	w/ T supervised	patch	0.0 ± 0.0	54.1 ± 0.4	0.0 ± 0.0
	w/ T unsupervised	patch	0.0 ± 0.0	53.4 ± 0.5	0.0 ± 0.0
	w/ T supervised	rain	0.0 ± 0.0	47.6 ± 0.8	0.0 ± 0.0
	w/ T unsupervised	rain	0.0 ± 0.0	51.1 ± 0.2	0.0 ± 0.0
	w/ T supervised	all	0.0 ± 0.0	53.7 ± 0.4	0.0 ± 0.0
	w/ T unsupervised	all	0.0 ± 0.0	61.1 ± 0.2	0.0 ± 0.0
ObsNet (OURS)	from MCDropout	glare	3.1 ± 0.5	69.3 ± 0.6	0.3 ± 0.4
	from MCDropout	patch	3.1 ± 0.6	70.4 ± 0.6	0.8 ± 0.4
	from MCDropout	rain	32.5 ± 0.8	86.9 ± 0.2	24.7 ± 1.2
	from MCDropout	all	**33.2 ± 1.3**	**87.2 ± 0.2**	**25.1 ± 2.0**

Table A.2: **Rain Evaluation**. Evaluation of aleatoric uncertainty, tested on rain.

Test Patch	Method	Train	R@P=0.95	AuPR	Trigger 75%
no-retrain	MCDropout T=50	-	63.9 ± 7.3	94.9 ± 0.7	19.6 ± 2.9
	MCDropout T=2	-	23.4 ± 5.1	93.5 ± 0.5	18.5 ± 2.5
	Softmax	-	0.3 ± 0.1	90.3 ± 0.3	15.9 ± 1.2
	Void Class	-	0.2 ± 0.1	86.4 ± 0.3	8.7 ± 1.1
	MCDA T=50	-	32.0 ± 1.3	90.4 ± 0.2	12.4 ± 1.1
Distillation	w/ T supervised	glare	0.0 ± 0.0	85.2 ± 0.3	3.1 ± 0.4
	w/ T unsupervised	glare	1.3 ± 0.2	84.8 ± 0.6	3.2 ± 0.9
	w/ T supervised	rain	0.0 ± 0.0	54.0 ± 0.1	0.0 ± 0.0
	w/ T unsupervised	rain	0.0 ± 0.0	54.1 ± 0.4	0.0 ± 0.0
	w/ T supervised	patch	1.4 ± 0.4	79.6 ± 0.7	4.4 ± 0.9
	w/ T unsupervised	patch	2.2 ± 0.3	82.7 ± 0.5	4.5 ± 1.5
	w/ T supervised	all	0.0 ± 0.0	79.5 ± 0.7	1.9 ± 0.9
	w/ T unsupervised	all	1.3 ± 0.3	84.7 ± 0.6	3.4 ± 1.3
ObsNet (OURS)	from MCDropout	glare	46.5 ± 0.9	92.6 ± 0.2	15.2 ± 2.1
	from MCDropout	rain	2.7 ± 0.5	70.2 ± 0.2	0.9 ± 0.6
	from MCDropout	patch	72.5 ± 0.4	95.6 ± 0.1	23.6 ± 2.7
	from MCDropout	all	**77.5 ± 0.4**	**96.1 ± 0.1**	**30.5 ± 1.4**

Table A.3: **Square Patch Evaluation**. Evaluation of aleatoric uncertainty, tested on patch.

(red pixels) or not (blue pixels). We can see that even if some pixels overlap by the glare, the corresponding label is $c = +1$. This means that if the target network can predict the right class for this pixel, then there is enough information in the data to make a safe prediction.

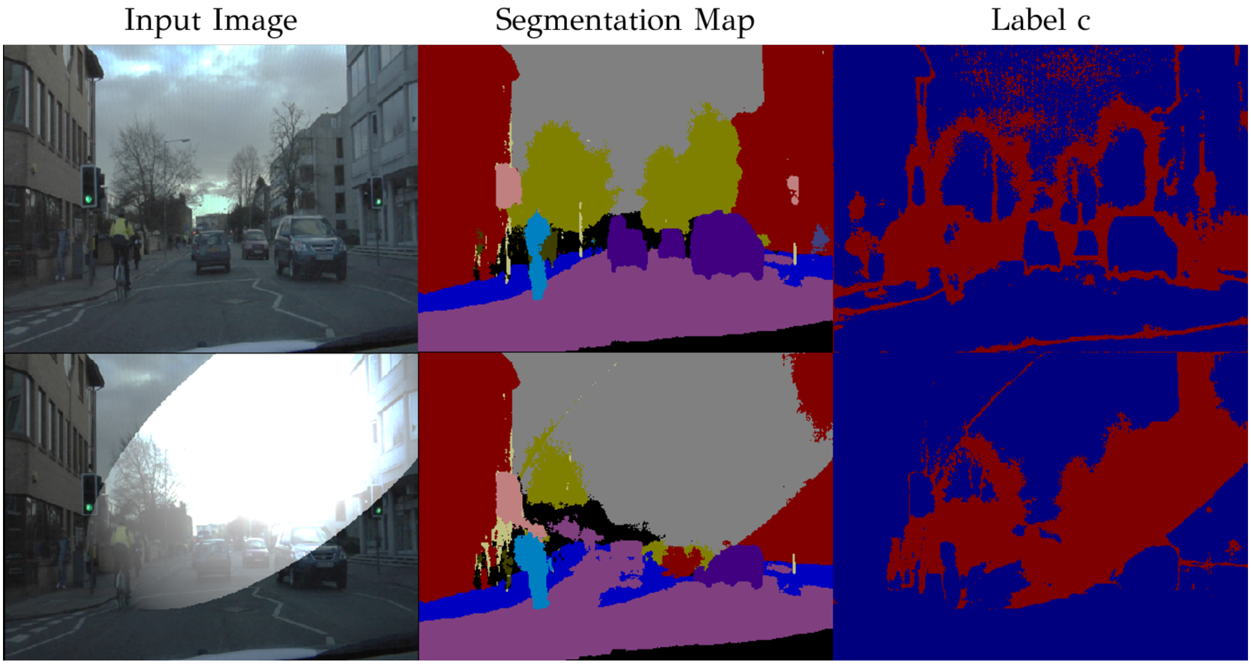


Figure A.3: **Label c visualisation** First row: Epistemic uncertainty, Last row: Aleatoric uncertainty. The last column shows self-supervised labels, $c = +1$ in blue and $c = -1$ in red.

A.2 LEARNING FROM LOCAL ADVERSARIAL ATTACKS

A.2.1 *Implementation details & hyper-parameters*

For our implementation, we use Pytorch[1] and will release the code after the review. We share each hyper-parameter in Table A.4. We train ObsNet with SGD with momentum and weight decay for at most 50 epochs using early-stopping. ObsNet is not trained from scratch as we initialize the weights with those of the segmentation network. We also use a scheduler to divide the learning rate by 2 at epoch 25 and epoch 45. We use the same data augmentation (*i.e.*, Horizontal Flip and Random Crop) for training of the segmentation network and as well as for ObsNet. As there are few errors in the training of ObsNet, we increase the weight of positive examples in the loss contribution (Pos Weight in Table A.4).

In Figure A.4, we propose a detailed view of our observer network training scheme. The architecture corresponds to a SegNet [BKC17]. As we can see, we put residual connections after each down-sample or up-sample block. Here, we use a sigmoid output to directly predict the pixel-wise error map.

[1] A Paszke et al., *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, NIPS 2019

Params	CamVid	StreetHazards	Bdd Anomaly
Epoch	50	50	50
Optimizer	SGD	SGD	SGD
LR	0.05	0.02	0.02
Batch Size	8	6	6
Loss	BCE	BCE	BCE
Pos Weight	2	3	3
LAA shape	rand shape	rand shape	rand shape
LAA type	$\min_{p(c)}$	$\max_{p(k \neq c)}$	$\max_{p(k \neq c)}$
epsilon	0.02	0.001	0.001

Table A.4: Hyper-parameters to train ObsNet on the different datasets.

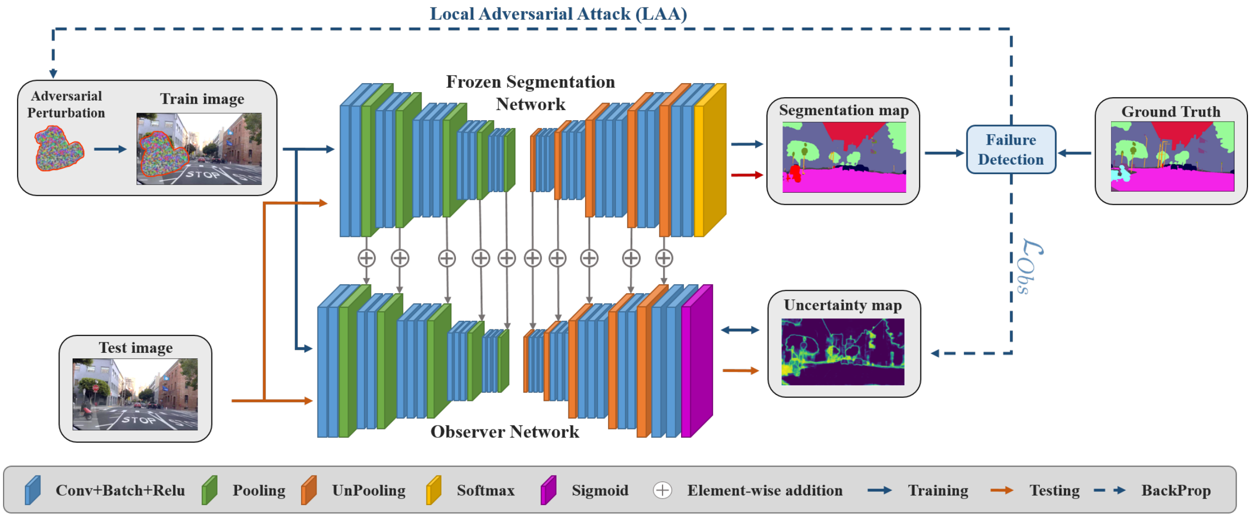


Figure A.4: **Detailed architecture. Training (blue arrow)** The *Segmentation Network* is frozen. The input image is perturbed by a local adversarial attack. Then the *Observer Network* is trained to predict *Segmentation Network*'s errors, given the images and some additional skip connections. **Testing (red arrow)** No augmentation is performed. The *Observer Network* highlights the out-of-distribution sample, here a motorcycle. To compute the uncertainty map, the *Observer Network* requires only one additional forward pass compared to the standard segmentation prediction.

A.2.2 ablation study on adversarial attacks

We outline most of the experiments we make on LAA. First, there are two different kinds of setups, we can either minimize the prediction class (*i.e.*, $\min_{p(c)}$) or maximize instead a different class (*i.e.* $\max_{p(k \neq c)}$), with $p = Seg(x)$ the class vector, $c = \max_p$ the maximum class prediction and k a random class. Then, we attack with five different strategies: all pixels in the image, random sparse pixels, the area of a random class, all pixels in a square patch and all pixels in a random shape. We show in Table A.5 the complete results on CamVid ODD. We can see that that random shape is the most effective. We use the FSGM because it's a well-known and easy-to-use adversarial attack. Since our goal is to hallucinate OOD objects, we believe the location and the shape of the attacked region are the important part.

	Type	Fpr95Tpr ↓	AuPR ↑	AuRoc ↑	ACE ↓
	MCDropout	49.3	97.3	90.1	0.463
	ObsNet base	54.2	97.1	89.1	0.396
$\min_{p(c)}$	all pixels	53.2	97.1	89.5	0.410
	sparse pixels	61.1	97.1	89.2	**0.387**
	class pixels	45.6	97.3	90.3	0.428
	square patch	47.4	97.3	90.1	0.461
	rand shape	**44.6**	**97.6**	**90.9**	0.446
$\max_{p(k \neq c)}$	all pixels	51.9	97.1	89.6	0.405
	sparse pixels	54.2	97.2	89.6	**0.374**
	class pixels	46.8	97.2	89.9	0.432
	square patch	45.5	**97.4**	90.5	0.464
	rand shape	**44.6**	**97.4**	**90.6**	0.446

Table A.5: **Ablation on adversarial attacks.** We can see that the random shape is the best method to train the Observer.

As shown on Figure A.5, we can see that the best ϵ for the attack is 0.02 with a random shape blit at a random position in the image. We can also see that even with a large ϵ, ObsNet still achieves reasonable performance.

A.2.3 Error detector

The observer is trained to assess whether the prediction differs from the true class (which is always the case for OOD regions), so it also tends to assign low confidence scores for in-domain regions with likely high errors, as shown in Figure A.6. This behavior is not caused by ObsNet, but depends on the accuracy of the main network at test time and should lessen with more accurate networks. This effect shows that our method can be used for error detection, and outperforms all other methods, as illustrated in Table A.6.

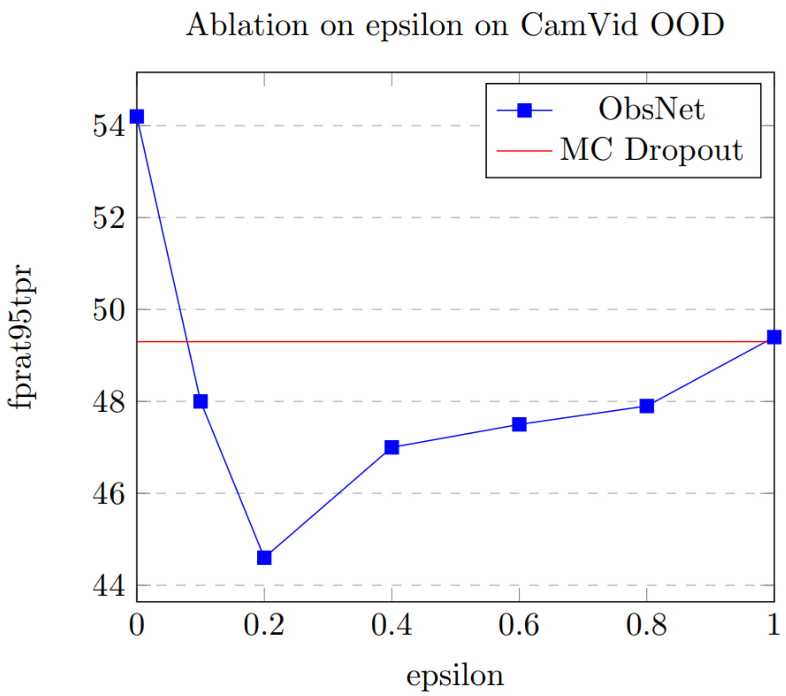


Figure A.5: **Ablation on Epsilon**. Evolution of the Fpr at 95 Tpr for different values of epsilon on CamVid OOD.

Method	Fpr95Tpr ↓	AuPR ↑	AuRoc ↑	ACE ↓
Softmax	61.9	96.5	84.4	0.480
Void	79.9	90.7	67.3	0.504
MCDA	65.8	96.3	83.1	0.440
Temp. Scale	61.9	96.6	84.6	**0.302**
ODIN	58.3	97.2	87.9	0.478
ConfidNet	52.2	97.5	88.6	0.412
Gauss Pert.	60.2	96.8	85.6	0.497
Deep Ensemble	55.3	97.5	88.1	0.343
MCDropout	52.5	97.9	88.5	0.443
ObsNet + *LAA*	**47.7**	**98.1**	**90.3**	0.370

Table A.6: Error detection evaluation on CamVid (best method in bold, second best underlined).

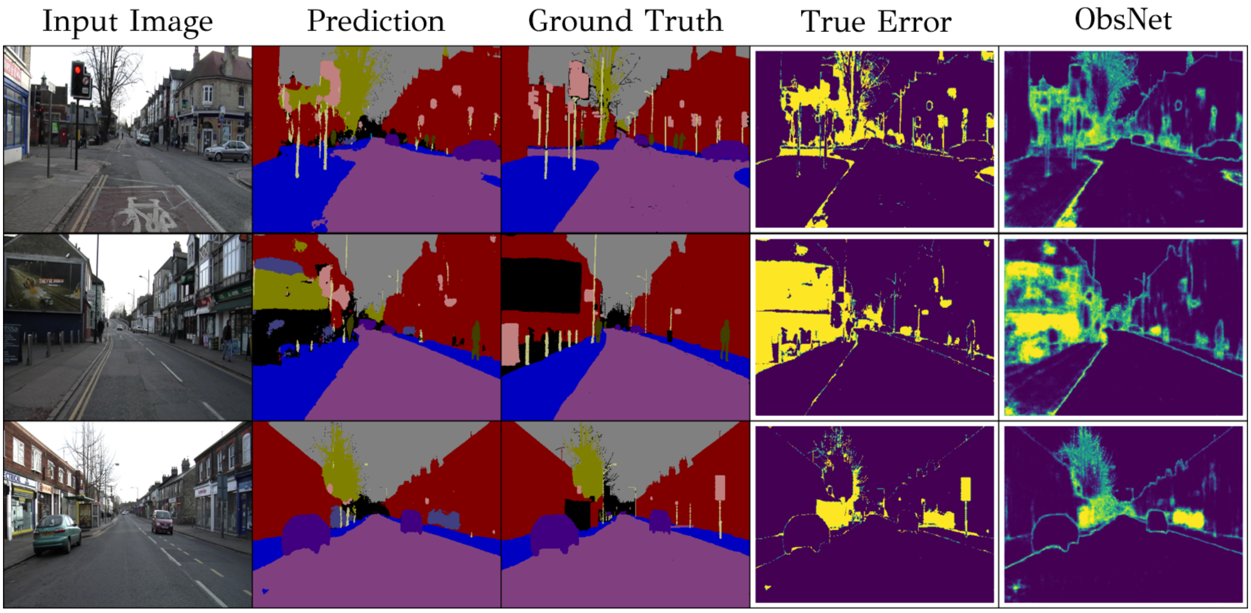


Figure A.6: Evaluation of the error detection on the test set of CamVid. ObsNet prediction is close to real errors even without OOD objects.

A.2.4 *Additional Experiments: DeepLab v3+*

We show on Table A.7, the results on BDD Anomaly with a more recent Deeplab v3+[1] with ResNet-101 encoder. Our methods performs the best, while methods like ConfidNet do not scale when the segmentation accuracy increases as they have fewer errors to learn from.

Method	Fpr95Tpr ↓	AuPR ↑	AuRoc ↑	ACE ↓
Softmax	60.3	95.8	81.4	0.228
Void	68.8	90.2	74.0	0.485
MCDA	68.1	95.1	78.8	0.265
ConfidNet	64.5	95.4	80.9	0.254
Gauss Pert.	61.4	<u>96.1</u>	<u>82.4</u>	<u>0.186</u>
MCDropout	<u>60.0</u>	96.0	82.0	0.219
ObsNet + *LAA*	**58.8**	**96.3**	**83.0**	**0.185**

Table A.7: Evaluation on Bdd Anomaly (best method in bold, second best underlined), with a DeepLab v3+.

A.2.5 *Segment Me If You Can*

In Chapter 4 we introduce some results on the Segment Me If you can Dataset. Here we discuss the new metrics for this specific dataset. The authors introduce instance-wise metrics. They argue that from a practitioner point of view, it is important to detect every anomaly. Pixels-wise metrics do not reflect this need. In fact, pixel-wise metrics, such as AUPR and Fpr95Tpr barely penalize the model when it miss classified a small object for image segmentation. These metrics are more affected by bigger objects. Instance-wise metrics better reflect the capacity of the model to detect components in the image. Here are the three metrics the authors used: component-wise

[1] LC Chen et al., *Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation*, ECCV 2018

intersection over union (sIoU), predictive positive value (PPV) and component-wise F1-score (mean F1). Each metric computes how much the predictions cover the ground-truth regions, independently of whether prediction/ground truth belongs to a single or multiple objects. The scoreboard is regularly updated in the following address: https://segmentmeifyoucan.com/leaderboard.

In Figure A.7, we show additional results on the dataset. We can see that our ObsNet + *LAA* and minor adaptation succeed in segmenting the anomalies nearly perfectly, despite never having seen any bear, or plane during train time.

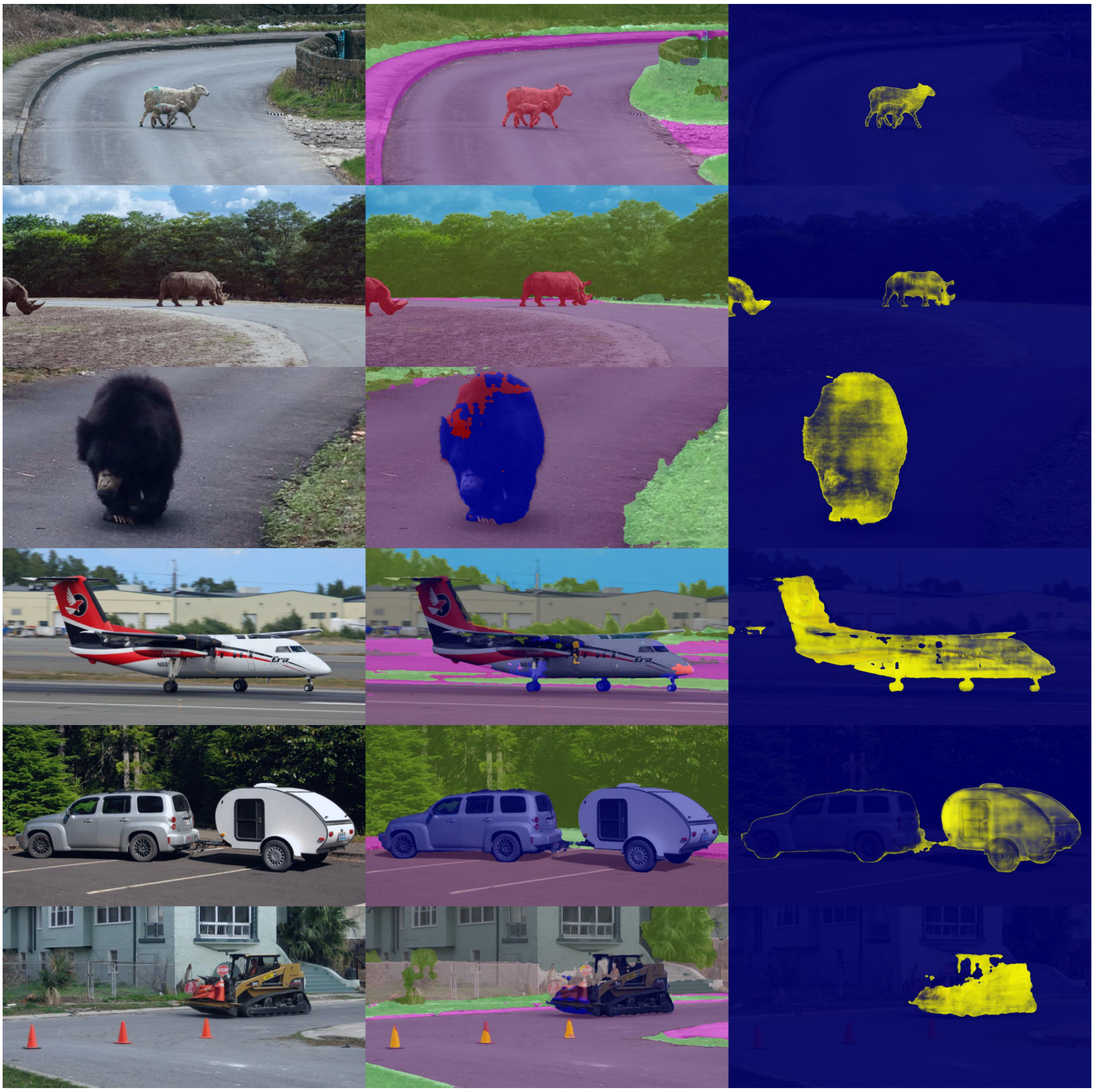


Figure A.7: **SMIYC additional results**: Our ObsNet is able to detect OoD objects such as the bear or the plane.

A.3 CAMVID OOD DATASET

For our experiments, we use urban street segmentation datasets with anomalies withheld during training. Unfortunately, there are few datasets with anomalies in the test set. For this reason we propose the CamVid OOD that will be made public after the review. To design CamVid OOD, we blit random animals in test images of CamVid. We add one different such anomaly in each of the 233 test images. The rest of the 367 training images remain unchanged. The anomalous animals are *bear*, *cow*, *lion*, *panda*, *deer*, *coyote*, *zebra*, *skunk*, *gorilla*, *giraffe*, *elephant*, *goat*, *leopard*, *horse*, *cougar*, *tiger*, *sheep*, *penguin*, and *kangaroo*. Then, we add them to a 13th class which is *animals/anomalies* as the corresponding ground truth of the test set.

This setup is similar to the Fishyscape dataset, without the constraint of sending a Tensorflow model online for evaluation. Thus, our dataset is easier to work with. We present some examples of the anomalies in Figure A.8 with the ground truth highlighted in cyan. We have presented in [BBPB21], the results of our method plus several other frameworks that can serve as a baseline for others.

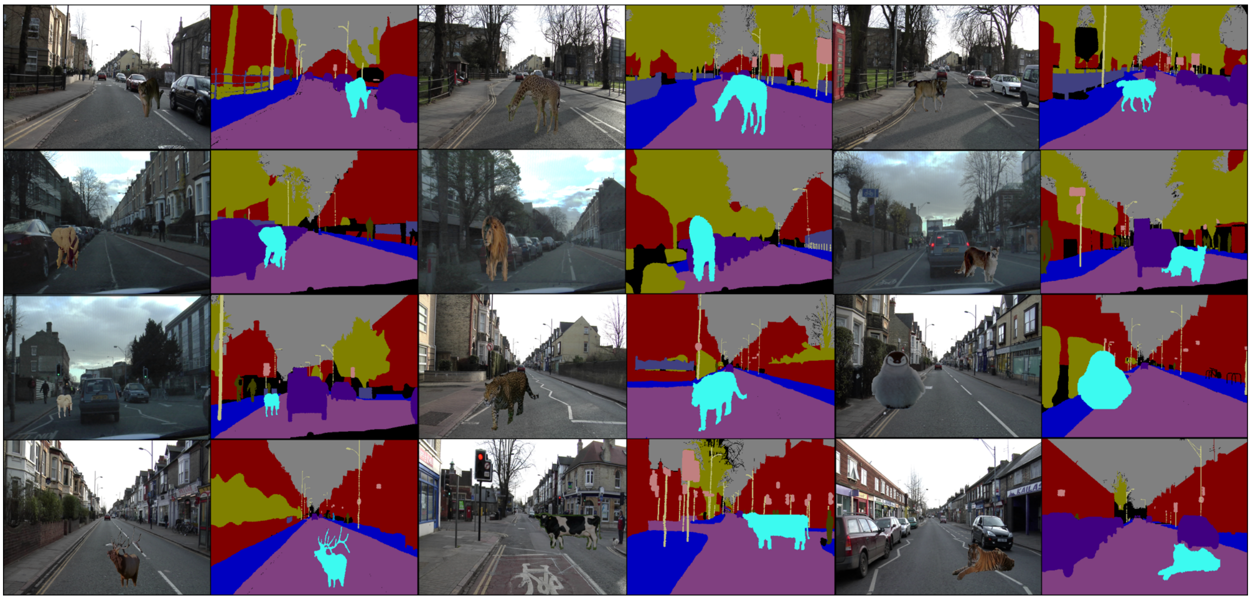


Figure A.8: Examples of our dataset with anomalies and the ground truth.

A.4 LEARNING CAR

In Figure A.9, we show additional results on the Learning Car, from the left camera. We see that most of the areas with low confidence correspond are under the cars, which should be predicted as a parking zone (color black). But as we can see these parking places are wrongly classified as the road (in purple). Moreover, the spot on the road is also wrongly predicted as a text marking but the segmentation network.

To show our evolution on the quality of our framework, we

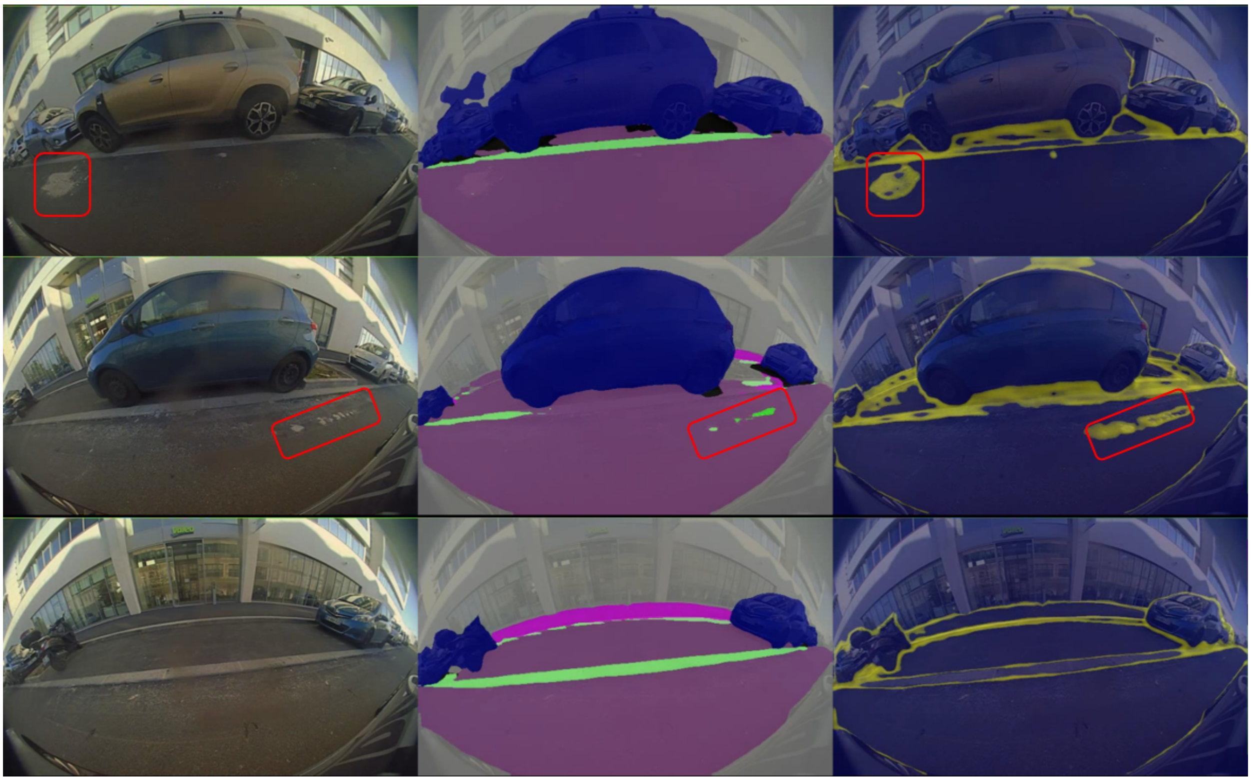


Figure A.9: Additional view from online testing of the Demo Car.

highlight in Figure A.10 the improvement between each round. From the first row (First round) to the last one, we can see that both networks produce better results in the last row that in the first row. In the beginning, the segmentation network prediction was overflowing on the object and not very accurate. Moreover, the ObsNet overreacted on the prediction of the segmentation network. The last row shows that our ObsNet is much more precise, and the segmentation is more accurate.

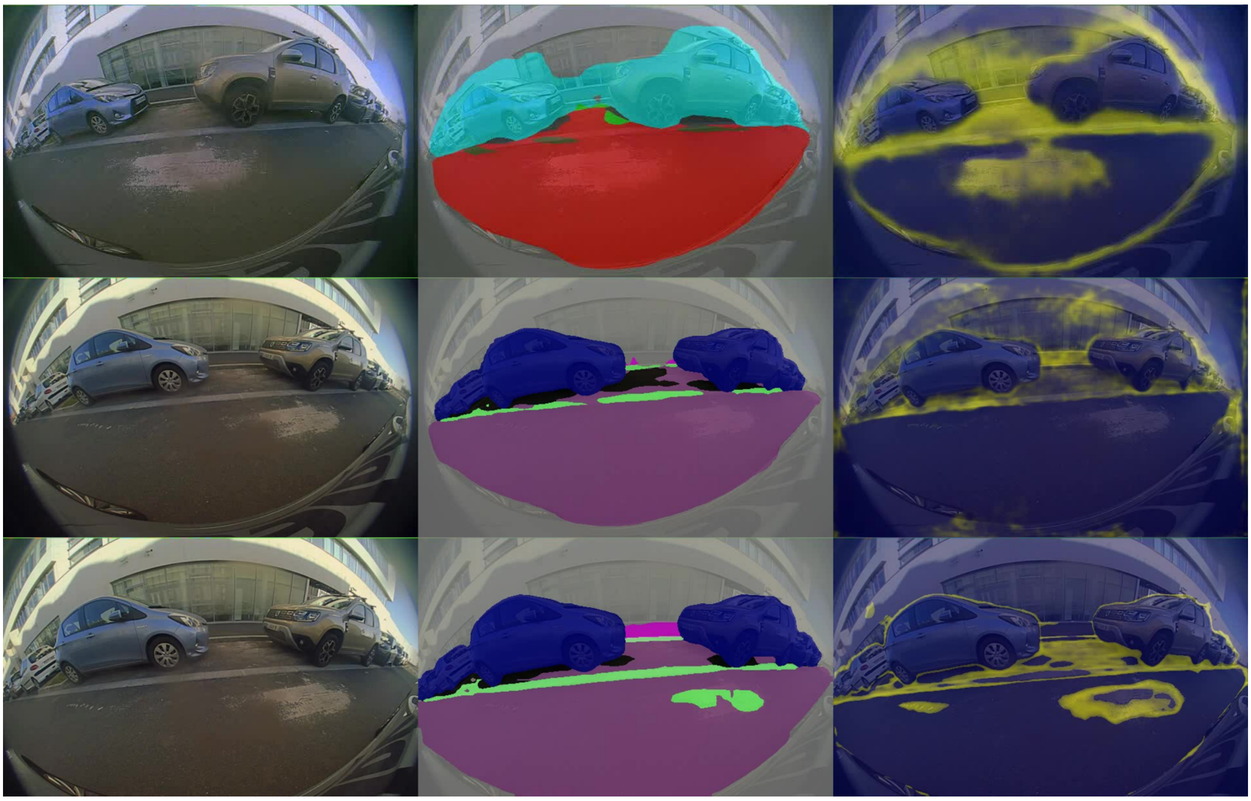


Figure A.10: Evolution of the improvement of our Demo Car.