Apprentissage profond robuste pour la conduite autonome

Charles Corbière

HESAM UNIVERSITÉ

le cnam

ÉCOLE DOCTORALE SCIENCES DES MÉTIERS DE L'INGÉNIEUR
Centre d'études et de recherche en informatique et communications

THÈSE

présentée par : **Charles CORBIÈRE**
soutenue le : **16 mars 2022**

pour obtenir le grade de : **Docteur d'HESAM Université**

préparée au : **Conservatoire national des arts et métiers**

Discipline : **Mathématiques, Informatique et Systèmes**

Spécialité : **Informatique**

Robust Deep Learning for Autonomous Driving

THÈSE dirigée par :
M. THOME Nicolas, Professeur, Cedric, Cnam

et co-encadrée par :
M. PÉREZ Patrick, Directeur scientifique, valeo.ai

Jury

Mme Florence d'ALCHÉ-BUC	Professeur, LTCI, Télécom Paris	Présidente du jury
M. Stéphane CANU	Professeur, LITIS, INSA Rouen	Rapporteur
M. Graham TAYLOR	Professeur, University of Guelph	Rapporteur
M. Alex KENDALL	Chercheur associé, University of Cambridge et co-fondateur, Wayve	Examinateur
M. Matthieu CORD	Professeur, ISIR, Sorbonne Université et chercheur senior, valeo.ai	Examinateur
M. Patrick PÉREZ	Directeur scientifique, valeo.ai	Co-directeur de thèse
M. Nicolas THOME	Professeur, Cedric, Cnam	Directeur de thèse

THÈSE

Remerciements

Je souhaite remercier ici les nombreuses personnes qui m'ont accompagné, orienté, conseillé et soutenu tout au long de cette thèse.

Tout d'abord, je tiens à remercier mon directeur de thèse Nicolas Thome pour m'avoir accueilli au sein de son équipe et de m'avoir encadré pendant ces presque quatre ans. Sa disponibilité, son accompagnement et nos nombreuses discussions techniques m'ont été précieux. J'ai toujours pu compter sur son soutien à l'approche des échéances et l'en remercie encore.

Je voudrais également remercier mon co-directeur de thèse Patrick Pérez pour avoir cru en moi au moment de la création valeo.ai et des premiers recrutements au sein de son équipe de recherche. Patrick m'a été d'une grande aide via ses conseils avisés et pertinents tout au long de la thèse.

Conduire mes travaux entre l'équipe VERTIGO au Cnam et l'équipe de valeo.ai a été pour moi une expérience très enrichissante. Je remercie tous les chercheurs, doctorants et stagiaires de valeo.ai avec qui j'ai eu de nombreux échanges instructifs. En particulier, je tiens à remercier Tuan-Hung Vu et Antoine Saporta pour notre travail en commun sur ConDA, Andrei Bursuc pour ses conseils et son impressionnante capacité à indiquer des papiers pertinents de la littérature, et bien évidemment Arthur Ouaknine, Simon Roburin et Maxime Bucher pour les bons moments passés ensemble. Côté VERTIGO, je remercie mes camarades Laura Calem, Olivier Petit et Vincent Le Guen avec qui j'ai partagé une grande partie de mes aventures au Cnam. Mais aussi plus récemment Elias Ramzi, Loïc Themyr, Perla Doubinski, Yannis Karmin, Clément Rambour et Nicolas Audebert. Enfin, je remercie Marc Lafon pour tous nos échanges sur l'incertitude depuis son stage et lui souhaite bonne chance pour sa thèse.

Je remercie le jury pour l'intérêt porté à mes travaux, pour avoir accepté de relire ma thèse et d'assister à la soutenance qui finalise le travail de ces presque quatre années de recherche.

Pour finir, je tiens à remercier bien évidemment ma famille et mes amis pour leur soutien et leur présence tout au long de la thèse. Tout particulièrement Marlène Pivard avec qui je partage ma vie pour son importance vitale à mes côtés et ses encouragements. Mais aussi mes parents, Evelyne et Jean-Paul, qui m'ont toujours poussé à réaliser mes rêves tout en me laissant m'épanouir dans ce qui me plaisait. Enfin, la grande équipe des Equidés et la Troupe du Rire pour ces moment chaleureux d'une vie heureuse.

Abstract

The last decade's research in artificial intelligence and hardware development had a significant impact on the advance of autonomous driving. Yet, safety remains a major concern when it comes to deploying such systems in high-risk environments. Modern neural networks have been shown to struggle to correctly identify their mistakes and to provide over-confident predictions instead of abstaining when exposed to unseen situations. Progress on these issues is crucial to achieve certification from transportation authorities but also to arouse enthusiasm from users.

The objective of this thesis is to develop methodological tools which provide reliable uncertainty estimates for deep neural networks. In particular, we aim to improve the detection of erroneous predictions and anomalies at test time.

First, we introduce a novel target criterion for model confidence, the true class probability (TCP). We show that TCP offers better properties than current uncertainty measures for the task of failure prediction. Since the true class is by essence unknown at test time, we propose to learn TCP criterion from data with an auxiliary model (*ConfidNet*), introducing a specific learning scheme adapted to this context. The relevance of the proposed approach is validated on image classification and semantic segmentation datasets, demonstrating superiority with respect to strong uncertainty quantification baselines on failure prediction.

Then, we extend our learned confidence approach to the task of domain adaptation for semantic segmentation. A popular strategy, self-training, relies on selecting predictions on the unlabeled data and re-training a model with these pseudo-labels. Termed *ConDA*, the proposed adaptation improves self-training methods by providing effective confidence estimates used to select pseudo-labels. To meet the challenge of domain adaptation, we equipped the auxiliary model with a multi-scale confidence architecture and supplemented the confidence loss with an adversarial training scheme to enforce alignment between confidence maps in source and target domains.

Finally, we consider the presence of anomalies and we tackle the ultimate practical objective of jointly detecting misclassification and out-of-distributions samples. To this end, we introduce *KLoS*, an uncertainty measure based on evidential models and defined on the class-probability simplex. By keeping the full distributional information, KLoS captures both uncertainty due to class confusion and lack of knowledge, which is related to out-of-distribution samples. We further improve performance across various image classification datasets by using an auxiliary model with a learned confidence approach.

Résumé

Le véhicule autonome est revenu récemment sur le devant de la scène grâce aux avancées fulgurantes de l'intelligence artificielle. Pourtant, la sécurité reste une préoccupation majeure pour le déploiement de ces systèmes dans des environnements à haut risque. Il a été démontré que les réseaux de neurones actuels peinent à identifier correctement leurs erreurs et fournissent des prédictions sur-confiantes, au lieu de s'abstenir, lorsque exposés à des anomalies. Des progrès sur ces questions sont essentiels pour obtenir la certification des régulateurs mais aussi pour susciter l'enthousiasme des utilisateurs.

L'objectif de cette thèse est de développer des outils méthodologiques permettant de fournir des estimations d'incertitudes fiables pour les réseaux de neurones profonds. En particulier, nous visons à améliorer la détection des prédictions erronées et des anomalies lors de l'inférence. Tout d'abord, nous introduisons un nouveau critère pour estimer la confiance d'un modèle dans sa prédiction : la probabilité de la vraie classe (TCP). Nous montrons que TCP offre de meilleures propriétés que les mesures d'incertitudes actuelles pour la prédiction d'erreurs. La vraie classe étant, par essence, inconnue à l'inférence, nous proposons d'apprendre TCP avec un modèle auxiliaire (*ConfidNet*), introduisant un schéma d'apprentissage spécifique adapté à ce contexte. La qualité de l'approche proposée est validée sur des jeux de données de classification d'images et de segmentation sémantique., démontrant une supériorité par rapport aux méthodes de quantification incertitude utilisées pour la prédiction de d'erreurs.

Ensuite, nous étendons notre approche d'apprentissage de confiance à la tâche d'adaptation de domaine. Une stratégie populaire, l'auto-apprentissage, repose sur la sélection de prédictions sur données non étiquetées puis le réentraînement d'un modèle avec ces pseudo-étiquettes. Appelée *ConDA*, l'adaptation proposée améliore la sélection de pseudo-labels grâce à des meilleures estimations de confiance. Afin de relever le défi de l'adaptation de domaine, nous avons équipé le modèle auxiliaire d'une architecture multi-échelle et complété la fonction de perte par un schéma d'apprentissage contrastif afin de renforcer l'alignement entre les cartes de confiance des domaines source et cible.

Enfin, nous considérons la présence d'anomalies et nous attaquons au défi pratique de la détection conjointe des erreurs de classification et des échantillons hors distribution. A cette fin, nous introduisons *KLoS*, une mesure d'incertitude définie sur le simplexe et basée sur des modèles évidentiels. En conservant l'ensemble des informations de distribution, KLoS capture à la fois l'incertitude due à la confusion de classe et au manque de connaissance du modèle, cette dernière type d'incertitude étant liée aux échantillons hors distribution. En utilisant ici aussi un modèle auxiliaire avec apprentissage de confiance, nous améliorons les performances sur divers ensembles de données de classification d'images.

Table of Contents

List of Tables

List of Figures

Nomenclature

Acronyms

i.i.d.	independent and identically distributed
AI	Artificial intelligence
BNN	Bayesian neural network
ConvNet	Convolutional neural network
CV	Computer vision
DL	Deep learning
DNN	Deep neural network
ENN	Evidential neural network
KL	Kullback-Leibler
MC	Monte-Carlo
MCP	Maximum class probability
ML	Machine learning
MLP	Multi-layer Perceptron
NLL	Negative log-likelihood
NLP	Natural language processing
NN	Neural network
OOD	Out-of-distribution
TCP	True class probability

Greek symbols

$\boldsymbol{\alpha}$	concentration parameters of a Dirichlet distribution
$\boldsymbol{\omega}$	auxiliary network's parameters
$\boldsymbol{\varphi}$	confidence module's parameters
$\boldsymbol{\pi}$	parameters of a categorical distribution
$\boldsymbol{\theta}$	classification network's parameters

Roman symbols

$\mathbb{H}[X]$	entropy of random variable X
$\mathbb{I}[X, Y]$	mutual information between random variables X and Y
$\mathcal{D}$	training dataset
$\mathcal{H}$, h	hypothesis space, hypothesis
$\mathcal{L}$, ℓ	loss function
$\mathcal{X}$, $\boldsymbol{x}$, $\boldsymbol{x}_n$	input space, input
$\mathcal{Y}$, y, y_n	output space, output
$\mathbb{1}[\cdot]$	indicator function
$\mathbb{E}[X]$	expected value of random variable X
$\mathbb{KL}(p \mid\mid q)$	KL divergence from distribution p to q
$\mathbb{R}$	set of real numbers
$\mathcal{N}$	Normal distribution
$\mathrm{Var}[X]$	variance of random variable X
P, p	probability measure, density function
X, Y	random variables
K	number of classes

Chapter 1

Introduction

1.1 Context

From Rosenblatt's Perceptron [1] to the rise of attention-based neural networks ('Transformers') [2], the field of artificial intelligence (AI) has been experiencing alternative periods of hype cycles followed by disappointment, reduced funding and interest. However, the current advances in deep learning (DL) [3] not only raised interest among AI researchers but also drive technological progress in many science disciplines, including physics, biology, as well as in manufacturing and other industrial applications[1]. Since the stunning victory of a convolutional neural network architecture, AlexNet [4], at the Large Scale Visual Recognition Challenge (LSVRC) in 2012, deep learning is now ubiquitous in the fields of computer vision (CV) [5, 6, 7], natural language processing (NLP) [8, 9], speech recognition [10, 11] and reinforcement learning [12] (see Fig. 1.1), accounting for a larger portion of papers published in each respective conference.

Recent breakthroughs in computer vision thanks to deep learning largely explain the spectacular revival of autonomous driving with major tech players such as Waymo, Tesla, Baidu and Yandex investing in self-driving car programs. Deep learning is now being used in various autonomous driving modules. In perception, convolutional neural networks process information coming from visual cameras to understand a scene and detect crucial aspects of the environment in real-time [18]: nature and position of other vehicles, bicycles, pedestrians; position and meaning of road markings, signs, lights; navigable space; position of obstacles; etc. To enrich scene analysis, perception systems in autonomous driving complements traditional cameras with active sensors such as radar sensors and LiDARs (Light Detection and Ranging) measuring sparse but direct tri-dimensional aspects of dynamic scenes. Perception with these sensors also tend to rely more and more and deep learning models [19, 20], hence they can be combined to improve the quality of the higher-level decision system via sensor fusion [21]. As one of the world leaders in automotive sensors, Valeo, which is funding this thesis, is positioned at the heart of this current revolution, developing high-quality LiDARs with their SCALA technology. While deep learning is mostly used in perception modules, promising research

[1] To grasp the impact of AI on today's world, the 'State of AI Report' published every year is a thorough compilation of developments in research, industry and politics: https://www.stateof.ai.

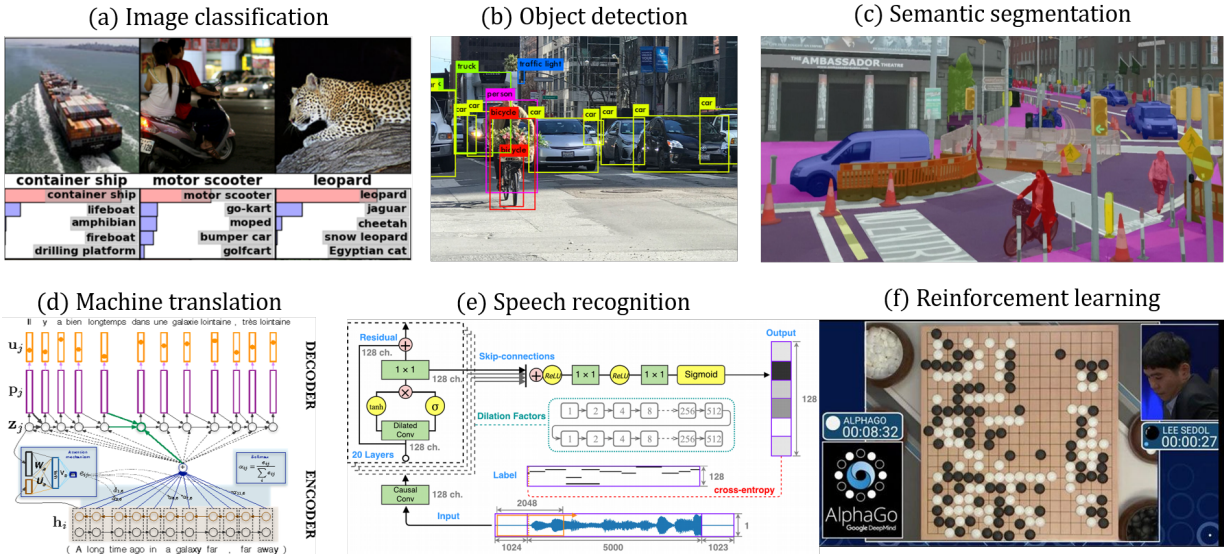


Figure 1.1: **Examples of successful applications of AI:** in computer vision (a,b,c), natural language processing (d), speech recognition (e) and reinforcement learning mastering the game of Go (f). *Image credits to Krizhevsky et al. [4], Hui [13], Neuhold et al. [14], van den Oord et al. [15], Garcia-Martinez et al. [16] and DeepMind [17].*

aims to apply it also in planning, such as trajectory forecasting for the objects present in the car's environment. End-to-end approaches, from perception to control, by predicting steering angle and acceleration is also starting to emerge with deep reinforcement learning [22]. While these incredible progresses are undeniable, at the time of writing of this manuscript, robotaxis are not deployed yet and many challenges still need to be resolved for autonomous driving before large scale commercialisation, in particular by addressing issues related to safety.

1.2 Motivations

Despite its clear benefits to many applications, the deployment of machine learning (ML) models in high-stakes environments raises serious questions about its impacts on our society. *AI safety* [23, 24] is an area of research that aims to identify causes of unintended and harmful behaviour in machine learning systems and to develop tools to ensure these systems work safely and reliably. Such behaviour may emerge from machine learning systems [25] when:

- exposed to unusual situations, distributional changes on inputs [26] or long-tail scenarios [27] (*'Robustness'*);
- subjected to corruptions during training, such as data poisoning [28], or during inference with adversarial attacks from malicious opponents [29] (*'External Safety'*);
- the learning objective is not aligned with human values – which may be hard to specify though

– or the model uses shortcuts during optimization [30] which are not transferable to more challenging testing conditions (*'Alignment'*).

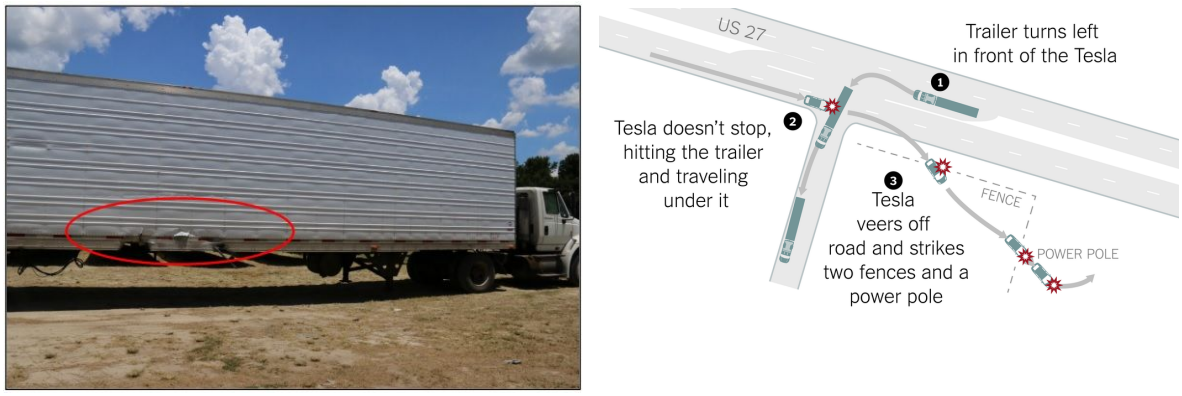
When deployed in the wild, a machine learning system should be able to detect these hazardous cases (*'Monitoring'*) and estimate whether it is confident in its prediction in order to prevent accidents.

Accidents occurring with autonomous cars are typical examples where repercussions can be catastrophic. During the perception step, low-level feature extraction such as image segmentation and image localisation are used to process raw sensory inputs [31]. Outputs of such models are then fed into higher-level decision-making procedures. However, mistakes done by lower-level machine learning components can propagate up the decision-making process and lead to devastating results. One striking example is the tragic incident which happened on May 7$^{\text{th}}$ 2016 near Williston (Florida, USA) and resulted in the first death caused by a car with highly automated driving assistance [32]. Tesla, the car manufacturer, stated that the accident originated from the vision system which incorrectly classified the white side of a turning trailer truck as a bright sky[2] (Fig. 1.2). Visual signals can indeed be fooled or not adapted in some arduous conditions such as heavy raining, bright sky or night time. As the NTSB noted in their report [32], "introducing automation in complex and unstructured environment is very challenging" and they recommended to manufacturers of vehicles equipped with automation systems to "incorporate system safeguards that limit the use of automated vehicle control systems to those conditions for which they were designed". Since then, several other accidents and crashes with self-driving cars continue to occur, including the first recorded case of a pedestrian fatality in 2018 [33]. Among the factors explaining the collusion, the NTSB report stated that the system of the Uber test vehicle failed to recognize the woman, first identifying her as an unknown object, next as a vehicle, then as the bicycle she was pushing. Correctly monitoring and assessing system confidence in its predictions appears to be more than necessary to safely deploy ML models in high-stakes environments [34]. Progress on these issues is crucial in autonomous driving to achieve certification from transportation authorities but also to arouse enthusiasm from users.

Knowing when a model doesn't know is important to improve trustworthiness and safety [34]. By assigning high levels of uncertainty to erroneous predictions, a ML system could have been able to avoid previous catastrophes by sending a trigger alarm or giving back control to users. Related to the example of sensor fusion mentioned earlier, when evaluating high uncertainty for a prediction output by the visual camera during night time, a system could decide to rely more on active sensors predictions which are more robust to these light conditions. One key of a good uncertainty-based fusion of multiple sensor's predictions is to ensure that probabilities are well *calibrated* (see Section 2.3.3 for details about the meaning of probability calibration). One would also like the confidence criterion to correlate successful predictions with high values. Some paradigms, such as self-training with pseudo-labeling [37, 38], consist in picking and labeling the most confident samples before retraining the network accordingly. The performance improves by selecting successful predictions thanks to an accurate confidence criterion.

For practical systems, it may also be important to understand what the model does not know.

[2] https://www.tesla.com/blog/tragic-loss
[3] https://www.nytimes.com/interactive/2016/07/01/business/inside-tesla-accident.html



(a) Photo of the involved white truck

(b) Outline of the accident

Figure 1.2: Tesla's deadly crash in May 2016: photo of the white truck confused with bright sky by Tesla's vision system (Fig. 1.2a) and outline of the accident (Fig. 1.2b). Image credits: New York Times[3].

Common classification in uncertainty estimation distinguishes two types of uncertainty. Aleatoric uncertainty, also termed *data* uncertainty, is due to the inherent stochasticity of the outcome of an experiment. This type of uncertainty arises due to class confusion, sensor noise, or non-discriminant features, such as in Fig. 1.3b where the model confuses a cat on a chair with a bird (*known-unknown*). Epistemic uncertainty refers to uncertainty caused by a lack of knowledge of the model, for instance an input from another distribution or from an unknown class (*unknown-unknown*), as illustrated in Fig. 1.3c. A good estimation of uncertainty is also useful to discriminate unusual situations from regular inputs, such as driving conditions in snowy roads in Russia while the car's ML system has been trained on data collected in California. By providing more data to a model, we can reduce this uncertainty. Identifying samples with large epistemic uncertainty is also beneficial for classification improvements in active learning [39] and for efficient exploration in reinforcement learning [40].

While confidence estimation[4] has a long history in machine learning [41, 42, 43, 44], a series of recent works showed that modern neural networks (NNs) suffer from several conceptual drawbacks which make them unreliable [45, 46, 40, 47, 48]. In classification, the output of the last layer is fed to the softmax function, which produces a probability distribution over class labels. However, with modern NNs, these probabilities have been shown to be non-calibrated [49] which makes NNs unsuited for a larger decision-making pipelines. To obtain uncertainty estimates, a widely used baseline with NNs is to take the value of the predicted class' probability [45], namely the *maximum class probability* (MCP), or to use the predicted entropy given the predicted probability distribution. But as shown in Fig. 1.3, these measures can produce high confident predictions for in-distribution errors, which hardens error detection or selective classification where one would filter out these samples. When deployed in real conditions, machine learning models often encounter samples that are away from the training distribution, such as covariate shift or new classes. However, NNs are known to be brittle to

[4]The terms *uncertainty* and *confidence* estimation are used alternatively in this manuscript, the latter referring to the opposite of uncertainty.

(a) Correct prediction (b) In-distribution error (c) Out-of-distribution sample

Figure 1.3: Comparison of confidence estimates using MCP or predictive entropy for a model trained on CIFAR-10 dataset [35] between a correct prediction (Fig. 1.3a), an in-distribution error (Fig. 1.3b), and an out-of-distribution (OOD) sample taken from SVHN dataset [36]. The model assigned a higher MCP value and a lower entropy, hence a large confidence score, to the error and the OOD sample than to the correct prediction, which is not desirable in uncertainty estimation.

distribution shifts [26] with their prediction performances severely decreasing as they tend to rely on spurious correlations [30]. Multiple works also showed that NNs provide over-confident predictions for samples far from the training data [48], including fooling images [46] or adversarial inputs [50]. Fig. 1.3c shows the example of an out-of-distribution sample taken from SVHN dataset [36] and predicted as a bird with high confidence by a model trained on CIFAR-10 dataset [35].

The development of principled methods for deep learning models such as Bayesian neural networks (BNNs) [51, 40, 52] and ensembles [53] enable deep neural networks to capture epistemic uncertainty more accurately. Such as with ensemble, predictions with BNNs are obtain by averaging multiple forward pass due to their finite approximation of the predictive distribution (Monte Carlo sampling). But this comes at the expense of an increased computational cost to obtain uncertainty estimates. In addition, recent works [54, 55] show they still fall short in giving useful estimates of their predictive uncertainty. Despite these progress in uncertainty estimation, there remains a gap to be filled in detecting in-distribution errors and abnormal samples to avoid serious repercussions when deploying a fleet of driverless robotaxis.

1.3 Contributions and outline

In this thesis, we tackle the challenge of providing reliable uncertainty estimates along with deep neural network predictions with applications for autonomous driving. In particular, we aim to improve the detection of erroneous predictions at test time by distinguishing them from correct ones. Errors can be of different natures and the following contributions will firstly address the task of in-distribution misclassification detection, also known as *failure prediction* (Chapter 3). Along with the detection of such examples at test time, we also elaborate on leveraging our proposed approach in the case of

domain adaptation (Chapter 4), where self-training approaches rely on uncertainty estimates to select samples in the re-labelling phase. Finally, we consider the presence of anomalies and consequently propose to detect both in-distribution errors and out-of-distribution samples with a single uncertainty measure (Chapter 5).

Outline. In regards with the challenges mentioned above, our contributions are the following:

- Chapter 3: Learning A Model's Confidence via An Auxiliary Model
  After exposing the limits of standard uncertainty measures with deep neural networks in classification, we define a new confidence criterion, *True Class Probability*, which provides theoretical guarantees and empirical evidence for confidence estimation. We propose to design an auxiliary neural network, coined *ConfidNet*, which aims to learn this confidence criterion from data. An exploration of the classification-with-rejection framework strengthens the rationale of the proposed approach. Extensive experiments are conducted for validating the relevance of the proposed approach on image classification and semantic segmentation datasets. An analysis of the impact of loss function, criterion and learning scheme is also presented.

- Chapter 4: Self-Training with Learned Confidence for Domain Adaptation
  Self-training has recently proven a potent strategy to improve the effectiveness of Unsupervised Domain Adaptation (UDA) in semantic segmentation. This line of work mostly relies on the generation of pseudo-labels over the unannotated target domain to incorporate target images and learn a better segmentation adaptation model. A crucial issue is to base the pseudo-label selection on reliable confidence measures. We propose to adapt our learned confidence approach to estimate the confidence of the segmentation network in its predictions and to use these confidence estimates as a criterion for pseudo-label selection. Named *ConDA*, the proposed adaptation of our original approach to this new context includes two further contributions: (1) an adversarial training scheme to reduce the gap between confidence maps in source and target domains; (2) an enhanced architecture for the confidence network to perform multi-scale confidence estimation. We show that this strategy produces more accurate pseudo-labels and outperforms strong baselines on challenging UDA segmentation benchmarks.

- Chapter 5: Simultaneous Detection of Misclassifications and Out-of-Distribution Samples with Evidential Models
  Beyond errors due to misclassifications by deep neural networks, models may encounter data that is unlike the model's training data when deployed in the wild. In this chapter, we tackle the task of jointly detecting errors and anomalies in a single uncertainty measure. To this end, we leverage the second-order uncertainty representation provided by evidential models [56, 57], a Bayesian method based on subjective logic, and we introduce *KLoS*, a KL-divergence criterion defined on the class-probability simplex. We show that KLoS quantifies in-distribution and out-of-distribution uncertainty more accurately than first-order measures such as the predictive

entropy. In a similar spirit to the previous contribution, we design an auxiliary neural network, *KLoSNet*, to learn a refined measure directly aligned with the evidential training objective. Our experiments show that KLoSNet acts as a class-wise density estimator and outperforms current first-order and second-order uncertainty measures to simultaneously detect misclassifications and OOD samples. We study the impact of the choice of OOD training samples on our method and concurrent measures, which sheds a new light on the impact of the vicinity of this data with OOD test data.

Before delving in the core of the thesis, we present in Chapter 2 an overview of the recent progress in uncertainty estimation with deep neural networks, including a thorough characterization of the source of uncertainty, methods to model uncertainty and the various ways of evaluating the quality of uncertainty estimates. Finally, in Chapter 6, we conclude this thesis with an overview of the contributions of each chapter and we propose several interesting perspectives for future works.

1.4 Related publications

This thesis is based on the material published in the following papers:

Publication	Chapter
Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, Patrick Pérez. "Addressing Failure Prediction by Learning Model Confidence", in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.	3
Charles Corbière, Nicolas Thome, Antoine Saporta, Tuan-Hung Vu, Matthieu Cord, Patrick Pérez. "Confidence Estimation via Auxiliary Models", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.	3,4
Charles Corbière, Marc Lafon, Nicolas Thome, Matthieu Cord, Patrick Pérez. "Beyond First-Order Estimation with Evidential Models for Open-World Recognition", *ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning.*	5

Chapter 2

Uncertainty Estimation in Deep Learning for Classification

Chapter Abstract

In this chapter, we propose a general overview of the literature regarding uncertainty estimation with deep neural networks. The sources of uncertainties are primarily discussed in Section 2.1 with a formalisation in the context of supervised learning. Traditionally, uncertainty is modeled in a probabilistic way and probabilistic methods have always been perceived as the natural tool to handle uncertainty. In particular, the Bayesian framework provides a probabilistic representation of uncertainty by incorporating degrees of belief. After reviewing the basic concepts of deep learning, we will see in Section 2.2 how Bayesian approaches model the different sources of uncertainty. In addition, we enumerate the measures proposed along with standard and Bayesian approaches to quantify uncertainty. We describe their behavior and, above all, their limits that will be addressed in this thesis. While reliable uncertainty estimates are crucial in many safety-critical applications, their evaluation remains challenging as the 'ground truth' uncertainty estimates are usually not available. One would expect them to truly reflect probabilities in a multi-sensor perception system where late fusion rely on these probabilities. On the other hand, the goal may be to detect errors or anomalies and thus a reliable ranking between correct predictions and abnormal samples is desired. In Section 2.3, we present the existing tasks commonly used in the literature to evaluate the quality of uncertainty estimates with deep neural networks.

Contents

2.1 Sources of uncertainty

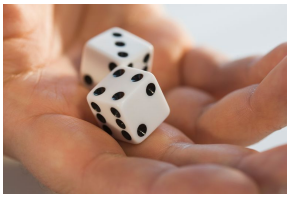
Uncertainty can arise from various reasons and may require a different handling depending of their nature. After introducing a traditional categorization of sources in uncertainty in machine learning literature, we dive into a more precise identification within the setting of supervised learning.

2.1.1 General characterisation

The nature of uncertainties has been a topic of discussion by statisticians [58], economists [59], engineers [60] and other specialists facing random processes. In the machine learning literature [61, 62, 63, 64, 57], sources of uncertainty are traditionally characterized as either *aleatoric* or *epistemic*. When an outcome of an experiment may variate due to intrinsic randomness of a phenomenon, *e.g.* coin flipping, we refer to aleatoric uncertainty. Another term used alternatively is *data uncertainty*, which emphasizes that the stochasticity is inherent to the observed object rather than the model. This type of uncertainty arises due to class confusion, noise, non-discriminant features *e.g.*, sun glare or rain drop in autonomous driving images (see Fig. 2.1).

Epistemic uncertainty refers to uncertainty caused by a lack of knowledge, hence intricately linked to the model representing the random process. For models trained on computer vision tasks, we show some examples of samples with large epistemic uncertainty in Fig. 2.2. In contrast with aleatoric uncertainty, it can be reduced by providing additional information, here in the form of training data. To illustrate the distinction between the types of uncertainty, let us consider weather forecasting [66] which discriminates a predicted probability score from the uncertainty in that prediction: "[...] a weather forecaster can be very certain that the chance of rain is 50%; or her best estimate at 20% might be very uncertain due to lack of data.". Here, the amount of aleatoric uncertainty corresponds to 50%, due to the complex and multi-variate factors resulting to rain; while the weather forecaster


(a) Rain drops and sun glare



(b) Dice roll


(c) Dog/wolf confusion

Figure 2.1: **Examples of aleatoric uncertainty in computer vision.** (a) Adverse weather conditions in autonomous driving [65] harden perception for image-based modules; (b) a dice roll is inherently stochastic due to the extreme sensitivity to initial conditions that cannot be measured with sufficient precision; (c) although being a breed of dog, huskies share many similarities with wolves.

acknowledges not being confident in his prediction (20%), which relates to epistemic uncertainty. With machine learning predictions, epistemic uncertainty is expected to be high for samples far from the training distribution. Also known as distribution shifts, mismatch between input data in deployment stage and the original training distribution can arise in many real-world tasks [26]. By providing more data to a model, we can reduce this type of uncertainty. In summary, epistemic uncertainty refers to the reducible part of the total uncertainty, whereas aleatoric uncertainty refers to the non-reducible part.


(a) Snowy road


(b) Rickshaw


(c) Drawing of a dog

Figure 2.2: **Examples of epistemic uncertainty in computer vision.** An autonomous car can be exposed to unseen conditions (Fig. 2.2a) or new semantic class (Fig. 2.2b). While being only trained on natural images, an ImageNet-trained classifier might encounter images of known classes but with different rendering *e.g.*, drawings (Fig. 2.2c).

The idea between distinguishing the two types of uncertainty is to characterize the uncertainty coming from the model and to take adequate actions to reduce it. For example, in active learning, selecting and labelling regions with large epistemic uncertainty should better improve the model's capacity to generalize, while focusing on large aleatoric uncertainty would be inefficient [39]. However, in some cases, such distinction may be unnecessary. For instance, when deploying a ML system for autonomous driving applications, the source of uncertainty might be inconsequential: the main purpose is to decide whether the agent should send a trigger alarm – or give back control to user – if the total uncertainty estimation regarding its prediction is high. This case is studied in Chapter 5. On a related matter, aleatoric and epistemic uncertainties are not absolute notions but are context-dependent. As

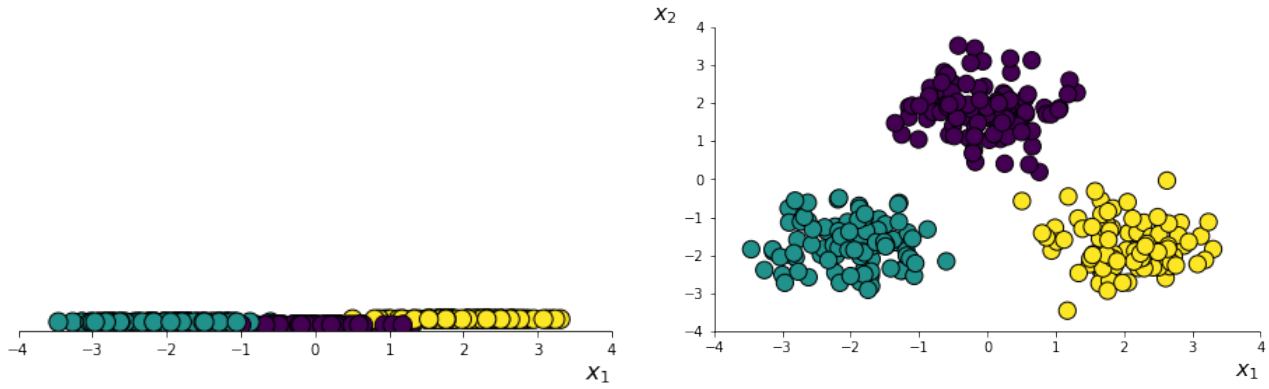


Figure 2.3: **Embedding data in a higher-dimensional space can reduce aleatoric uncertainty.** While the two classes are overlapping on the left plot, by adding a second feature $\boldsymbol{x}_2$ they become separable, and consequently aleatoric uncertainty is reduced [63].

illustrated by [63] and replicated in Fig. 2.3, embedding data in a higher-dimensional space can reduce aleatoric uncertainty but it may also increase epistemic uncertainty as more data are required to fit a model.

This is why uncertainty modelling in machine learning should come with a clear description of the setting of the learning problem.

2.1.2 Uncertainty in supervised learning

Let us consider a training dataset $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ composed of N *i.i.d.* training samples, where $\boldsymbol{x}_n \in \mathcal{X}$ is an input, deep feature maps from an image or the image itself for instance, and $y_n \in \mathcal{Y}$ its corresponding output. These samples are drawn from an unknown joint distribution $P(X, Y)$ over $(\mathcal{X}, \mathcal{Y})$. Given loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$, the goal of supervised learning is to find a *hypothesis* $h^* : \mathcal{X} \to \mathcal{Y}$ within a fixed class $\mathcal{H}$ of functions that minimizes the *true risk*:

$$h^* \in \arg\min_{h \in \mathcal{H}} \int_{\mathcal{X} \times \mathcal{Y}} \ell(h(\boldsymbol{x}), y) dP(\boldsymbol{x}, y). \tag{2.1}$$

Because the distribution $P(X, Y)$ is unknown to the learning algorithm, we restricted the goal to find the hypothesis $\hat{h}$ that minimizes the *empirical risk* given training data $\mathcal{D}$:

$$\hat{h} \in \arg\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{n=1}^{N} \ell(h(\boldsymbol{x}_n), y_n). \tag{2.2}$$

Hypotheses h^* and $\hat{h}$ are also known respectively as the *Bayes estimator* and the *empirical Bayes estimator* as they minimize the posterior expected value of loss ℓ. Obviously, $\hat{h}$ is only an estimate of h^* whose quality depends on the amount and diversity of training data. The uncertainty that arises due to this discrepancy is referred as *approximation uncertainty*.

Eventually, given an input $\boldsymbol{x}$, what we're interested in is to evaluate the *predictive uncertainty* $p(y|\boldsymbol{x})$, *i.e.* the uncertainty related to predicting an outcome. In the case of a stochastic dependency

between $\mathcal{X}$ and $\mathcal{Y}$, even with a perfect knowledge of P there still remain uncertainty, which is characterized as *aleatoric uncertainty*. Given an input $\boldsymbol{x}$ with true label y, the best prediction would be the *point-wise Bayes estimator* f^*:

$$f^*(\boldsymbol{x}) = \arg\min_{\hat{y}\in\mathcal{Y}} \int_{\mathcal{Y}} \ell(y, \hat{y}) dP(y|\boldsymbol{x}). \tag{2.3}$$

Due to the choice of the hypothesis space $\mathcal{H}$, the Bayes estimator h^* does not coincide with the point-wise Bayes estimator f^*, which give rise to *model uncertainty*. Approximation uncertainty and model uncertainty are related to model aspects, either due to model design or to training data. Hence, they can be grouped as epistemic uncertainty. In practice, epistemic uncertainty is reduced to approximation uncertainty as deep neural networks with non-linear activations can theoretically approximate any continuous function ('universal approximation theorem' [67]), thus $h^* \approx f^*$.

2.2 Modelling uncertainty with deep neural networks

Capturing both aleatoric and epistemic uncertainty is crucial to provide accurate uncertainty estimates. The Bayesian framework provides a natural probabilistic representation of uncertainty by incorporating degrees of belief. After an overview of recent developments of deep learning, we will see in this section how Bayesian approaches model uncertainty and which measures are used to quantify uncertainty in practice.

2.2.1 Deep neural networks

Inspired by the simplified modelling of a biological neuron [68], an artificial feed-forward neural network, simply shortened here as neural network (NN), is a non-linear function $f : \mathcal{X} \to \mathcal{Y}$ composed of a succession of non-linear mathematical functions, called *layers*, that progressively transforms an input $\boldsymbol{x}$ to an output y:

$$f(\boldsymbol{x}) = f^{(L)} \circ f^{(L-1)} \circ f^{(L)} \circ \cdots \circ f^{(1)}(\boldsymbol{x}), \tag{2.4}$$

where L is the number of layers. Each layer $l \in [\![1, L]\!]$ is parametrized by $\boldsymbol{\theta}_l$ and we denote the overall set of parameters of the neural network as $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_L)$.

A classic layer is the *fully-connected* layer which consists in a linear combination of the input followed by a nonlinear activation $\mathbf{h}_l = \phi(\mathbf{w}_l \mathbf{h}_{l-1} + \mathbf{b}_l)$ applied element-wise and where $\boldsymbol{\theta}_l = (\mathbf{w}_l, b_l)$. The typical nonlinearities are the sigmoid function, the hyperbolic tangent and the Rectified Linear Unit (ReLU), the latter being currently the most popular. A neural network composed of at least one hidden layer is called *multi-layer Perceptron* (MLP). Thanks to their depth, DNNs are able to transform raw input data into more and more complex representations, from the low-level concepts *e.g.*, colours or contours in computer vision, to high-level concepts such as objects, which is particularly useful for image classification. Interestingly, even with one sufficiently large hidden layer, MLPs can model any arbitrary function of the input thanks to the universal approximation theorem [67]. The last layer, also called the output layer, is followed by an activation reflecting the desired output.

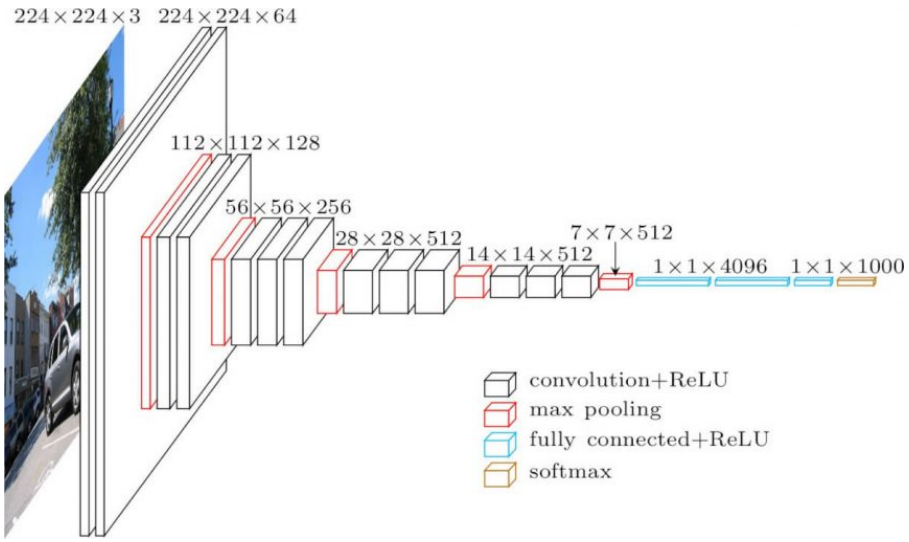


Figure 2.4: **VGG-16 [74] architecture** consists in a succession of convolutional layers and max-pooling layers followed by fully-connected layers for image classification. *Image credits: Durand [75].*

For instance, in multi-class classification where $\mathcal{Y} = [\![1, K]\!]$ with K being the number of classes, the softmax function is commonly used to output a vector of categorical probabilities[1]:

$$\forall k \in \mathcal{Y}, \quad P(Y = k|\, \boldsymbol{x}, \boldsymbol{\theta}) = \frac{\exp\left(f_k(\boldsymbol{x}, \theta)\right)}{\sum_{j \in \mathcal{Y}} \exp\left(f_j(\boldsymbol{x}, \theta)\right)}. \tag{2.5}$$

Training. Neural networks are trained using gradient descent optimizers, such as stochastic gradient descent (SGD) with momentum [69, 70], Adagrad [71], AdaDelta [72] and Adam [73]. The gradient of the loss with respect to the model's parameters $\boldsymbol{\theta}$ is obtained via back-propagation [69]. In classification tasks, a NN with softmax activation is commonly trained via *maximum likelihood estimation* on the training data, or equivalently minimizing the *negative log-likelihood*:

$$\hat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} -\log P(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}). \tag{2.6}$$

Convolutional neural networks. Deep learning became ubiquitous in computer vision in 2012 when AlexNet [4] won the ImageNet Large Scale Visual Recognition Challenge. AlexNet is an architecture that is part of a class of neural networks named *convolutional neural networks* (ConvNets). ConvNets are composed of a succession of convolutional and pooling layers, the former being a special case of matrix multiplication with circulant structure. By sharing a convolutional filter for all spatial positions, convolutional layers reduce the storage requirements of the model and encode translation equivariance. Pooling layers, or alternatively adding stride in a convolution, progressively aggregates spatial information as we go deeper in the network and produce invariance to small translations,

[1] In the following, we write $f(\boldsymbol{x}, \boldsymbol{\theta})$ to denote explicitly the dependence of f on its parameters $\boldsymbol{\theta}$

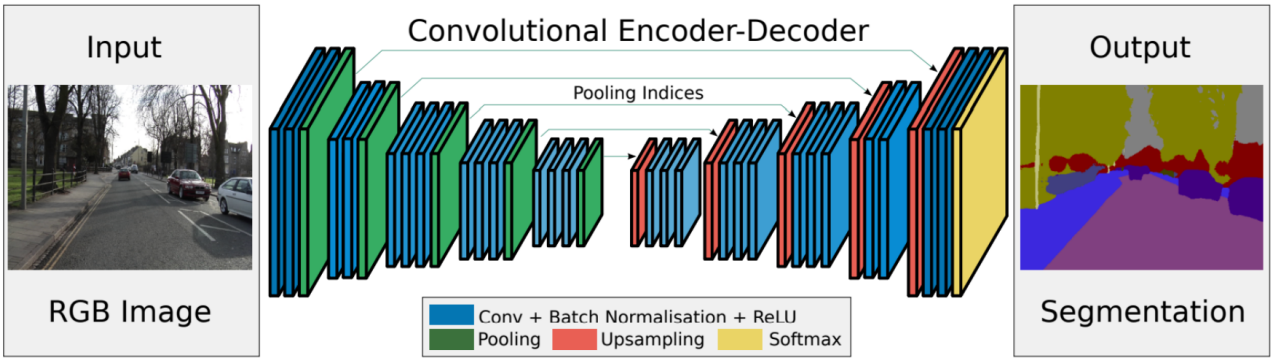


Figure 2.5: **SegNet [47] architecture** is a fully-convolutional neural network with the particularity that the decoder upsamples its input using the pool indices from its encoder. *Image credits: Badrinarayanan et al. [47].*

which is particularly relevant for computer vision as we often want the response of a classifier to be independent of the location of objects in the image. In Fig. 2.4, we show the architecture of a typical ConvNet, VGG-16 [74]. As deep neural networks became deeper and deeper, training issues due to vanishing gradient started to emerge: through a large number of layers, the loss gradient becomes smaller and smaller. ResNets [76] overcome this issue by adding skip connections between blocks of layers. Due to their strong performance on ImageNet, ResNets are now considered as a standard architecture for computer vision. In Chapter 5, we use one instance of ResNets with 18 layers, ResNet-18, which reaches 69.8% top-1 accuracy on ImageNet, compared to 56.5% with AlexNet and 71.6% with VGG-16 but with considerably fewer parameters (12M for ResNet-18 vs. 138M VGG-16).

Fully-convolutional neural networks. Semantic segmentation can be seen a pixel-wise classification problem. The desired output is a semantic map of the same size as the image input. To meet this challenge, *fully-convolutional neural networks* adopt an encoder-decoder structure where the encoder which reduce the spatial resolution and encodes a meaningful intermediate representation, then the decoder progressively recovers the spatial information by using successive upsampling operations. An example of fully-convolutional architecture used in Chapter 3 is shown in Fig. 2.5 with SegNet [47] which is based on VGG architecture. In Chapter 4, we also use DeepLab [7] which showed tremendous performance on various benchmarks for semantic segmentation. While new architectures now outperform DeepLab, this architecture became a standard, used in autonomous driving benchmarks such as Cityscapes [77].

2.2.2 Bayesian approaches

In Bayesian statistics, a probability expresses a degree of belief or information about an event. Given hypothesis h, we fix a prior distribution $p(h)$ over h and learning consists in updating that prior

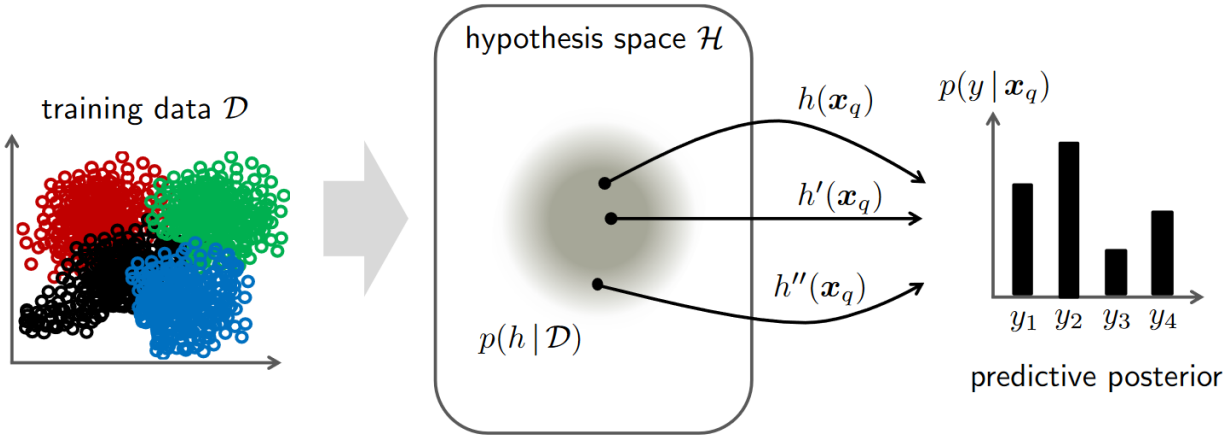


Figure 2.6: **Illustration of Bayesian inference**. Training data $\mathcal{D}$ is used to update the posterior distribution $p(h|\mathcal{D})$. Then, given an input $\boldsymbol{x}_q$, the posterior predictive distribution is obtained by Bayesian model averaging. *Image credits: Hüllermeier et al.* [63].

with the probability of the data given h, *i.e.* likelihood, according to Bayes' rule:

$$p(h|\mathcal{D}) = \frac{p(\mathcal{D}|h)p(h)}{p(\mathcal{D})} \propto p(\mathcal{D}|h)p(h). \tag{2.7}$$

The posterior distribution $p(h|\mathcal{D})$ captures the model's knowledge regarding hypothesis h given data $\mathcal{D}$. The more peaked this distribution is, the more certain the model will be in regards to epistemic uncertainty.

In Bayesian inference, given an unknown input $\boldsymbol{x}$, the posterior predictive distribution $p(y|\boldsymbol{x}, \mathcal{D})$ is obtained by *Bayesian model averaging*:

$$p(y|\boldsymbol{x}, \mathcal{D}) = \int_{\mathcal{H}} p(y|\boldsymbol{x}, h) dp(h|\mathcal{D}). \tag{2.8}$$

Hence, the posterior predictive distribution is a weighted average over its probabilities under all hypotheses in $\mathcal{H}$, weighted by the posterior probability $p(h|\mathcal{D})$ (see Fig. 2.6).

Bayesian neural networks. While traditional deep neural networks output a single point-wise prediction, Bayesian neural networks [78, 79] (BNNs) propose to apply Bayesian inference by considering distributions over a network's parameters $\boldsymbol{\theta}$ and learning the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$. The posterior predictive distribution $p(y|\boldsymbol{x}^*, \mathcal{D})$ is obtained by marginalizing over the parameters $\boldsymbol{\theta}$:

$$p(y|\boldsymbol{x}, \mathcal{D}) = \int p(y|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}. \tag{2.9}$$

When modelling complex real-world data, exact inference may be intractable because the previous integrals cannot be expressed in closed form, since the parameters are mapped through non-linearities in deep neural network architectures.

Since the posterior distribution cannot usually be evaluated analytically, a few approximation methods have been considered to compute the posterior predictive. A first simple approach consists in approximating the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ with a Dirac distribution centered on the maximum likelihood estimator $\hat{\boldsymbol{\theta}}_{\text{MLE}}$:

$$\boldsymbol{\theta}_{\text{MLE}} \in \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}) \tag{2.10}$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \log p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}). \tag{2.11}$$

Then, the posterior predictive distribution is simply the evaluation of the prediction on this point: $p(y|\boldsymbol{x}, \mathcal{D}) \approx p(y|\boldsymbol{x}, \hat{\boldsymbol{\theta}}_{\text{MLE}})$[2]. However, we obtain a point estimate for parameters $\boldsymbol{\theta}$ which may overfit. For instance, with a dataset composed of 3 tosses landed head, we would then estimate $\hat{\boldsymbol{\theta}}_{\text{MLE}}$ such that $p(y|\boldsymbol{x}^*, \mathcal{D}) = 1$ for any toss! To mitigate this issue, one could instead compute the *maximum-a-posteriori* estimate $\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D})$ but it still remains a point estimate that underestimates epistemic uncertainty.

With deep neural networks, a few methods have been explored including Laplace approximation [78], Hamiltonian Monte Carlo sampling [80], and expectation-propagation [81, 82]. In particular, variational inference [83, 84] gained a lot of popularity in the recent years due to better scaling. The goal is to learn to approximate the exact posterior distribution by defining a simpler variational distribution $q(\boldsymbol{\theta})$ and minimizing the Kullback-Leibler (KL) divergence between $q(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta}|\mathcal{D})$. For instance, *Variational Bayes* [85] defines the variational distribution $q(\boldsymbol{\theta})$ as a Gaussian distribution with a diagonal covariance, *i.e.* a fully factorized Gaussian. Another important example is *Monte-Carlo Dropout* (MC Dropout) where Gal and Ghahramani [40] establish a connection between variational inference and dropout layers [86], commonly used in neural networks for regularization. At inference, the predictive distribution is approximated by Monte Carlo sampling and averaging over all M forward predictions:

$$p(y|\boldsymbol{x}, \mathcal{D}) \approx \frac{1}{M} \sum_{m=1}^{M} p(y|\boldsymbol{x}, \boldsymbol{\theta}_m), \tag{2.12}$$

where $\boldsymbol{\theta}_m \sim p(\boldsymbol{\theta}|\mathcal{D})$ are the sampled weights from forward pass m. The total uncertainty can then be quantified in terms of variance in the case of regression and entropy as detailed in the following section.

With Bayesian Neural Networks, the crucial aspect is how well the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ is approximated [87]. Unfortunately, MCDropout has been shown to be a poor approximation to the true posterior [88], resulting in unreliable uncertainty estimates [54, 89, 90].

Ensembles. Lakshminarayanan *et al.* [53] propose a simple but effective approach named *Deep Ensembles* which outperforms Bayesian neural networks for uncertainty representation [54, 55]. An ensemble of M models is trained independently with random initialization. Such as with MC Dropout,

[2]Note that this method actually correspond to a standard neural network trained with maximum likelihood.

predictions are obtained by averaging the M samples and uncertainty estimates can be derived from the spread of the ensemble. While being originally considered as non-Bayesian, Deep Ensembles can actually be seen as a Bayesian model average [91], whose samples provides a richer functional diversity in the predictive integral Eq. (2.9). Further works [92, 93] explored ways to avoid training multiple models to reduce training time by leveraging intermediate checkpoints of a model during training. The main drawback of these approaches is the computational expense of training and storing weights of M models, which is not convenient for embedded systems such as autonomous vehicles.

Gaussian processes. Considered as the gold standard of uncertainty estimation [94], Gaussian processes are non-parametric Bayesian models. Unlike BNNs which define probability distributions over networks' weights, they directly specify distributions over the *function* $f(\cdot, \boldsymbol{\theta})$ induced by the network. This distribution is a joint Gaussian distribution defined over a collection of function values $f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_n)$. The computation of the covariance function (or kernel) of the distribution requires access to the full training dataset at inference time. Although some approximations [95] have been proposed, this family of probabilistic methods does not scale well with the dimension of the data.

Previous methods proposed adapting neural networks to capture epistemic uncertainty thanks to the spread of the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$. But how do we derive uncertainty estimates from these Bayesian approaches? This will be addresses in 2.2.4.

2.2.3 Evidential models

To overcome the issue of approximation due to sampling, a recent class of models, named *evidential* [57, 56] proposes instead to explicitly represent the distribution over probabilities. This line of work is based on subjective logic [96], a probabilistic framework which formalizes the Dempster-Shafer [97] theory's notion of belief as a Dirichlet distribution. In the multi-class setting, the subjective opinion of a multinomial random variable $y \in \mathcal{Y}$ is given by a triplet:

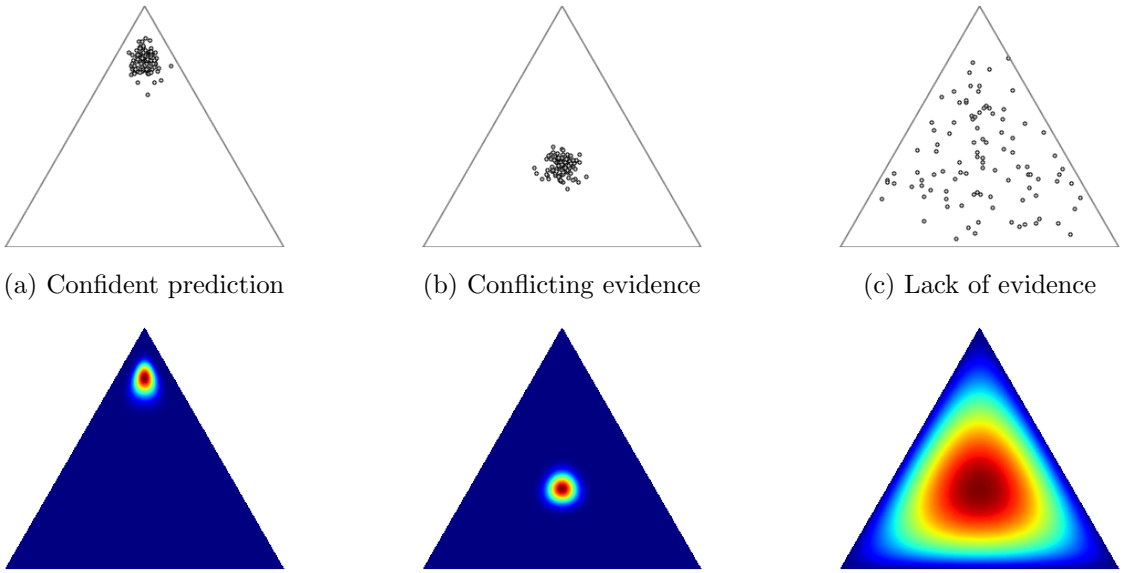
$$\omega = (\mathbf{b}, u, \mathbf{a}) \qquad \text{with} \quad \sum_{k \in \mathcal{Y}} b_k + u = 1, \tag{2.13}$$

where $\mathbf{b} = (b_1, \cdots, b_K)^T$ denotes the belief mass over $\mathcal{Y}$, $u \geq 0$ is the overall uncertainty mass and a is the base rate distribution. Let $e_k \geq 0$ be the evidence derived for class k. The class belief b_k and the uncertainty u are computed as:

$$b_k = \frac{e_k}{S} \qquad \text{and} \qquad u = \frac{K}{S}, \tag{2.14}$$

where $S = \sum_{k=1}^{K}(e_k + 1)$. Note that the uncertainty u is inversely proportional to the total evidence.

The link to the Dirichlet distribution can be grasped by first considering the simpler problem of inferring from a set $\mathcal{D}$ of N rolls the probability that a dice with K sides comes up as face k [98]. We denote $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_K)$ the random variable over categorical probabilities, where $\sum_{k=1}^{K} \pi_k = 1$, and which lives on the (K-1)-dimensional simplex $\triangle^{K-1}$. Assuming *i.i.d.* data, its likelihood reads $p(\mathcal{D}|\boldsymbol{\pi}) = \prod_{k=1}^{K} \pi_k^{N_k}$ where N_k is the count of class k among the N draws.



(a) Confident prediction (b) Conflicting evidence (c) Lack of evidence

Figure 2.7: **Uncertainty representation on the simplex**. Top row shows samples drawn from an ensemble or a BNN. Bottom row illustrates the implicit distribution they are sampled from. A confident prediction will have a distribution focused on a corner of the simplex and with a low dispersion (Fig. 2.7a). Conflicting evidence (aleatoric uncertainty) will result in a distribution close to the simplex center (Fig. 2.7b), reflecting a high class confusion. Finally, a lack of evidence (Fig. 2.7c) corresponds to a distribution with high dispersion (epistemic uncertainty): each sample can yield very different probabilities.

For its conjugate properties with the categorical distribution, the prior $p(\boldsymbol{\pi})$ can be modeled as a Dirichlet distribution with concentration parameters $\boldsymbol{\beta}$. Then, the posterior $p(\boldsymbol{\pi}|\mathcal{D})$ is also a Dirichlet distribution with parameters $(\beta_1 + N_1, ..., \beta_K + N_K)$ and the posterior predictive distribution for a single multinoulli trial has the closed form $P(Y = k \mid \mathcal{D}) = \mathbb{E}[\pi_k|\mathcal{D}] = \frac{\beta_k + N_k}{\sum_k \beta_k + N}$. We observe that the prior distribution acts as a *Bayesian smoothing* by adding pseudo-counts $\boldsymbol{\beta}$ to the empirical counts.

Let us extend the Bayesian treatment of a single categorical distribution to classification, *i.e.*, the goal is to predict the class label y from a categorical distribution that depends on input $\boldsymbol{x}$. The training dataset $\mathcal{D}$ consists of N *i.i.d.* samples $(\boldsymbol{x}, y)$ drawn from an unknown joint distribution $P(X, Y)$. Obviously, for a test sample $\boldsymbol{x}^*$, its label frequency count is now unknown and we are not able to estimate the posterior predictive distribution $P(Y|\boldsymbol{x}^*, \mathcal{D})$. Bayesian models and ensembling methods approximate the posterior predictive distribution by marginalizing over the network's parameters thanks to sampling. But this comes at the cost of multiple forward passes.

Evidential Neural Networks (ENNs) propose instead to model explicitly the posterior distribution over categorical probabilities $p(\boldsymbol{\pi}|\boldsymbol{x}, y)$ by a variational Dirichlet distribution,

$$q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{\theta})) = \frac{\Gamma(\alpha_0(\boldsymbol{x}, \boldsymbol{\theta}))}{\prod_{k=1}^{K} \Gamma(\alpha_k(\boldsymbol{x}, \boldsymbol{\theta}))} \prod_{k=1}^{K} \pi_k^{\alpha_k(\boldsymbol{x}, \boldsymbol{\theta}) - 1}, \tag{2.15}$$

whose concentration parameters $\boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{\theta}) = \exp f(\boldsymbol{x}, \boldsymbol{\theta})$ are output by a neural network f with parameters $\boldsymbol{\theta}$; Γ is the Gamma function and $\alpha_0(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{k=1}^{K} \alpha_k(\boldsymbol{x}, \boldsymbol{\theta})$ with $\alpha_k = \exp f_k(\boldsymbol{x}, \boldsymbol{\theta})$ indexing the k^{th} element of the vector of all K concentration parameters $\boldsymbol{\alpha}$. Precision α_0 controls the sharpness of the density with more mass concentrating around the mean as α_0 grows. By conjugate property, the predictive distribution for a new point $\boldsymbol{x}^*$ is

$$P(Y = k \mid \boldsymbol{x}^*, \mathcal{D}) \approx \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x}^*)}[\pi_k] = \frac{\exp f_k(\boldsymbol{x}^*, \boldsymbol{\theta})}{\sum_{j=1}^{K} \exp f_j(\boldsymbol{x}^*, \boldsymbol{\theta})}, \tag{2.16}$$

which is the usual output of a network f with softmax activation.

Instead of reasoning on first-order probabilities, we can now derive second-order uncertainty measures on the Dirichlet distribution. Evidential models provide a second-order uncertainty representation as shown in Fig. 2.7 where the expectation of the Dirichlet distribution relates to aleatoric uncertainty and its criterion concerning its dispersion can measure the the amount of evidence in a prediction, hence epistemic uncertainty

Training Objective The ENN training is formulated as a variational approximation to minimize the KL divergence between the distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x})$ and the true posterior distribution $p(\boldsymbol{\pi}|\boldsymbol{x}, y)$:

$$\mathcal{L}_{\text{var}}(\boldsymbol{\theta}; \mathcal{D}) = \mathbb{E}_{(\boldsymbol{x},y)\sim P(X,Y)}\big[\mathbb{KL}\big(q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x}) \parallel p(\boldsymbol{\pi}|\boldsymbol{x}, y)\big)\big] \tag{2.17}$$

The training objective and its derivation are further study in Chapter 5.

2.2.4 Uncertainty measures

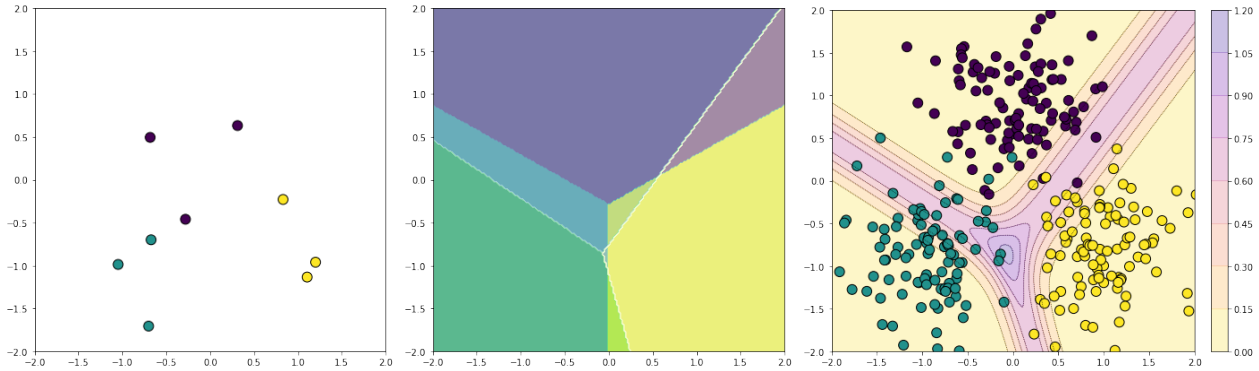
In regression, while the predictive distribution $p(y|\boldsymbol{x}, \mathcal{D})$ remains intractable (Eq. (2.8)), likelihood is assumed Gaussian and one can estimate the predictive distribution's first two moments empirically [40]. Given likelihood $p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y; f(\boldsymbol{x}, \boldsymbol{\theta}), \tau^{-1}\boldsymbol{I})$, the first moment $\mathbb{E}_{p(y|\boldsymbol{x},\mathcal{D})}[y]$ can be approximated by the unbiased estimator $\tilde{\mathbb{E}}[y] = \frac{1}{M}\sum_{m=1}^{M} f(\boldsymbol{x}, \boldsymbol{\theta}_m)$ following Monte-Carlo sampling. The model's predictive variance $\text{Var}_{p(y|\boldsymbol{x},\mathcal{D})}[y]$ – the second moment – is given by the unbiased estimator $\tilde{\text{Var}}[y] = \tau^{-1}\boldsymbol{I} + \frac{1}{M}\sum_{m=1}^{M} f(\boldsymbol{x}, \boldsymbol{\theta}_m)^T f(\boldsymbol{x}, \boldsymbol{\theta}_m) - \tilde{\mathbb{E}}[y]^T\tilde{\mathbb{E}}[y]$. In particular, we note that the predictive variance accounts both for the aleatoric uncertainty with $\tau^{-1}\boldsymbol{I}$ and for the epistemic uncertainty with the second term.

When it comes to classification, the aleatoric uncertainty at an input point $\boldsymbol{x}$ is defined as the entropy of the *true* conditional distribution $p(Y|\boldsymbol{x}, \mathcal{D})$:

$$\mathbb{H}[Y|\boldsymbol{x}, \mathcal{D}] = -\sum_{k\in\mathcal{Y}} p(Y = k|\boldsymbol{x}, \mathcal{D}) \log p(Y = k|\boldsymbol{x}, \mathcal{D}). \tag{2.18}$$

The entropy attains its maximum value when all classes have equal uniform probability and its minimum value of zero when one class has probability 1 and all others probability 0. But in contrast with regression, we cannot rely on the previous derivation to estimate the predictive moments: likelihood is now a categorical distribution and we cannot estimate its first two moments anymore:

$$p(Y|\boldsymbol{x}, \boldsymbol{\theta}) = \text{Cat}\Big(Y; \phi(f(\boldsymbol{x}, \boldsymbol{\theta}))\Big), \tag{2.19}$$



(a) Train data (b) Bayes optimal's vs. model's decision frontiers (c) Entropy map

Figure 2.8: **Illustration of unreliable uncertainty estimates with MCP and entropy due to poor fitting of the conditional distribution.** A logistic regression classifier trained on only nine inputs sampled from the distribution (Fig. 2.8a) will have its decision frontier differ greatly from the Bayes optimal classifier h^* given the marginal distribution of the Gaussian mixture (Fig. 2.8b). Consequently, uncertainty measures estimated on the predictive distribution such as entropy (Fig. 2.8c) poorly reflect the true aleatoric uncertainty of the conditional distribution.

where the softmax operator, $\phi : \mathbb{R}^K \to \triangle^{K-1}$, transforms logits into probabilities on the (K-1)-dimensional unit simplex $\triangle^{K-1}$, thanks to an exponential form.

A first possibility is to use the entropy of the *predictive* distribution estimated by the model. In the case of Monte Carlo sampling, this corresponds to averaging the probability vectors from the M stochastic forward passes:

$$\mathbb{H}[Y|\boldsymbol{x}, \mathcal{D}] = -\sum_{k \in \mathcal{Y}} p(Y = k|\boldsymbol{x}, \mathcal{D}) \log p(Y = k|\boldsymbol{x}, \mathcal{D}) \tag{2.20}$$

$$\approx -\sum_{k \in \mathcal{Y}} \Big(\frac{1}{M} \sum_{m=1}^{M} p(Y = k|\boldsymbol{x}, \boldsymbol{\theta}_m) \Big) \log \Big(\frac{1}{M} \sum_{m=1}^{M} p(Y = k|\boldsymbol{x}, \boldsymbol{\theta}_m) \Big) \tag{2.21}$$

with $\boldsymbol{\theta}_m \sim p(\boldsymbol{\theta}|\mathcal{D})$ are the sampled weights from forward pass m.

Given an input $\boldsymbol{x}$, it is also possible to estimate aleatoric uncertainty by looking at the likelihood of the class predicted by the model m, which is by design the class associated with the *maximum* probability:

$$MCP_m(\boldsymbol{x}) = \max_{k \in \mathcal{Y}} p(Y = k|\boldsymbol{x}, \boldsymbol{\theta}_m). \tag{2.22}$$

In the case of Monte Carlo sampling, MCP is computed on the average probability vector. MCP values range from $1/K$ to its maximum value of one when all probabilities are 0 except for the predicted class, hence no uncertainty.

But the quality of uncertainty estimates given by MCP and predictive entropy depends on the quality of the approximation of the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ and consequently may be inaccurate. For instance, Fig. 2.8 shows an uncertainty map computed from the predictive entropy output by

a neural network trained only on nine samples on a toy dataset. This toy dataset is composed of a Gaussian mixture with three equally weighted components having equidistant centers and equal spherical covariance matrices. As the model's decision frontiers do not coincide with the Bayes optimal ones – given by the true conditional distribution $p(y|\boldsymbol{x})$ –, the predictive entropy might be incorrectly low in regions of high aleatoric uncertainty (close to Bayes optimal's decision frontiers) and high in confident regions.

To estimate the epistemic uncertainty, an intuitive idea with BNNs and ensembling is to consider the variance of the predictions produced by the M stochastic forward passes. Gal [40] proposes to compute the *variation-ratio* which is based on the frequency of prediction of the most predicted class:

$$\text{var-ratio}(\boldsymbol{x}) = 1 - \max_{k \in \mathcal{Y}} \Big(\frac{1}{M} \sum_{m=1}^{M} \mathbb{1}[\hat{y}_m = k] \Big). \tag{2.23}$$

More interestingly, Depeweg *et al.* [99] propose to measure the *mutual information* $\mathbb{I}[y, \boldsymbol{\theta}|\boldsymbol{x}, \mathcal{D}]$ between the prediction y and the posterior distribution, based on the decomposition of the predictive uncertainty. Assuming that predictive entropy $\mathbb{H}[y|\boldsymbol{x}, \mathcal{D}]$ contains aleatoric and epistemic uncertainty as it depends on dataset $\mathcal{D}$, the mutual information corresponds to the difference between predictive entropy and the expected entropy of each member of the ensemble $\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\Big[\mathbb{H}[y|\boldsymbol{x}, \boldsymbol{\theta}]\Big]$, which does not depend on the model anymore:

$$\begin{aligned} \mathbb{I}[Y, \boldsymbol{\theta}|\boldsymbol{x}, \mathcal{D}] &= \mathbb{H}\Big[\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\big[p(Y|\boldsymbol{x}, \boldsymbol{\theta})\big]\Big] - \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\Big[\mathbb{H}[Y|\boldsymbol{x}, \boldsymbol{\theta}]\Big] \qquad (2.24) \\ &\approx -\sum_{k \in \mathcal{Y}} \Big(\frac{1}{M} \sum_{m=1}^{M} p(Y = k|\boldsymbol{x}, \boldsymbol{\theta}_m) \Big) \log \Big(\frac{1}{M} \sum_{m=1}^{M} p(Y = k|\boldsymbol{x}, \boldsymbol{\theta}_m) \Big) \\ &\quad + \frac{1}{M} \sum_{m=1}^{M} \sum_{k \in \mathcal{Y}} p(Y = k|\boldsymbol{x}, \boldsymbol{\theta}_m) \log p(Y = k|\boldsymbol{x}, \boldsymbol{\theta}_m). \qquad (2.25) \end{aligned}$$

Consequently, mutual information is a dispersion measure which accounts for the variance of the predictions produced by the M stochastic forward passes.

To gain intuition about the behavior of the previous measures, let us consider a classification task with 3 classes and the following samples:

1. a sample where the model outputs probability vectors with maximum probability on the same class:

$$p_1 = \{(1, 0, 0), (1, 0, 0), ..., (1, 0, 0)\}$$

;

2. a sample where the model outputs probability vectors with uniform probability:

$$p_2 = \{(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}), (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}), ..., (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})\}$$

;

3. a sample where the model produces inconsistent probability vectors:

$$p_3 = \{(1, 0, 0), (0, 1, 0), ..., (0, 0, 1)\}$$

.

This first sample p_1 represents an input predicted with high confidence by the model and where the aleatoric uncertainty is low. The second sample p_2 presents high aleatoric uncertainty due to class confusion but low epistemic uncertainty as the model always predicted the same probability vector. In contrast, the model outputs very different predictions regarding the third sample p_3, denoting a large epistemic uncertainty. When computing the predictive entropy, we find that obviously $\mathbb{H}[p_1] = 0$ but $\mathbb{H}[p_2] = \mathbb{H}[p_3] = 1.09$, which does not enable us to separate the two sources of uncertainty. Now, the mutual information gives us more information about the third sample as $\mathbb{I}[p_1] = \mathbb{I}[p_3] = 0$ and $\mathbb{I}[p_2] = 1.09$, showing here it measures solely the dispersion between predictions. Again here, this theoretical decomposition depends on the approximation to this posterior distribution which may re-introduce approximation uncertainty in both terms.

Finally, with evidential models, a series of uncertainty measures based on the second-order Dirichlet distribution allows one to measure different sources of uncertainty [100, 101]. In particular, the *vacuity* is due to insufficient or unreliable information received from sources and represented by uncertainty mass u in subjective logic. On the second-order Dirichlet distribution, this is equivalent to its precision α_0 which is a measure of its dispersion, hence capturing epistemic uncertainty. Related to aleatoric uncertainty, *dissonance* corresponds to contradicting belief, such as in class confusion, and is defined as:

$$diss(\omega) = \sum_{k \in \mathcal{Y}} \Big(\frac{b_k \sum_{j \neq k} b_j \text{Bal}(b_j, b_k)}{\sum_{j \neq k} b_j} \Big), \tag{2.26}$$

where $\text{Bal}(b_j, b_k) = 1 - \frac{|b_j - b_k|}{b_j + b_k}$ if $b_k b_j \neq 0$ and 0 if $\min(b_j, b_k) = 0$ is the relative mass balance function between two belief masses. For instance, given opinion $(b_1, b_2, b_3, u, \mathbf{a}) = (0.3, 0.3, 0.3, 0.1, \mathbf{a})$, the dissonance value is equal to 0.9. Its maximum value is 1 and its minimum value is 0.

2.3 Evaluation of the quality of uncertainty estimates

Evaluating the quality of predictive uncertainties is challenging as the 'ground truth' uncertainty estimates are usually not available. Depending on the application, the desirable properties of uncertainty estimates can vary: in a multi-modal system, we aim for calibrated uncertainty estimates before fusion while one may only be interested in a reliable ranking between correct and erroneous predictions.

We present in this section the existing tasks commonly used in the literature to evaluate the quality of uncertainty estimates with deep neural networks.

2.3.1 Selective classification

The idea of a reject option with ML systems has been around for ages [41]. Classification with a reject option, also known as *selective classification* [44], consists in a scenario where a classifier can abstain on points where its confidence is below a certain threshold. By abstaining from predicting when in doubt, the main motivation is to reduce the error rate while keeping as many correct samples as possible.

To select which sample to reject, a *confidence-rate function* κ_f is associated to the classifier f in order to evaluate the degree of confidence of its predictions, the higher the value the more certain the prediction. Uncertainty estimates are used here to assess this degree of confidence. Then, given a threshold δ, an input $\boldsymbol{x}$ is rejected if its degree of confidence is lower than the threshold value,

$$g(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \kappa_f(\boldsymbol{x}) \geq \delta\,, \\ 0 & \text{otherwise.} \end{cases} \tag{2.27}$$

Ideally, uncertainty estimates should enable the selection function to split the test set in a subset containing all errors and the other set containing all correct predictions.

The performance of a selective model is quantified using coverage and risk. Re-using the notations introduced in Section 2.1.2, we also consider explicitly a test set $\mathcal{D}_{\text{test}}$ composed of labeled samples also following $P(X, Y)$. Coverage is defined to be the probability mass of the non-rejected region in $\mathcal{X}$, which can be approximated empirically by the number of non-rejected samples:

$$\hat{\phi}(g) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_{\text{test}}} g(\boldsymbol{x}), \tag{2.28}$$

where $|\mathcal{D}_{\text{test}}|$ is the number of samples in the test set. The selective risk corresponds to the evaluation of the loss ℓ on the non-rejected samples, which is commonly the 0/1 error with classification, divided by coverage. Its empirical approximation writes as:

$$\hat{R}(f, g) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_{\text{test}}} \frac{\ell(f(\boldsymbol{x}), y)g(\boldsymbol{x})}{\hat{\phi}(g)}. \tag{2.29}$$

Given test set $\mathcal{D}_{\text{test}}$, the task evaluation is based on risk-coverage curves such as shown in Section 3.5.2. These curves are obtained by computing the empirical selective risk for various values of coverage. The threshold δ depends on a user-specified cost for abstention. Consequently, to free ourselves from choosing this threshold, we compare methods by computing the following metrics:

- **AURC** measures the Area Under the Risk-Coverage curve. This metric is threshold-independent. The higher the AURC, the better the selective classifier.
- **Excess-AURC (E-AURC)**. This is a normalized AURC metrics defined in [102]. It takes into account the optimal ranking given the error rate of the classifier. More specifically, if we denote κ_f^* the perfect confidence-rate function and $\hat{r}$ the risk of classifier f, it writes as:

$$\text{E-AURC}(\kappa_f) = \text{AURC}(\kappa_f) - \text{AURC}(\kappa_f^*) \tag{2.30}$$

$$\approx \text{AURC}(\kappa_f) - (\hat{R} + (1 - \hat{R})\log(1 - \hat{R})). \tag{2.31}$$

With deep neural networks, we denote two types of approaches. The first one considers a trained prediction model and constructs a selection mechanism [103]. Most of the time, the confidence-rate function used is the value of MCP given by the softmax layer output. The second type of approaches aims to jointly learn the classifier and the selection function [104]. In particular, Geifman & El-Yaniv [105] train a DNN to optimize classification and rejection simultaneously. The reject function corresponds to the output of a second head of the DNN.

2.3.2 Misclassification detection

Given a trained model, *misclassification detection*, also referred as failure prediction [106], is the task of predicting at run-time whether the model has taken a correct decision or not for a given input. Uncertainty estimates are used here as confidence score to compare to a threshold δ. We say that the input $\boldsymbol{x}$ is estimated to be a correct prediction if its confidence score is above the threshold and to be an error, otherwise. Consequently, misclassification detection boils down to a binary classification task where instead of rejecting samples, we assign them a binary label:

$$g(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \kappa_f(\boldsymbol{x}) \geq \delta\,, \\ 0 & \text{otherwise.} \end{cases} \tag{2.32}$$

Such as for selective classification, the choice of the threshold impacts misclassification detection. Given a threshold δ, the test set can be split into true positives (TP), false positives (FP), false negative (FN) and true negatives (TN). From this confusion matrix, a common evaluation metrics is to choose a threshold such that the True Positive Rate (TPR) is equal to 95% and then evaluate the False Positive Rate (FPR):

$$FPR = \frac{FP}{FP + TN} \qquad \text{with} \quad TPR = \frac{TP}{TP + FN} = 95\%. \tag{2.33}$$

Threshold-independent evaluation metrics include the Area Under the ROC curve (AUROC), where the latter is a graphical plot showing the TPR and FPR against each other. It illustrates the ability of a binary classifier as its discrimination threshold is varied and it can be interpreted as the probability that a positive example has a greater detector score/value than a negative example. However, AUROC may suffer from unbalanced dataset, for instance when there is a larger amount of good predictions than wrong ones. In that case, AUROC will be close to 100% and the impact of wrongly ranking a misclassification is mitigated [45].

Alternatively, the Area Under the Precision-Recall curve (AUPR) is based on the graph between precision and recall:

$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{and} \qquad \text{recall} = \frac{TP}{TP + FN}. \tag{2.34}$$

AUPR is a metric that adjusts for different positive and negative base rates. As such, there is AUPR-Success where good predictions are considered positive and AUPR-Error where misclassification are now the positive class. In the second case, confidence scores are multiplied by -1.

As we will see in Chapter 3, a widely-used baseline method with deep neural networks [45] is to take the value of MCP given by the softmax layer output. A detailed review of proposed methods is presented in Section 3.4.

While these evaluation metrics can be used to assess the misclassification detection performance of a model, they cannot be used to directly compare performance across different models [55]. Correct and incorrect predictions are specific for every model, therefore, every model induces its own binary classification problem. The induced problems can differ significantly, since different models produce different confidences and misclassify different objects.

2.3.3 Calibration

In a number of applications of machine learning, it is of increasing importance to know whether the classifier output can be interpreted as actual probabilities. For instance, self-driving car with a multi-modal prediction system needs its individual components to provide comparable probabilities. Alternatively, in medical diagnosis, a ML system could request an additional analysis from human doctors if its output probability of a disease diagnosis is too low [107].

Given the probability distribution $\hat{p}(\boldsymbol{x}) = p(Y|\boldsymbol{x}, \boldsymbol{\theta})$ output by the model for a sample $\boldsymbol{x}$, a probabilistic classifier is calibrated if any predicted class probability is equal to the true class probability according to the underlying data distribution [108]:

$$\forall \boldsymbol{x} \in \mathcal{X}, \quad \mathbb{P}[Y \mid \hat{p}(\boldsymbol{x})] = \hat{p}(\boldsymbol{x}). \tag{2.35}$$

Any deviation from the perfect calibration is called miscalibration. A weaker condition [49] is to consider only the probability, or confidence estimate κ_f, associated with the predicted class $\hat{y}$:

$$\forall \boldsymbol{x} \in \mathcal{X}, \quad \mathbb{P}[Y = \hat{y} \mid \kappa_f(\boldsymbol{x})] = \kappa_f(\boldsymbol{x}). \tag{2.36}$$

For instance, given 100 predictions with a confidence $\kappa_f(\boldsymbol{x}) = 0.7$, we expect that 70 samples should be correctly classified if the classifier is perfectly calibrated according to Eq. (2.36).

Expected calibration error (ECE) [109] is a metric that estimates model miscalibration by splitting the probability scores into M bins B_m and comparing them to average accuracies inside these bins:

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{B_m}{N} \mid \mathrm{acc}(B_m) - \mathrm{conf}(B_m) \mid, \tag{2.37}$$

where

$$\mathrm{acc}(B_m) = \frac{1}{|B_m|} \sum_{\boldsymbol{x} \in B_m} \delta(\hat{y}(\boldsymbol{x}) - y(\boldsymbol{x})) \qquad \text{and} \qquad \mathrm{conf}(B_m) = \frac{1}{|B_m|} \sum_{\boldsymbol{x} \in B} \kappa_f(\boldsymbol{x}).$$

But ECE metric suffers from certain shortcomings. Due to binning, it does not monotonically increase as predictions approach ground truth (a biased estimator of the true calibration). Then, it only estimates miscalibration in terms of the maximum probability and does not evaluate the first condition in Eq. (2.35). Worse, a model may attain a perfect ECE score while being not accurate. For instance, on a binary classification, a model always predicting the first class $y = 1$ with confidence $\kappa_f(x) = 0.3$

may be perfectly calibrated on a dataset with 70% inputs of class 0 and 30% inputs of class 1, although its accuracy is only 0.3.

Alternatively, Lakshminarayanan *et al.* [53] argues that models should be trained and evaluated using a proper scoring rule to achieve good calibration. For instance, the *Brier score* measures the squared error between the predictive probability of a label and one-hot encoding of the correct label:

$$\text{BS} = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_{\text{test}}} \Big(\frac{1}{K} \sum_{k\in\mathcal{Y}} \big[p(Y = k \mid \boldsymbol{x}, \boldsymbol{\theta}) - \delta(y-k) \big]^2 \Big). \tag{2.38}$$

Finally, a popular metric for measuring the quality of in-distribution uncertainty is to measure the negative log-likelihood (NLL):

$$\text{NLL} = -\frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_{\text{test}}} \log p(y|\boldsymbol{x}, \mathcal{D}). \tag{2.39}$$

In classification, NLL boils down to computing the cross-entropy loss on the test set. It directly penalizes high probability scores assigned to incorrect labels and low probability scores assigned to the correct labels.

Recent work [49] revealed that deep neural networks are poorly calibrated. Among recalibration methods, a popular approach is to apply temperature scaling on models' logit [49]. The temperature parameter T is learned on a validation dataset $\mathcal{D}_{\text{val}}$ by minimizing the negative log-likelihood and keeping model's weight fixed:

$$\min_{T\in\mathbb{R}^+} \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_{\text{val}}} \log \frac{\exp\big(f_y(\boldsymbol{x},\boldsymbol{\theta})/T\big)}{\sum_{k\in\mathcal{Y}} \exp\big(f_k(\boldsymbol{x},\boldsymbol{\theta})/T\big)}. \tag{2.40}$$

Even though temperature scaling improves calibration, it does not affect the ranking of the confidence score between inputs. Consequently, temperature scaling is not effective to improve misclassification detection or selective classification mentioned in the previous sections.

2.3.4 Out-of-distribution detection

Until now, we reviewed tasks that evaluate uncertainty estimation on test samples assumed to be drawn *i.i.d.* from the same distribution than the training data. But as motivated in Chapter 1, in many safety-critical applications, ML systems are deployed in an open-world scenario [110]. Inputs can be subject to distributional shifts which are categorized either as covariate shifts maintaining semantic consistency, such as a drawing of a dog when training data was only composed of natural images, or semantic shifts where the label space $\mathcal{Y}$ is different from in-distribution data, such as input from a new class.

In the ML literature, several fields attempt to address the issue of identifying the unknowns/outliers/anomalies samples in the open-world setting. In their survey, Yang *et al.* [111] provide an interesting unified framework of these subtopics, summarized in Fig. 2.9 reproduced from their paper. In classification tasks, we can define five subcategories depending on the problem setting:

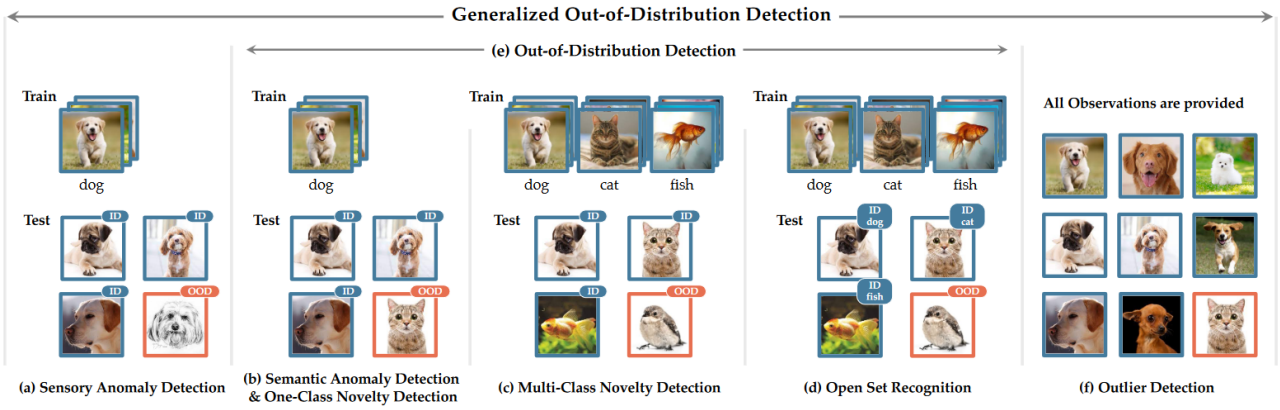


Figure 2.9: **Reproduction of the out-of-distribution framework proposed in [111]**. Five detection tasks are represented depending on their problem setting. Semantic anomaly detection, multi-class novelty detection and open-set recognition are considered as subcategories of out-of-distribution detection.

1. **Sensory anomaly detection**: training data is composed of only one class and test data may present non-semantic covariate shift, the goal is to detect these anomalies;
2. **Semantic anomaly detection**: training data is still composed of only one class and test data may now present semantic shift, such as new classes, the goal is to detect these anomalies;
3. **Multi-class novelty detection**: training data is composed of C classes and test data may present semantic shift, such as new classes, the goal is still to detect these anomalies;
4. **Open-set recognition**: training data is composed of C classes and test data may present semantic shift, such as new classes, but now the goal is twofold: correctly classify in-distribution data while detecting these anomalies;
5. **Outlier detection**: a transductive problem where all observations are provided, we do not consider a train/test split anymore, and some samples can present any distributional shift, the goal is still to detect these outliers, for instance to clean data.

Among these previous tasks, we commonly refer as *out-of-distribution detection* the sensory/semantic anomaly detection and multi-class novelty detection.

Out-of-distribution (OOD) detection shares similarities with misclassification detection as they both aim to detect errors or abnormal samples in a given test set. AUROC and AUPR metrics where the positive class is composed of OOD samples are used to evaluate the capacity of a model independently of a specific threshold to separate OOD samples from in-distribution samples according to a confidence score.

With deep neural networks, post-processing logits with temperature scaling using a large temperature T on a pre-trained model has been shown to be effective to reduce model's over-confidence on OOD samples [112]. At test time, the authors use the MCP as confidence score to detect OOD samples after applying the temperature scaling. Also in the family of post-processing methods, Lee

et al. [113] assumed that intermediate feature maps – in particular the penultimate before last classification layer – of a trained deep neural network follow class-conditional Gaussian distributions with a tied covariance matrix. They estimate its parameters on training data:

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{n:y_n=k}^{N} f(\boldsymbol{x}_n, \boldsymbol{\theta}) \quad \text{and} \quad \hat{\Sigma} = \frac{1}{N} \sum_{k \in \mathcal{Y}} \sum_{n:y_n=k}^{N} (f(\boldsymbol{x}_n, \boldsymbol{\theta}) - \hat{\mu}_k)(f(\boldsymbol{x}_n, \boldsymbol{\theta}) - \hat{\mu}_k)^T, \tag{2.41}$$

where N_k is the number of training samples with label k. Their confidence score correspond to the maximum Mahalanobis distance between input $\boldsymbol{x}$ and the closest class-conditional Gaussian distribution:

$$M(\boldsymbol{x}) = \max_{k \in \mathcal{Y}} -(f(\boldsymbol{x}, \boldsymbol{\theta}) - \hat{\mu}_k)^T \hat{\Sigma}^{-1} (f(\boldsymbol{x}, \boldsymbol{\theta}) - \hat{\mu}_k). \tag{2.42}$$

These two previous methods also used adversarial perturbations to improve the separability of OOD samples from in-distribution data. In contrast to the original literature on adversarial perturbation, they use fast gradient sign method [114] (FGSM) to increase the probability of the model on the predicted class (see Section 2.3.5). A limitation of these methods is that they need relevant OOD samples to find the right hyper-parameters T and ε.

On the other hand, a range of methods assumed that a set of OOD samples may be available during training. For instance, Hendrycks *et al.* [115] proposed to train a deep neural network to simultaneously classify in-distribution samples and to produce high predictive entropy for samples from a known large out-of-distribution dataset $\mathcal{D}_{out}$:

$$\mathcal{L}_{\text{OE}}(\boldsymbol{\theta}, \mathcal{D}, \mathcal{D}_{out}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\big[\log p(y|\boldsymbol{x}, \boldsymbol{\theta})\big] + \lambda \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{out}}\big[\mathbb{H}[p(\cdot|\boldsymbol{x}, \boldsymbol{\theta})]\big]. \tag{2.43}$$

While this method remains the best OOD detector so far, the assumption of available OOD data during training may be unrealistic in many applications [116, 117]. In addition, we show in Chapter 5 that all these previous methods are brittle to the choice of the OOD dataset.

Along with Mahalanobis-based OOD detection, a set of density-based methods relying on generative models attempt to model in-distribution data and to detect anomalous test data assuming OOD samples have low likelihood. In the context of deep learning, a classic method is to use an auto-encoder (AE) or a variational auto-encoder [118] (VAE) as generative model. However, Nalisnick *et al.* [119] find that the density learned by flow-based models [120], VAEs [121] and PixelCNNs [122] may assign a larger likelihood to OOD samples than in-distribution samples in some vision benchmarks (CIFAR-10 vs. SVHN, FashionMNIST vs MNIST, CelebA vs. SVHN, ImageNet vs. CIFAR-10). Finally, recent works [123, 124] explored using energy-based models (EBMs) for OOD detection, due to their natural fit within a discriminative framework. EBMs are generative models that use a scalar energy score to express probability density through unnormalized negative log probability [125]. But their learning process can be computationally unstable as they requires approximations, such as stochastic gradient Langevin dynamics [126] to estimate integrals.

2.3.5 Adversarial robustness

In contrast with anomalies, *adversarial examples* are inputs which are indistinguishable to the human eye but confuse a neural network, resulting in a misclassification (see Fig. 2.10). Adversarial

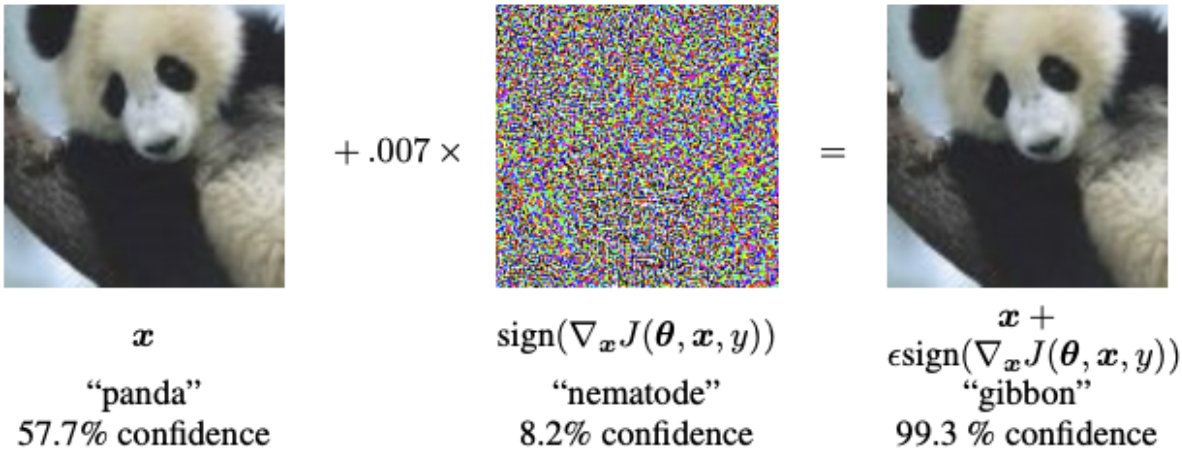


Figure 2.10: **Example of an adversarial attack with FGSM in classification**. An original input correctly classified as a panda becomes a misclassified input when applying the adversarial perturbation. Worse, its confidence score, here MCP, is arbitrarily high, wrongly indicating a confident prediction. *Image credits: Goodfellow et al. [114].*

examples are crafted by applying small perturbations to the input and restricting the magnitude of the attack to a value inferior to a bit of an 8-bit image encoding. In the original paper, the adversarial attack used rely on the fast gradient sign method [114] (FGSM):

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \varepsilon \cdot \text{sign} \nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{\theta}) \tag{2.44}$$

Since, a profusion of different adversarial attacks has been proposed in the literature [127, 128, 129, 130].

In adversarial robustness, the goal of an adversarial defense mechanism is to improve robustness of deep neural networks against adversarial attacks, *i.e.* reduce the gap between 'clean' accuracy on original inputs and 'adversarial' accuracy on adversarial examples. Multiple defense mechanisms have been proposed over the years but they almost all end up being defeated by new adversarial attacks, except for adversarial training [131] which remains correct under certain conditions. Consequently, recent advances in adversarial robustness tend to construct certified defenses where neural networks are provably robust against adversaries [132].

Recently, Tsipras *et al.* [133] showed the goal of adversarial robustness might fundamentally be at odds with that of standard generalization. For instance, adversarial training improves adversarial accuracy but also produce a slight decrease in original test accuracy. Instead of considering robustness to adversarial attacks, a different line of work [134, 135] investigates detection of these adversarial attacks. As with misclassification detection and OOD detection, the evaluation metric used are threshold-independent metrics such AUROC and AUPR.

2.4 Conclusion

Uncertainty estimation is a wide research area, ranging from theoretical perspectives with Bayesian approaches to practical considerations with the detection of abnormal samples to avoid critical failures. Uncertainty can arise due to the stochasticity of the latent data generative process (*aleatoric uncertainty*) or due to the lack of knowledge of the model on an input (*epistemic uncertainty*). While 'ground-truth' uncertainty estimates are usually not available, different tasks aims at evaluating the capacity of the model to provide accurate uncertainty estimates. Selective classification, misclassification detection and calibration evaluate in-distribution uncertainty, either for rejecting/detecting errors or to ensure a classifier which outputs correct probabilities. Out-of-distribution detection considers an open-world setting where distribution shifts and inputs from unknown classes may occur. Finally, adversarial robustness is a particular task where inputs have been corrupted to fool the classifier. In particular, one may see these adversarial examples as a worst-case analysis of distribution shift [136].

In this thesis, we will start by addressing in-distribution uncertainty estimation by proposing a learning confidence approach with auxiliary model to improve misclassification and selective classification in Chapter 3. The task of selective classification is also useful for self-training methods presented in Chapter 4. Finally, we tackle the challenge of jointly quantifying in-distribution and out-of-distribution (OOD) uncertainties in Chapter 5 with an uncertainty measure which account both for aleatoric and epistemic uncertainty.

Chapter 3

Learning A Model's Confidence via An Auxiliary Model

Chapter Abstract

Reliably quantifying the confidence of deep neural classifiers is a challenging yet fundamental requirement for deploying ML models in safety-critical applications. In this chapter, we are interested in the problem of detecting in-distribution erroneous predictions of deep neural networks in the context of classification. We introduce a novel target criterion for a model's confidence, namely the True Class Probability *(TCP) and show that TCP offers better properties for failure prediction than standard uncertainty measures. Since the true class is by essence unknown at test time, we propose to learn the TCP criterion from data with an auxiliary model,* ConfidNet*, introducing a specific learning scheme adapted to this context. A major benefit of ConfidNet is to be a separate network which can estimate the model confidence of any trained classifier. We evaluate our approach on the task of failure prediction and selective classification and we validate that the proposed approach provides accurate confidence estimates. We study various network architectures and experiment with small and large datasets for image classification and semantic segmentation. In every tested benchmark, our approach outperforms strong baselines.*

The work described in this chapter is based on the following publications:

- *Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, Patrick Pérez. "Addressing Failure Prediction by Learning Model Confidence". In Advances in Neural Information Processing Systems (NeurIPS), 2019.*
- *Charles Corbière, Nicolas Thome, Antoine Saporta, Tuan-Hung Vu, Matthieu Cord, Patrick Pérez. "Confidence Estimation via Auxiliary Models". In IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.*

Contents

3.1 Context

Propagating an erroneous prediction of a machine learning system or over-estimating its confidence may carry serious repercussions in critical visual-recognition applications such as in autonomous driving, medical diagnosis [137] or nuclear power plant monitoring [138]. In classification, *failure prediction* is the task of predicting at run-time whether a trained model has taken a correct decision or not for a given input. By detecting an erroneous prediction, a system could decide to stick to the prediction or, on the contrary, to hand it over to a human or a back-up system with, *e.g.* other sensors, or simply to trigger an alarm. Closely related to failure prediction, classification with a reject option [41], also known as *selective classification* [44], consists in a scenario where the classifier is given the option to reject an instance instead of predicting its label. These two tasks refer to the same problem of *ordinal ranking*, which aims to estimate confidence values whose ranking of samples is effective to distinguish correct from incorrect predictions (see Fig. 3.1). Then, the user can specify a threshold so that some inputs with predicted confidence is below it are considered as erroneous predictions.

In failure prediction, a widely used baseline with neural-network classifiers is to take the value of the predicted class' probability, namely the *maximum class probability* (MCP), given by the softmax layer output. Although recent evaluations of MCP with modern deep models reveal reasonable performance [45], they still suffer from several conceptual drawbacks. In particular, MCP leads by design to high confidence values, even for erroneous predictions, since the largest softmax output is used. This design tends to make erroneous and correct predictions overlap in terms of confidence and thus limits the capacity to distinguish them. Another common uncertainty measure is the predictive

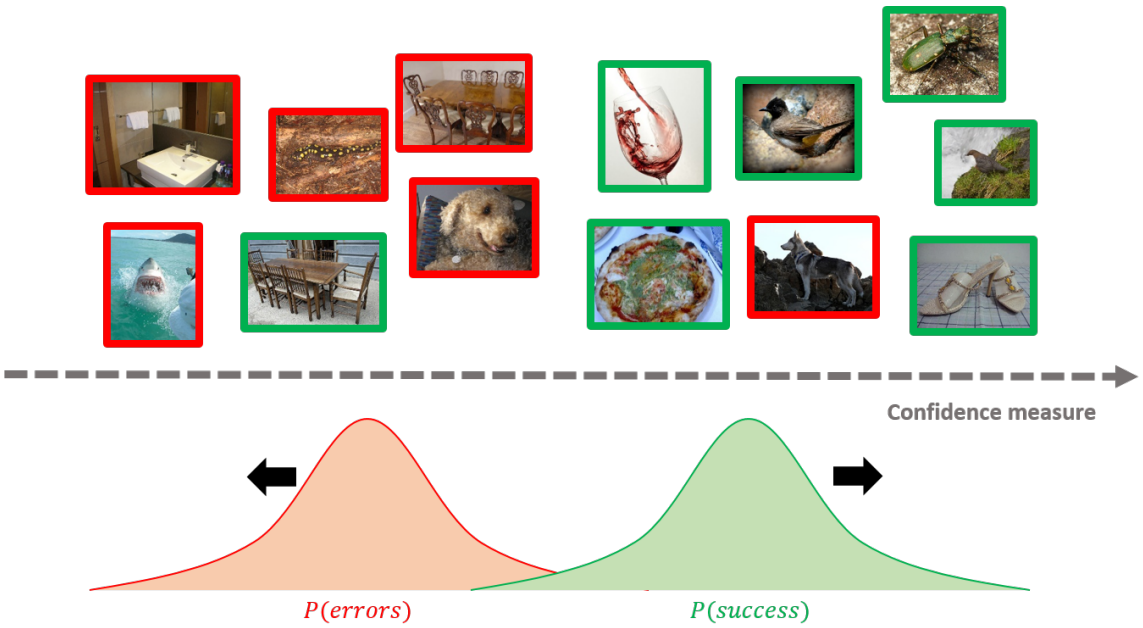


Figure 3.1: **Illustration of an effective confidence measure for ordinal ranking on in-distribution samples.** When ranking samples according to their confidence score, correct predictions (in green) should have higher values on average than misclassifications (in red) to enable the model to distinguish them.

entropy [139] which captures the average amount of information contained in the probability vector output by the model. It is worth mentioning that these entropy-based criteria measure the softmax output dispersion, where the uniform distribution has maximum entropy. But it is not clear how well these dispersion measures are adapted to distinguishing failures from correct predictions. We elaborate on these limits in Section 3.2.2.

In this chapter, we identify a better confidence criterion, the *true class probability* (TCP), for deep neural network classifiers with a reject option (Section 3.2). We provide simple guarantees of the quality of this criterion regarding confidence estimation. Since the true class is obviously unknown at test time, we propose a novel approach, *ConfidNet*, which consists in designing an auxiliary network specifically dedicated to estimate the confidence of a prediction (Section 3.3). Given a trained classifier f, this auxiliary network learns the TCP criterion from data. When applied to failure prediction, we observe significant improvements over strong baselines (Section 3.5.2). A thorough analysis of our approach, including relevant variations, ablation studies and qualitative evaluations of confidence estimates, helps to gain insight about its behavior in Section 3.5.3.

3.2 Defining a confidence measure for effective ordinal ranking

In this section, we first briefly introduce the task of classification with a reject option, along with necessary notations. We also address semantic image segmentation, which can be seen as a

pixel-wise classification problem, where a model outputs a dense segmentation mask with a predicted class assigned to each pixel. As such, all the following material is formulated for classification, and implementation details for segmentation are specified when necessary. We point out the limits of current measures and present our effective confidence-rate function for neural-net classifiers.

3.2.1 Problem formulation

Following notations introduced in Chapter 2, we consider a training dataset $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$ composed of N *i.i.d.* training samples, where $\boldsymbol{x}_n \in \mathcal{X} \subset \mathbb{R}^D$ is a D-dimensional data representation, deep feature maps from an image or the image itself for instance, and $y_n \in \mathcal{Y} = [\![1, K]\!]$ is its true class among the K predefined categories. These samples are drawn from an unknown joint distribution $P(X, Y)$ over $(\mathcal{X}, \mathcal{Y})$.

Definition 3.1 (Selective classifier)**.** *A selective classifier [44, 103] is a pair (f, g) where $f : \mathcal{X} \to \mathcal{Y}$ is a prediction function and $g : \mathcal{X} \to \{0, 1\}$ is a selection function which enables to reject a prediction:*

$$(f, g)(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}), & \text{if } g(\boldsymbol{x}) = 1\,, \\ \text{reject}, & \text{if } g(\boldsymbol{x}) = 0\,. \end{cases} \tag{3.1}$$

We focus on classifiers based on artificial neural networks. Given an input $\boldsymbol{x}$, such a network F with parameters $\boldsymbol{\theta}$ outputs non-negative scores over all classes, which are normalized through softmax. If well trained, this output can be interpreted as the predictive distribution $F(\boldsymbol{x}; \hat{\boldsymbol{\theta}}) = P(Y|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) \in \Delta^{K-1}$, with Δ^{K-1} the probability (K-1)-simplex in $\mathbb{R}^K$ and $\hat{\boldsymbol{\theta}}$ the learned weights. Based on this distribution, the predicted sample class is usually the *maximum-a-posteriori* estimate:

$$f(\boldsymbol{x}) = \underset{k \in \mathcal{Y}}{\operatorname{argmax}}\ P(Y = k|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) = \underset{k \in \mathcal{Y}}{\operatorname{argmax}}\, F(\boldsymbol{x}; \hat{\boldsymbol{\theta}})[k]. \tag{3.2}$$

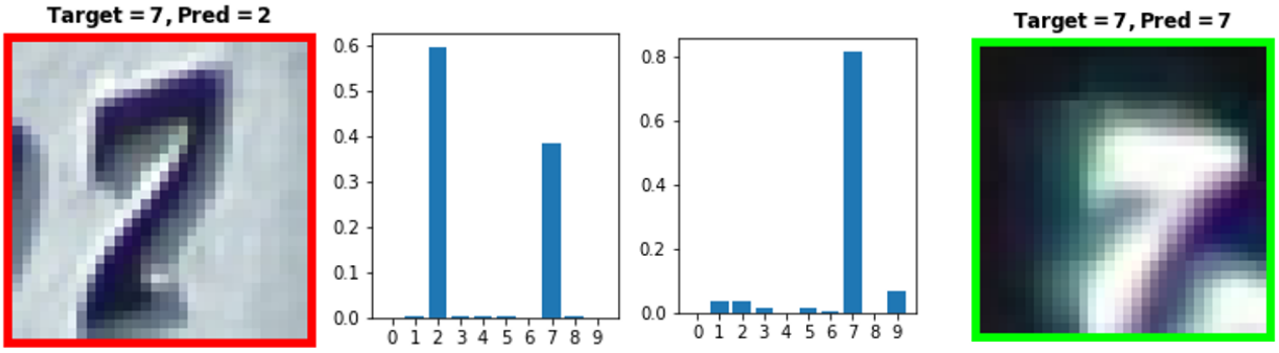
We are not interested here in trying to improve the accuracy of the already-trained model F, but rather in making its future use more reliable by endowing the system with the ability to recognize when the prediction might be wrong.

To this end, a *confidence-rate function* $\kappa_f : \mathcal{X} \to \mathbb{R}^+$ is associated to f so as to assess the degree of confidence of its predictions, the higher the value the more certain the prediction [44, 103]. A suitable confidence-rate function should correlate erroneous predictions with low values and successful predictions with high values. Finally, given a user-defined threshold $\delta \in \mathbb{R}^+$, the selection function g can be simply derived from the confidence rate:

$$g(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \kappa_f(\boldsymbol{x}) \geq \delta\,, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

3.2.2 Limits of current uncertainty measures

For a given input $\boldsymbol{x}$, a standard uncertainty measure for a classifier F is the probability associated to the predicted max-score class, that is the *maximum class probability*:



(a) Erroneous prediction, entropy = 0.79

(b) Correct prediction, entropy = 0.79

Figure 3.2: **Illustrating the limits of predictive entropy as confidence estimation on the SVHN test samples**. Red-border image (Fig. 3.2a) is misclassified by the classification model; green-border image (Fig. 3.2b) is correctly classified. Predictions exhibit similar high entropy in both cases. For each sample, we provide a plot of their softmax predictive distribution.

Definition 3.2 (Maximum Class Probability)**.** *For a given input* $\boldsymbol{x}$ *and a classifier* F*, the Maximum Class Probability (MCP) is defined as:*

$$\mathrm{MCP}_F(\boldsymbol{x}) = \max_{k \in \mathcal{Y}} P(Y = k | \boldsymbol{x}, \hat{\boldsymbol{\theta}}) = \max_{k \in \mathcal{Y}} F(\boldsymbol{x}; \hat{\boldsymbol{\theta}})[k]. \tag{3.4}$$

However, by taking the largest softmax probability as a confidence estimate, MCP leads to high confidence values both for correct and erroneous predictions alike, making it hard to distinguish them, as shown in Fig. 3.3a.

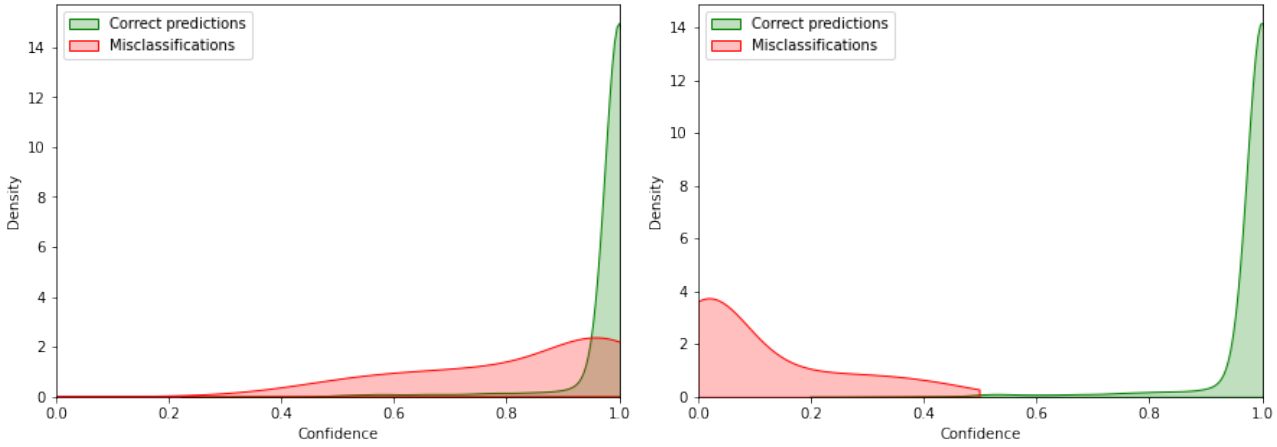
Taking the predictive entropy as uncertainty measure may not also be always adequate. In Fig. 3.2, we show side-by-side two samples with a similar distribution entropy taken from a small convolutional network trained on SVHN, a street-view numbers dataset [36]. Left image (red-border) is misclassified while the right one enjoys a correct prediction (green-border). Predictions exhibit similar high entropy in both cases. But as entropy is a symmetric measure in regards to class probabilities: a correct prediction with $[0.65, 0.35]$ distribution is evaluated as confident as an incorrect one with $[0.35, 0.65]$ distribution, which is undesirable for accurate failure prediction.

3.2.3 The True Class Probability

When the model misclassifies an example, the probability associated to the true class y is lower than the maximum one and likely to be low. Based on this simple observation, we propose to consider instead this *true class probability* as a suitable confidence-rate function.

Definition 3.3 (True Class Probability)**.** *Given a classifier* F*, for any admissible input* $\boldsymbol{x} \in \mathcal{X}$ *we assume the true class* $y(\boldsymbol{x})$ *is known, which we denote* y *for simplicity. The True Class Probability (TCP) is defined as*

$$\mathrm{TCP}_F(\boldsymbol{x},\, y) = P(Y = y |\, \boldsymbol{x}, \hat{\boldsymbol{\theta}}) = F(\boldsymbol{x}; \hat{\boldsymbol{\theta}})[y]. \tag{3.5}$$



(a) Maximum Class Probability

(b) True Class Probability

Figure 3.3: **Distributions of MCP and TCP confidence estimates computed over correct and erroneous predictions by a trained VGG-16 model on CIFAR-10.** When ranking according to MCP (a) the test predictions of a convolutional model trained on CIFAR-10, we observe that correct ones (in green) and misclassifications (in red) overlap considerably, making it difficult to distinguish them. On the other hand, ranking samples according to TCP (b) alleviates this issue and allows a much better separation.

In Fig. 3.3, we can observe that TCP allows a much better separation than MCP. In particular, TCP offers the following interesting guarantees regarding its ability to characterize correct or erroneous predictions of a model.

Proposition 3.1. *Given a properly labelled example* $(\boldsymbol{x}, y)$*, then:*

- $\text{TCP}_F(\boldsymbol{x}, y) > 1/2 \;\Rightarrow f(\boldsymbol{x}) = y$*, i.e. the example is correctly classified by the model;*
- $\text{TCP}_F(\boldsymbol{x}, y) < 1/K \Rightarrow f(\boldsymbol{x}) \neq y$*, i.e. the example is wrongly classified by the model,*

where class prediction $f(\boldsymbol{x})$ *is defined by Eq. (3.2).*

Proof. Let F be a trained neural network classifier with learned weights $\hat{\boldsymbol{\theta}}$, K be the number of labels and $\boldsymbol{x} \in \mathbb{R}^D$ a sample with its associated true label $y \in \mathcal{Y}$ such that $\text{TCP}_F(\boldsymbol{x}, y) > \frac{1}{2}$. Starting from the definition of TCP we have:

$$\text{TCP}_F(\boldsymbol{x}, y) = P(Y = y|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) > \frac{1}{2} \tag{3.6}$$

$$\iff 1 - \sum_{k \in \mathcal{Y}, k \neq y} P(Y = k|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) > \frac{1}{2} \tag{3.7}$$

$$\iff \sum_{k \in \mathcal{Y}, k \neq y} P(Y = k|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) < \frac{1}{2}. \tag{3.8}$$

Since probabilities are positive, we obtain that $\forall k \neq y$, $P(Y = k|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) < \frac{1}{2} < P(Y = y|\boldsymbol{x}, \hat{\boldsymbol{\theta}})$. Denoting $\hat{y} = f(\boldsymbol{x})$ the class predicted by the network, we have $\hat{y} = \arg\max_{k \in \mathcal{Y}} P(Y = k|\boldsymbol{x}, \hat{\boldsymbol{\theta}})$. Hence $\hat{y} = y$.

In the same way, for any $(\boldsymbol{x}, y) \in \mathbb{R}^D \times \mathcal{Y}$, such that $\text{TCP}_F(\boldsymbol{x}, y) < \frac{1}{K}$, we have:

$$P(Y = y|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) < \frac{1}{K} \tag{3.9}$$

$$\iff 1 - \sum_{k \in \mathcal{Y}, k \neq y} P(Y = k|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) < \frac{1}{K} \tag{3.10}$$

$$\iff \sum_{k \in \mathcal{Y}, k \neq y} P(Y = k|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) > \frac{k-1}{K}. \tag{3.11}$$

If the model correctly classifies this sample, *i.e.*, $\hat{y} = y$, then $\forall k \neq y$, $P(Y = y|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) \geq P(Y = k|\boldsymbol{x}, \hat{\boldsymbol{\theta}})$. We have:

$$\sum_{K \in \mathcal{Y}, K \neq y} P(Y = k|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) \leq (K-1)P(Y = y|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) \leq \frac{K-1}{K}, \tag{3.12}$$

which contradicts Eq. (3.11). Hence, there exists at least one k such that $P(Y = k|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) > P(Y = y|\boldsymbol{x}, \hat{\boldsymbol{\theta}})$, which results in $\hat{y} \neq y$. □

Within the range $[1/K, 1/2]$, there is no guarantee that correct and incorrect predictions will not overlap in terms of TCP. However, when using deep neural networks, we observe that the actual overlap area is extremely small in practice, as illustrated in Fig. 3.3b on the CIFAR-10 dataset. One possible explanation comes from the fact that modern deep neural networks output overconfident predictions and therefore non-calibrated probabilities (see Section 2.3.3). We provide consolidated analyses on this aspect in Section 3.5 and further results on other datasets in Section A.2.

We also introduce a normalized variant of the TCP confidence criterion, which consists in computing the *ratio* between TCP and MCP:

Definition 3.4 (Normalized True Class Probability)**.** *Given a classifier F, for any admissible input $\boldsymbol{x} \in \mathcal{X}$ we assume the true class $y(\boldsymbol{x})$ is known, which we denote y for simplicity. The normalized True Class Probability (nTCP) is defined as*

$$n\text{TCP}_F(\boldsymbol{x}, y) = \frac{P(Y = y|\,\boldsymbol{x}, \hat{\boldsymbol{\theta}})}{P(Y = \hat{y}|\,\boldsymbol{x}, \hat{\boldsymbol{\theta}})}. \tag{3.13}$$

The normalized criterion nTCP presents stronger theoretical guarantees than TCP, since correct predictions will be, by design, assigned the value of 1, whereas errors will range in $[0, 1[$. On the other hand, learning this criterion may be more challenging since all correct predictions must match a single scalar value.

3.3 ConfidNet: learning to predict TCP with an auxiliary model

Using TCP as a confidence-rate function on a model's output would be of great help when it comes to reliably estimate its confidence. However, the true classes y are obviously not available when estimating confidence on test inputs.

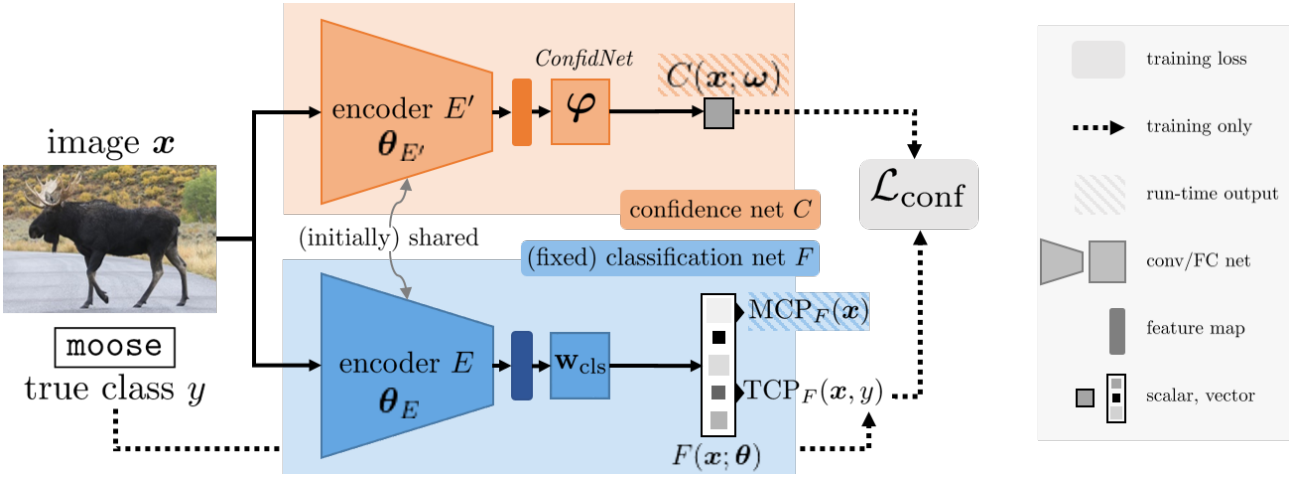


Figure 3.4: **Learning confidence approach.** The fixed classification network F, with parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_E, \boldsymbol{\theta}_{\text{cls}})$, is composed of a succession of convolutional and fully-connected layers (encoder E) followed by last classification layers with softmax activation. The auxiliary confidence network C, with parameters $\boldsymbol{\omega}$, builds upon the feature maps extracted by the encoder E, or its fine-tuned version E' with parameters $\boldsymbol{\theta}_{E'}$: they are passed to ConfidNet, a trainable multi-layer module with parameters $\boldsymbol{\varphi}$. The auxiliary model outputs a confidence score $C(\boldsymbol{x};\boldsymbol{\omega}) \in [0,1]$, with $\boldsymbol{\omega} = \boldsymbol{\varphi}$ in absence of encoder fine-tuning and $\boldsymbol{\omega} = (\boldsymbol{\theta}_{E'}, \boldsymbol{\varphi})$ in case of fine-tuning.

3.3.1 Principle

We propose to *learn TCP confidence from data*. More formally, for the classification task at hand, we consider a parametric selective classifier (f, g), with f based on an already-trained neural network F. We aim at deriving its companion selection function g from a learned estimate of the TCP function of F. To this end, we introduce an *auxiliary model* C, with parameters $\boldsymbol{\omega}$, that is intended to predict TCP_F and to act as a confidence-rate function for the selection function g. An overview of the proposed approach is available in Fig. 3.4. This model is trained such that, at runtime, for an input $\boldsymbol{x} \in \mathcal{X}$ with (unknown) true label y, we have:

$$C(\boldsymbol{x};\boldsymbol{\omega}) \approx \text{TCP}_F(\boldsymbol{x}, y). \tag{3.14}$$

In practice, this auxiliary model C will be a neural network trained under full supervision on $\mathcal{D}$ to produce this confidence estimate. To design this network, we can transfer knowledge from the already-trained classification network. Throughout its training, F has indeed learned to extract increasingly-complex features that are fed to its final classification layers. Calling E the encoder part of F, a simple way to transfer knowledge consists in defining and training a multi-layer head with parameters $\boldsymbol{\varphi}$ that regresses TCP_F from features encoded by E. We call *ConfidNet* this module. As a result of this design, the complete confidence network C is composed of a frozen encoder followed by trained ConfidNet layers. The complete architecture can be later fine-tuned, including the encoder, as in classic transfer learning. In that case, $\boldsymbol{\omega}$ will encompass the parameters of both the encoder and the ConfidNet's layers.

3.3.2 Architecture

Standard image classification models are composed of convolutional layers followed by one or more fully-connected layers and a final softmax operation. In order to work with such a classification network F, we build ConfidNet upon a late intermediate representation of F. ConfidNet is designed as a small multilayer perceptron composed of a succession of dense layers with a final sigmoid activation that outputs $C(\boldsymbol{x};\boldsymbol{\omega}) \in [0,1]$. ConfidNet is train in a supervised manner, such that it predicts well the true-class probability assigned by F to the input image. Regarding the capacity of ConfidNet, we have empirically found that increasing further its depth leaves performance unchanged for estimating the confidence of the classification network (see Section 3.5.3).

3.3.3 Loss function

As we want to regress a score between 0 and 1, we use a mean-square-error (MSE) loss to train the confidence model:

$$\mathcal{L}_{\text{conf}}(\boldsymbol{\omega};\mathcal{D}) = \frac{1}{N}\sum_{n=1}^{N}\left(C(\boldsymbol{x}_n;\boldsymbol{\omega}) - \text{TCP}_F(\boldsymbol{x}_n, y_n)\right)^2. \tag{3.15}$$

Since the final task here is the prediction of failures, with confidence prediction being only a means toward it, a more explicit supervision with failure/success information could be considered. In that case, the previous regression loss could still be used, with 0 (failure) and 1 (success) target values instead of TCP. Alternatively, a binary cross entropy loss (BCE) for the error-prediction task using the predicted confidence as a score could be used. Seeing failure detection as a ranking problem, where good predictions must be ranked before erroneous ones according to the predicted confidence, a batch-wise ranking loss can also be utilized [140]. We experimentally assessed all these alternative losses, including a focal version [141] of the BCE to focus on hard examples, as discussed in Section 3.5.3. They lead to inferior performance compared to using Eq. (3.15). This might be due to the fact that TCP conveys more detailed information than a mere binary label on the quality of the classifier's prediction for a sample. Hinton *et al.* [142] make a similar observation when using soft targets in knowledge distillation. In situations where only very few error samples are available, this finer-grained information improves the performance of the final failure detection (see Section 3.5.3).

3.3.4 Learning scheme

We decompose the parameters of the classification network F into $\boldsymbol{\theta} = (\boldsymbol{\theta}_E, \boldsymbol{\theta}_{\text{cls}})$, where $\boldsymbol{\theta}_E$ denotes its encoder's weights and $\boldsymbol{\theta}_{\text{cls}}$ the weights of its last classification layers. Such as in transfer learning, the training of the confidence network C starts by fixing the shared encoder and training only ConfidNet's weights $\boldsymbol{\varphi}$. In this phase, the loss Eq. (3.15) is thus minimized only w.r.t. $\boldsymbol{\omega} = \boldsymbol{\varphi}$.

In a second phase, we further fine-tune the complete network C, including its encoder which is now untied from the classification encoder E (the main classification model must remain unchanged, by definition of the addressed problem). Denoting E' this now independent encoder, and $\boldsymbol{\theta}_{E'}$ its weights, this second training phase optimizes Eq. (3.15) w.r.t. $\boldsymbol{\omega} = (\boldsymbol{\theta}_{E'}, \boldsymbol{\varphi})$ with $\boldsymbol{\theta}_{E'}$ initially set to $\boldsymbol{\theta}_E$.

We also deactivate dropout layers in this last training phase and reduce learning rate to mitigate stochastic effects that may lead the new encoder to deviate too much from the original one used for classification. Data augmentation can thus still be used. ConfidNet can be trained using either the original training set or a validation set. The impact of this choice is evaluated in Section 3.5.3.

3.4 Related work

Confidence estimation [43, 42] has a long history in the machine learning community, tightly related to classification with a reject option [41]. The following works [143, 144, 104] explored alternative rejection criteria. In particular, [104] proposes to jointly learn the classifier and the selection function. El-Yaniv [44] provides an analysis of the risk-coverage trade-off that occurs when classifying with a reject option. More recently, [103, 105] extend the approach to deep neural networks, considering various confidence measures. Since the wide adoption of deep learning methods, confidence estimation has raised even more interest as recent works [46, 48] reveal that modern neural networks tend to be overconfident and provide unreliable uncertainty estimates.

Bayesian neural networks [79] offer a principled approach for confidence estimation by adopting a Bayesian formalism which models the weight posterior distribution. As the true posterior cannot be evaluated analytically in complex models, various approximations have been developed, such as variational inference [85, 52, 40] or expectation propagation [81]. In particular, MC Dropout [40] has raised a lot of interest due to the simplicity of its implementation. Predictions are obtained by averaging softmax vectors from multiple feed-forward passes through the network with dropout layers. When applied to regression, the predictive distribution uncertainty can be summarized by computing statistics, *e.g.*, variance. However, when using MC Dropout for uncertainty estimation in classification tasks, the predictive distribution is averaged to a point-wise softmax estimate before computing standard uncertainty criteria such as entropy. It is worth mentioning that these entropy-based criteria measure the softmax output dispersion, where the uniform distribution has maximum entropy. It is not clear how well these dispersion measures are adapted to distinguishing failures from correct predictions as we will see in Section 3.2.2. [62] presented a framework to decompose the uncertainty into aleatoric and epistemic terms. But it requires multiple forward passes for inference. Lakshminarayanan *et al.* [53] propose an alternative to Bayesian neural networks by leveraging an ensemble of neural networks to produce well-calibrated uncertainty estimates. However, it requires training multiple classifiers, which has a considerable computing cost in training and inference time.

In failure prediction, a widely used baseline is to take the value of the predicted class' probability given by the softmax layer output, namely the *maximum class probability* (MCP), suggested by [145] and revised by [45]. As stated before, MCP presents several limits regarding both failure prediction and out-of-distribution detection, as it outputs unduly high confidence values. More recently, [146] proposed a new confidence measure, 'Trust Score', which measures the agreement between the classifier and a modified nearest-neighbor classifier on the test examples. More precisely, the confidence criterion used in Trust Score is the ratio between the distance from the sample to the nearest class different from the predicted class and the distance to the predicted class. One clear drawback of this approach

is its lack of scalability, since computing nearest neighbors in large datasets is extremely costly in both computations and memory. Another more fundamental limitation related to the Trust Score itself is that local distance computation becomes less meaningful in high dimensional spaces [147], which is likely to negatively affect the performances of this method as shown in experiments. Finally, DeVries & Taylor [148] share with us the same purpose of learning confidence in neural networks. Their work differs by focusing on out-of-distribution detection and learning jointly a distribution confidence score and classification probabilities. In addition, they use predicted confidence scores to interpolate output probabilities and target whereas we specifically craft our proposed criterion for failure prediction.

In tasks closely related to failure prediction, Guo *et al.* [49], for confidence calibration, and Liang *et al.* [112], for out-of-distribution detection, proposed to use temperature scaling to mitigate confidence values. However, this does not affect the ranking of confidence scores and therefore the separability between errors and correct predictions. For instance, we plot in Fig. 3.5 the distribution of MCP confidence estimates after temperature scaling ith a VGG-16 model on CIFAR-10. The temperature parameter T has been found using validation data, such as described in [49]. Even though overconfidence is reduced, the separability between errors and correct predictions still remains problematic.

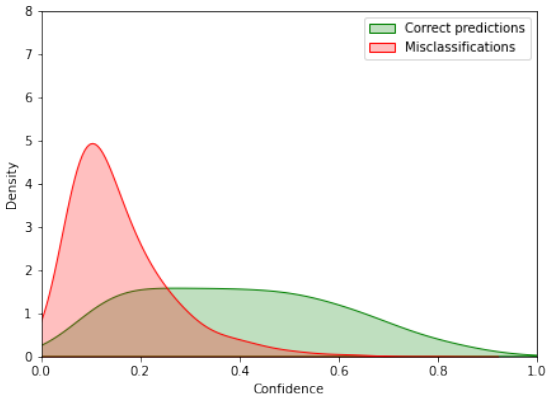


Figure 3.5: Distribution of MCP after temperature scaling for a VGG-16 on CIFAR-100.

3.5 Application to failure prediction

We evaluate our approach to predict failures in both classification and segmentation settings. First, we run comparative experiments against strong confidence estimation and Bayesian uncertainty estimation methods on various datasets. These results are then completed by a thorough analysis of the influence of the confidence criterion, the training loss and the learning scheme in our approach. Finally, we provide a few visualizations to get additional insight into the behavior of our approach. Our code is available at https://github.com/valeoai/ConfidNet.

3.5.1 Experiment setup

Datasets. We run experiments on image datasets of varying scale and complexity: MNIST [149] and SVHN [36] datasets provide relatively simple and small (28×28) images of digits (10 classes). They are split into 60,000 training samples and 10,000 testing samples. CIFAR-10 and CIFAR-100 [35] bring more complexity to classify low resolution images. In each dataset, we further keep 10% of training samples as a validation dataset. We also report experiments for semantic segmentation on CamVid [150], using ConfidNet's training and architecture introduced in Section 3.3.2, with dense layers replaced by 1×1 convolutions with an adequate number of channels. CamVid is a standard

road scene dataset. Images are resized to 360×480 pixels and are segmented according to 11 classes such as 'road', 'building', 'car' or 'pedestrian'.

Classification network. For each dataset, we use standard neural network architectures as classifiers. Network architectures range from small convolutional networks for MNIST [149] and SVHN [36] to VGG-16 architectures [74] for CIFAR datasets [35]. We also added a multi-layer perceptron (MLP) with 1 hidden layer of size 100 for MNIST dataset in order to investigate performances on small models. Finally, we implemented SegNet following [47]. All models are trained in a standard way with a cross-entropy loss and an SGD optimizer with a learning rate of 10^{-3}, a momentum of 0.9 and a weight decay of 10^{-4}. The number of training epochs depends on the dataset considered, varying from 100 epochs on MNIST to 250 epochs on CIFAR-100. As we also want to compute Monte Carlo samples following [40], we include dropout layers. Best models are selected on validation-set accuracy.

Baselines. To demonstrate the effectiveness of our method, we implemented competitive confidence and uncertainty estimation approaches including Maximum Class Probability (MCP) as a baseline [45], Trust Score [151], and Monte-Carlo Dropout (MC Dropout) [40]. For Trust Score, we used the code provided by the authors[1]. With MC Dropout, we use the same model as baseline (which already includes dropout layers) and we sample 100 times from the classification model at test time keeping dropout layers activated. We then compute the average softmax probability over all samples to conduct Monte Carlo integration. Model uncertainty is estimated by calculating the entropy of the averaged probability vector across the class dimension.

Evaluation metrics We measure the quality of failure prediction following standard metrics used in the literature [45]. We enumerate these metrics in the following and refer the reader to Section 2.3 for a more detail description:

- **FPR at 95% TPR** measures the False Positive Rate (FPR) when the True Positive Rate (TPR) is equal to 95%. True Positive Rate can be computed by $\text{TPR} = \text{TP}/(\text{TP}+\text{FN})$, where TP and FN denote numbers of true positives and false negatives respectively. The False Positive Rate can be computed by $\text{FPR} = \text{FP}/(\text{FP}+\text{TN})$, where FP and TN denote the number of false positives and true negatives respectively. This metric can be interpreted as the probability that an error is misclassified as a correct prediction when the True Positive Rate (TPR) is as high as 95%.

- **AUROC** measures the Area Under the Receiver Operating Characteristic curve (AUROC). The ROC curve is a graph showing True Positive Rate versus False Positive Rate. This metric is a threshold-independent performance evaluation, such as AUPR. It can be interpreted as the probability that a positive example has a greater prediction score than a negative example.

[1] https://github.com/google/TrustScore

- **AUPR** measures the Area Under the Precision-Recall (PR) curve. The PR curve is a graph showing precision $= \text{TP}/(\text{TP} + \text{FP})$ versus recall $= \text{TP}/(\text{TP} + \text{FN})$. As we specifically want to detect failures, we use AUPR-Error (shortened here AUPR) as the primary metrics to assess performances.

As an additional, indirect way to assess the quality of the predicted classifier's confidence, we also consider the selective classification problem that was discussed in Section 2.3.1. In this setup, the predictions by the classifier F that get a predicted confidence below a defined threshold are rejected. Given a coverage rate (the fraction of examples that are not rejected), the performance of the classifier should improve. The impact of this selection is measured in average with:

- **Area under the risk-coverage curve (AURC)**. In classification with a reject option, the risk-coverage curve is the graph of the empirical risk of the classifier given a loss (usually 0/1 loss) as a function of the empirical coverage, which is the proportion of the non-rejected samples. This metric is threshold-independent, as AUROC and AUPR.
- **Excess-AURC (E-AURC)**. This is a normalized AURC metric defined in [102]. It takes into account the optimal ranking given the error rate of the classifier.

ConfidNet. For each of the considered classification models, ConfidNet is built upon the penultimate layer, which is a convolutional layer with non-linear activation and optionally followed by a normalization layer. We train ConfidNet for 100 epochs with the Adam optimizer with a learning rate 1×10^{-4}, dropout, weight decay 10^{-4} and the same data augmentation as in the classifier's training. The relevance of selecting the same training dataset used for classifier learning or a hold-out dataset is specifically discussed in Section 3.5.3. We select the best model based on the AUPR on the validation dataset. In the second training step involving encoder fine-tuning, the training is completed on very few epochs based on previous best model, using Adam optimizer with learning rate 1×10^{-6} or 1×10^{-7} and no dropout to mitigate stochastic effects that may lead the new encoder to deviate too much from the original one used for classification. Once again, the best model is selected on validation-set AUPR.

3.5.2 Comparative results

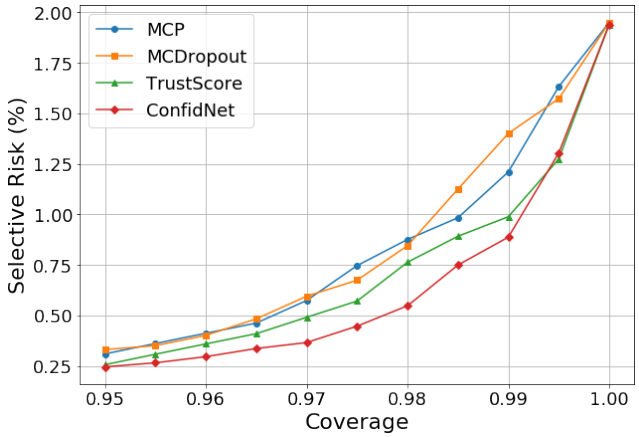
Comparative results are summarized in Table 3.1. We observe that our approach outperforms baseline methods in every setting, with a significant gap on small models/datasets. This confirms both that TCP is an adequate confidence criterion for failure prediction and that our approach ConfidNet is able to learn it. TrustScore also presents good results on small datasets/models such as MNIST where it improved the baseline. While ConfidNet still performs well on more complex datasets, Trust Score's performance drops, which might be explained by high dimensionality issues with distances as mentioned in Section 3.5.1. For its application to semantic segmentation where each training pixel is a 'neighbor', computational complexity forced us to reduce drastically the number of training neighbors and of test samples. We sampled randomly in each train and test image a small percentage of pixels

Table 3.1: **Comparison of confidence estimation methods for failure prediction and selective classification**. For each dataset, all methods share the same classification network. For MC Dropout, test accuracy is averaged through random sampling. The first three metrics are percentages and concern failure prediction. The two last ones (the lower, the better) concern selective classification and their values have been multiplied by 10^3 for clarity. Scores are averaged over 5 runs, best results are in **bold**, second best ones are underlined.

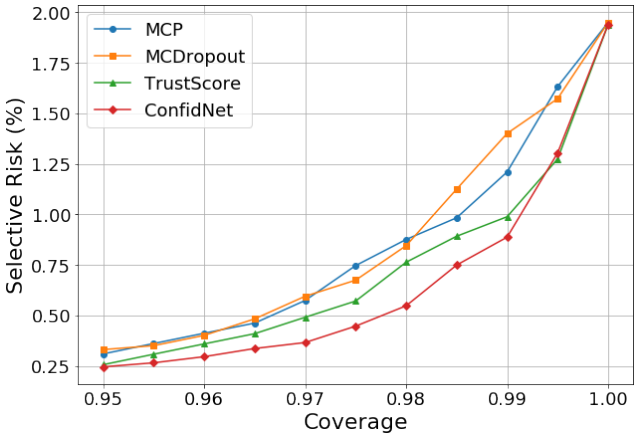
Dataset	Model	FPR @ 95% TPR ↓	AUPR ↑	AUROC ↑	AURC ↓	E-AURC ↓
MNIST MLP	MCP [45]	14.88 ±1.42	47.25 ±1.67	97.28 ±0.20	0.83 ±0.07	0.61 ±0.06
	MC Dropout [40]	15.17 ±1.08	40.98 ±1.24	97.10 ±0.18	0.85 ±0.07	0.63 ±0.06
	TrustScore [151]	14.80 ±2.03	52.13 ±1.79	97.36 ±0.10	0.82 ±0.04	0.59 ±0.03
	ConfidNet	**11.61** ±1.96	**59.72** ±1.90	**97.89** ±0.14	**0.70** ±0.05	**0.47** ±0.04
MNIST SmallConvNet	MCP [45]	5.53 ±1.25	36.08 ±3.60	98.49 ±0.07	0.15 ±0.01	0.12 ±0.01
	MC Dropout [40]	**5.03** ±0.72	42.12 ±5.52	98.53 ±0.12	0.16 ±0.01	0.12 ±0.01
	TrustScore [151]	9.60 ±2.69	33.47 ±3.82	98.20 ±0.23	0.18 ±0.03	0.15 ±0.02
	ConfidNet	5.32 ±1.14	**45.45** ±3.75	**98.72** ±0.07	**0.13** ±0.02	**0.10** ±0.01
SVHN SmallConvNet	MCP [45]	32.17 ±0.91	46.20 ±0.50	92.93 ±0.13	5.58 ±0.14	4.50 ±0.09
	MC Dropout [40]	33.54 ±1.06	45.15 ±1.29	92.84 ±0.08	5.70 ±0.11	4.61 ±0.09
	TrustScore [151]	34.01 ±1.11	44.77 ±1.30	92.65 ±0.29	5.72 ±0.11	4.64 ±0.12
	ConfidNet	**29.90** ±0.76	**48.64** ±1.08	**93.15** ±0.15	**5.51** ±0.09	**4.43** ±0.08
CIFAR-10 VGG16	MCP [45]	49.19 ±1.42	48.37 ±0.69	91.18 ±0.32	12.66 ±0.61	8.71 ±0.50
	MC Dropout [40]	49.67 ±2.66	48.08 ±0.99	90.70 ±1.96	13.31 ±2.63	9.46 ±2.41
	TrustScore [151]	54.37 ±1.96	41.80 ±1.97	87.87 ±0.41	17.97 ±0.45	14.02 ±0.34
	ConfidNet	**45.08** ±1.58	**53.72** ±0.55	**92.05** ±0.34	**11.78** ±0.58	**7.88** ±0.44
CIFAR-100 VGG16	MCP [45]	66.55 ±1.56	71.30 ±0.41	85.85 ±0.14	113.23 ±2.98	51.93 ±1.20
	MC Dropout [40]	63.25 ±0.66	71.88 ±0.72	86.71 ±0.30	**101.41** ±3.45	**46.45** ±1.91
	TrustScore [151]	71.90 ±0.93	66.77 ±0.52	84.41 ±0.15	119.41 ±2.94	58.10 ±1.09
	ConfidNet	**62.70** ±1.04	**73.55** ±0.57	**87.17** ±0.21	108.46 ±2.62	47.15 ±0.95
CamVid SegNet	MCP [45]	63.87 ±0.76	48.53 ±0.34	84.42 ±0.09		
	MCDropout [40]	62.95 ±0.72	49.35 ±0.30	84.58 ±0.08		
	TrustScore [151]		20.42 ±1.02	68.33 ±1.17		
	ConfidNet	**61.52** ±0.67	**50.51** ±0.26	**85.02** ±0.08		

to compute TrustScore. ConfidNet, in contrast, is as fast as the original segmentation network. We also improve state-of-art performances at the time of publication from MCDropout, whose drawbacks regarding failure prediction have been highlighted previously in Fig. 3.2.

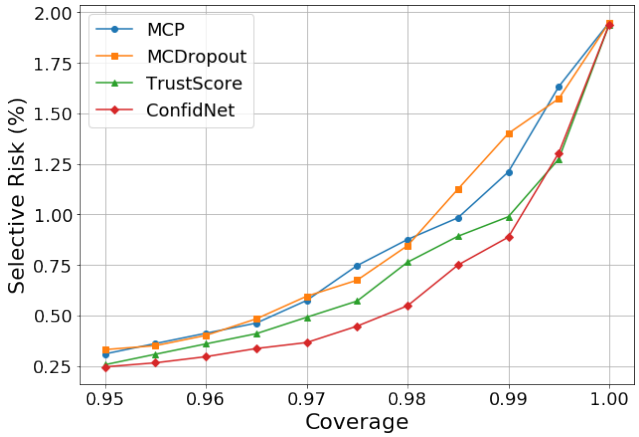
Risk-coverage curves [44, 103] depicting the performance of ConfidNet and other baselines for every tested dataset appear in Fig. 3.6. 'Coverage' corresponds to the probability mass of the non-rejected region after using a threshold as a selection function [103]. For both datasets, ConfidNet presents a better coverage potential for each selective risk that a user can choose beforehand. In addition, we can see that the improvement is more pronounced at high coverage rates - *e.g.* in $[0.8, 0.95]$ for CIFAR-10 (Fig. 3.6d) and in $[0.86, 0.96]$ for SVHN (Fig. 3.6c) - which highlights the capacity of ConfidNet to identify successfully critical failures.



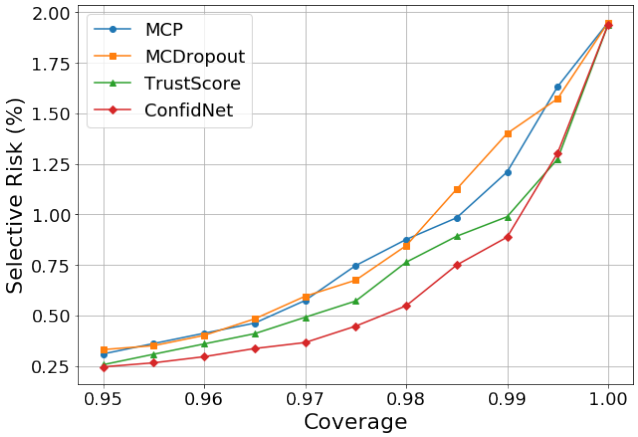
(a) MLP on MNIST



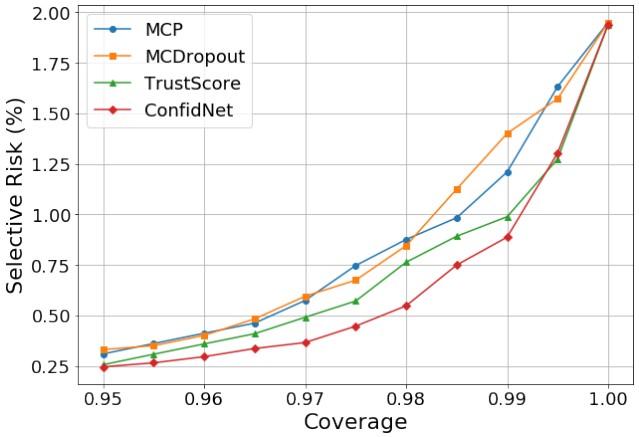
(b) Small ConvNet on MNIST



(c) Small ConvNet on SVN



(d) VGG-16 on CIFAR-10



(e) VGG-16 on CIFAR-100

Figure 3.6: **Comparison of risk-coverage curves for various uncertainty measures on respective test sets**. 'Selective risk' (y-axis) represents the percentage of errors in the remaining test set for a given coverage percentage.

3.5.3 Effect of learning variants

Confidence loss function. In Table 3.2, we compare training ConfidNet with the MSE loss (Eq. (3.15)) to training with a binary-classification cross-entropy loss (BCE), a focal BCE loss [141] and a batch-wise approximate ranking loss. Even though BCE specifically addresses the failure prediction task, it achieves lower performances on CIFAR-10 datasets. Similarly, the focal loss and the ranking one yield results below TCP's performance in every tested benchmark. Similar results on SVHN and CamVid dataset are available in Section A.1. Our intuition is that TCP regularizes the training by providing finer-grained information about the quality of the classifier's predictions. This is especially important in the difficult learning configuration where only very few error samples are available due to the good performance of the classifier.

Table 3.2: **Effect of loss function with ConfidNet** on CIFAR-10. Results are percentages (%).

Dataset	Loss	FPR@95% TPR↓	AUPR ↑	AUROC ↑
CIFAR-10 VGG-16	BCE	45.20	47.95	91.94
	Focal	45.20	47.76	91.93
	Ranking	46.99	44.04	91.49
	nTCP	45.02	48.78	92.06
	TCP	**44.94**	**49.94**	**92.12**

We also evaluate the impact of regression to the normalized criterion nTCP: performance is lower than the one of TCP on small datasets such as CIFAR-10 where few errors are present, but can be higher on larger datasets such as CamVid where each pixel is a sample (see Table A.1). This emphasizes once again the complexity of incorrect/correct classification training.

Hold-out dataset for training ConfidNet. Most neural networks used in our experiments tend to overfit. On small datasets such as MNIST and SVHN, convolutional neural networks already achieve nearly perfect accuracy on test set, above 96%, which leaves very few errors available. For this reason, we also experimented with training ConfidNet on a hold-out dataset. We report results on all datasets in Table 3.3 for validation sets with 10% of samples. We observe a general performance drop when using a validation set for training TCP confidence. The drop is especially pronounced for small datasets (MNIST), where models reach >97% train and val accuracies. Consequently, with a high accuracy and a small validation set, we do not get a larger absolute number of errors using the validation set compared to the train set. One solution would be to increase validation set size but this would damage the model's prediction performance. By contrast, we take care with our approach to base our confidence estimation on models with levels of test predictive performance that are similar to those of baselines. On CIFAR-100, the gap between train accuracy and validation accuracy is substantial (95.56% vs. 65.96%), which may explain the slight improvement for confidence estimation using the validation set (+0.17%). We think that training ConfidNet on the validation set with models reporting low/middle test accuracies could improve the approach.

Table 3.3: **Ablation study between training ConfidNet on train set or on validation set**. Comparison in AUPR (the higher, the better) on all benchmarks. Results are percentages (%).

	MNIST MLP	**MNIST** SmallConvNet	**SVHN** SmallConvNet	**CIFAR-10** VGG-16	**CIFAR-100** VGG-16	**CamVid** SegNet
ConfidNet (using train set)	57.34	43.94	50.72	49.94	73.68	50.28
ConfidNet (using val set)	33.41	34.22	47.96	48.93	73.85	50.15

Table 3.4: **Impact of the encoder fine-tuning on the error-prediction performance of ConfidNet**. Comparison in AUPR (the higher, the better) on all benchmarks. Results are percentages (%).

	MNIST MLP	**MNIST** SmallConvNet	**SVHN** SmallConvNet	**CIFAR-10** VGG-16	**CIFAR-100** VGG-16	**CamVid** SegNet
Confidence training	58.42	44.54	48.49	50.18	71.30	50.12
+ Fine-tuning ConvNet	59.72	45.45	48.64	53.72	73.55	50.51

ConfidNet's encoder fine-tuning. We analyse in Table 3.4 the effect of the encoder fine-tuning. Learning only ConfidNet on top of the pre-trained encoder E (that is, $\boldsymbol{\omega} = \boldsymbol{\varphi}$), our confidence network already achieves significant improvements w.r.t. the baselines. With a subsequent fine-tuning of both modules (that is, $\boldsymbol{\omega} = (\boldsymbol{\theta}_{E'}, \boldsymbol{\varphi})$), its performance is further boosted in every setting, by around 1-2%. Note that using a vanilla fine-tuning without the deactivation of the dropout layers did not bring any improvement.

ConfidNet's architecture We experiment different architectures for ConfidNet on the SVHN dataset, varying the number of layers. Except for the first and last layers, whose dimensions respectively depend on the size of the input and of the output, each layer presents the same number of units (400). In Fig. 3.7, we observe that starting from 3 layers, ConfidNet already improves baseline performance.

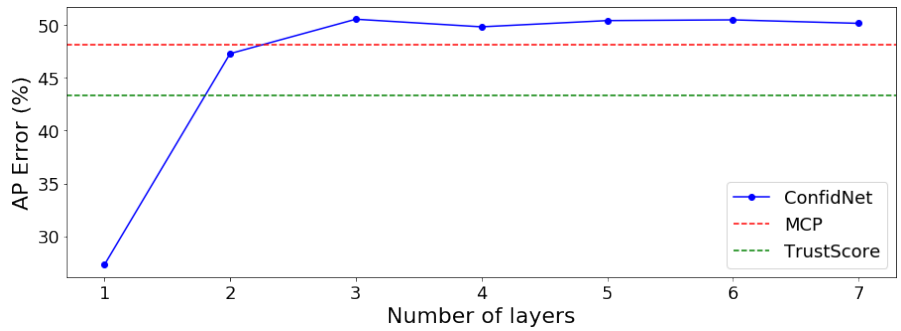


Figure 3.7: **Influence of ConfidNet's depth on its performance**. Performance in AUPR as a function of the number of layers used in ConfidNet on SVHN test set and compared to the performance of MCP and True Score baselines.

3.5.4 Comparison with a two-fold ensemble

As ConfidNet with fine-tuning induces an increased capacity of the whole model by using a complete auxiliary network in conjunction with the original one, one might hypothesize that this contributes to its superior performance. To investigate this question, we compare its performance with the MCP metric taken from an ensemble of two neural networks. Comparative results for each dataset are presented in Table 3.5. At equal computational cost, ConfidNet outperforms an ensemble of two neural networks in every setting.

Table 3.5: **Equal-capacity comparisons**. Comparison between ConfidNet trained on 1 neural network (1NN) and the MCP metric taken from the average prediction of an ensemble of two neural networks (2NNs). Results are percentages (%).

	MNIST MLP		**MNIST** SmallConvNet		**SVHN** SmallConvNet		**CIFAR-10** VGG-16		**CIFAR-100** VGG-16	
Method	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC	AUPR	AUROC
MCP-1NN	47.30	97.22	37.87	98.54	46.17	92.92	47.88	91.45	71.39	85.76
MCP-2NNs	43.70	97.25	43.16	98.70	46.35	92.98	48.81	92.26	70.52	86.38
ConfidNet-1NN	**60.17**	**97.90**	**47.81**	**98.75**	**48.67**	**93.17**	**54.00**	**92.39**	**73.30**	**87.00**

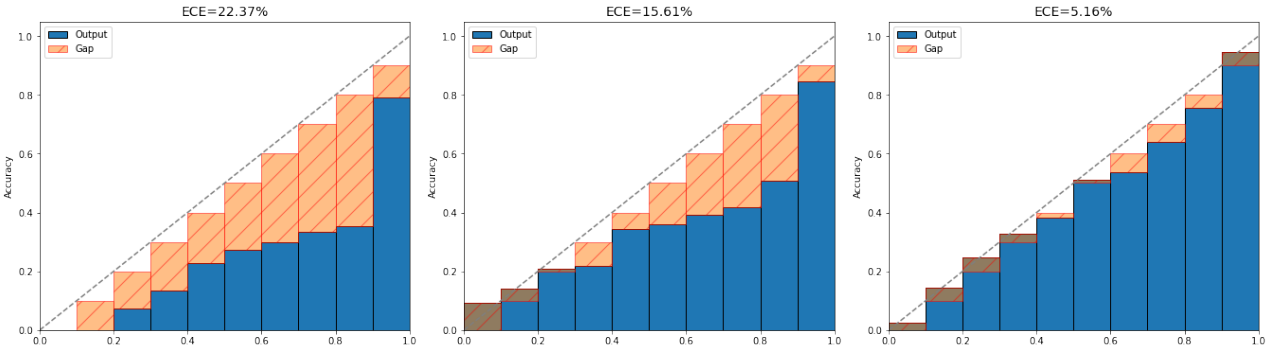
3.5.5 Effect on calibration

Table 3.6: **Comparative calibration results**. Performance in ECE (the lower, the better) when using MCP baseline ('Baseline') or ConfidNet as confidence estimator on the six benchmarks, and when using dedicated temperature scaling ('T. Scaling'). Results are percentages (%)

	MNIST MLP	**MNIST** SmallConvNet	**SVHN** SmallConvNet	**CIFAR-10** VGG-16	**CIFAR-100** VGG-16	**CamVid** SegNet
Baseline	0.37	**0.20**	**0.50**	4.48	22.37	9.65
ConfidNet	0.66	0.30	1.11	3.45	15.61	7.57
Baseline + T. Scaling	**0.20**	0.69	1.30	**2.88**	**5.16**	**4.77**

We observed that ConfidNet tends to lower the confidence of an example that the model wrongly classified while being over-confident (high MCP). As a side experiment, we study whether using ConfidNet as confidence estimation can improve the calibration of deep neural networks.

In Table 3.6, we report the expected calibration error (ECE) which is an approximate measure of miscalibration between confidence and accuracy [49]. ConfidNet yields equivalent or better ECE results than the MCP baseline, with clear superiority on complex datasets such as CIFAR-10, CIFAR-100 and CamVid. On MNIST and SVHN, the baseline already offers a small ECE. These results confirm our intuition about the capacity of ConfidNet to address over-confident predictions, even though it has not been designed for. Nevertheless, dedicated methods such as temperature scaling used in [49] remain preferred for calibrating deep neural networks. Reliability diagrams in Fig. 3.8 illustrates this result with a VGG-16 architecture trained on CIFAR-100.



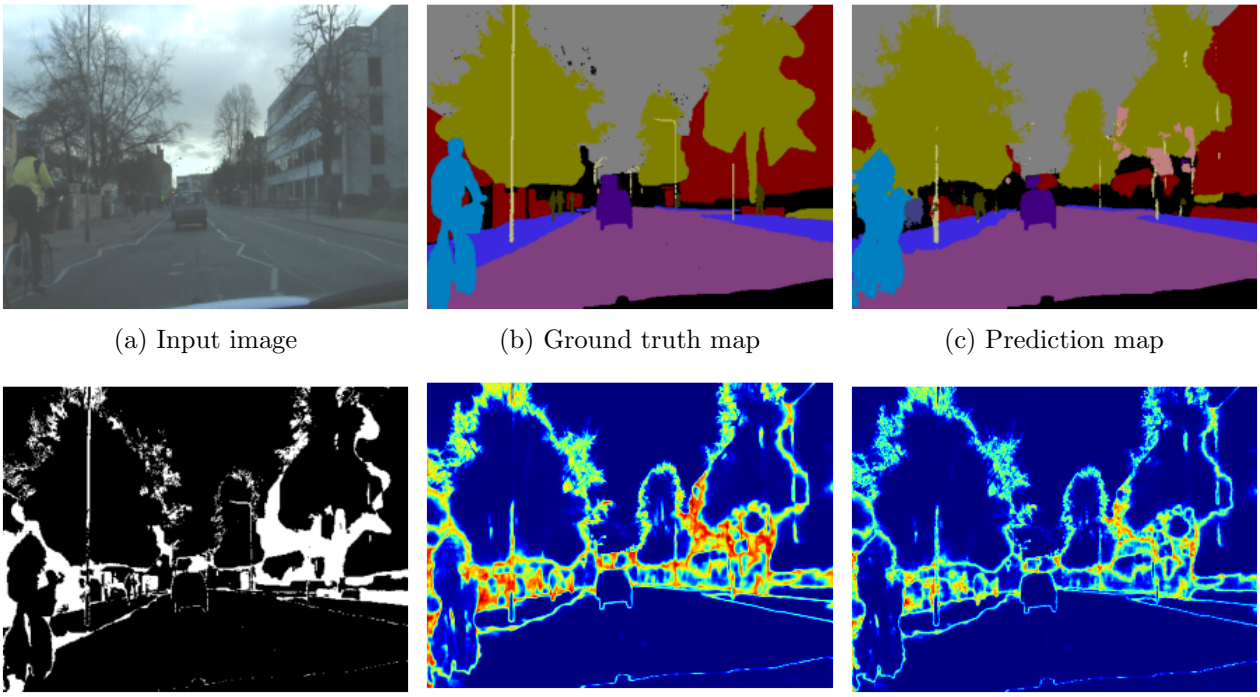
(a) MCP (b) ConfidNet (c) Temperature Scaling

Figure 3.8: **Reliability diagrams for a VGG-16 model on CIFAR-100.** Even though ConfidNet improves calibration over MCP, it remains less effective than dedicated methods for calibration, such as temperature scaling.

3.5.6 Visualisations and failure cases

We provide an illustration on CamVid (Fig. 3.9) to better understand our approach for failure prediction. Compared to MCP baseline, our approach produces higher confidence scores for correct pixel predictions and lower ones on erroneously predicted pixels, which allows a user to better detect error area in semantic segmentation.

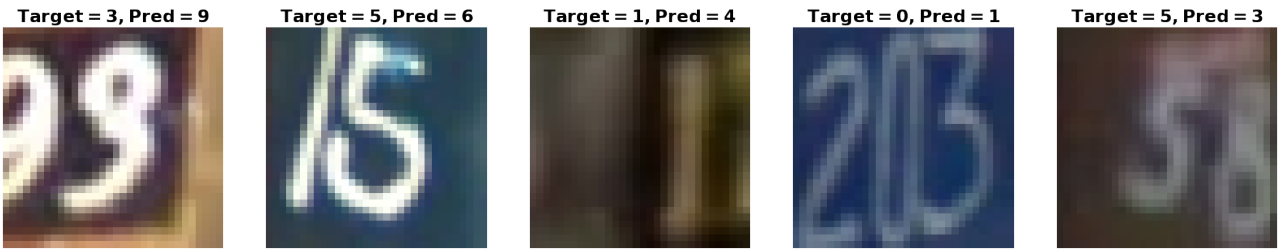
In Fig. 3.10, we present some failures of ConfidNet on the SVHN dataset. On these selected samples, the classifier outputs an erroneous prediction with high MCP, but Confidnet fails to predict the low TCP values, hence to identify these misclassifications. We can observe that these samples are hard to classify, due to low image quality and confusing shapes. For instance, the classifier has assigned Fig. 3.10a to a 9 while the correct label was 3. In this example, even though ConfidNet output remains high (0.74), it manages at least to be significantly lower than the original MCP value (0.93).



(a) Input image (b) Ground truth map (c) Prediction map

(d) Model errors (e) ConfidNet (f) MCP

Figure 3.9: **Visualization of inverse confidence (uncertainty) map for ConfidNet (Fig. 3.9e) and MCP (Fig. 3.9f) on one CamVid scene**. The top row shows the input image (Fig. 3.9a) with its ground-truth (Fig. 3.9b) and the semantic segmentation mask (Fig. 3.9c) predicted by the original classification model. The error map associated with the predicted segmentation is shown in (Fig. 3.9d), with erroneous predictions flagged in white. ConfidNet (55.53% AP-Error) allows a better prediction of these errors than MCP (54.69% AP-Error).



(a) MCP = 0.93 , ConfidNet = 0.74, TCP = 0.06

(b) MCP = 0.94 , ConfidNet = 0.78, TCP = 0.06

(c) MCP = 0.89 , ConfidNet = 0.90, TCP = 0.11

(d) MCP = 0.82 , ConfidNet = 0.91, TCP = 0.17

(e) MCP = 0.90 , ConfidNet = 0.94, TCP = 0.10

Figure 3.10: **Failure cases of ConfidNet**. On these misclassified digits from SVHN's test images, ConfidNet fails to regress the corresponding TCP values, hence to predict the low confidence that should be assigned to the classifier's decisions.

3.6 Conclusion

In this chapter, we defined a new confidence criterion, TCP, which enjoys simple guarantees and empirical evidence of improving the confidence estimation for classifiers with a reject option. We proposed a specific method to learn this criterion with an auxiliary neural network built upon the encoder of the model that is monitored. Applied to failure prediction, this learning scheme consists in training the auxiliary network and then enabling the fine-tuning of its encoder (the one of the monitored classifier remains frozen). In each image classification experiment, we were able to improve the capacity of the model to distinguish correct from erroneous samples and to achieve better selective classification. Besides failure prediction, other applications can benefit from this improved confidence estimation. In the next chapter, we propose a new application of our learned confidence approach related to the task of unsupervised domain adaptation for semantic segmentation using self-training.

Chapter 4

Self-Training with Learned Confidence for Domain Adaptation

Chapter Abstract

Semantic segmentation is a key component for scene understanding with application in self-driving cars and robotics. But collecting and manually annotating urban street scenes with dense pixel-level labels is extremely costly due to the large amount of human effort required. On the other hand, recent advances in computer graphics make it possible to train models on photo-realistic synthetic images with computer-generated annotations. Unsupervised domain adaptation (UDA) aims at learning only from source supervision a well-performing model on target-domain samples.

Self-training has recently proven a potent strategy to improve the effectiveness of UDA in semantic segmentation. This line of work mostly relies on the generation of pseudo-labels over the unannotated target domain to incorporate target-domain images and learn a better segmentation adaptation model. A crucial issue in this context is to base the pseudo-label selection on reliable confidence measures.

In this chapter, we propose to adapt our learning confidence approach with an auxiliary model to estimate the confidence of the segmentation network in its predictions and to use these confidence estimates as a criterion for pseudo-label extraction. We further enforce confidence distribution alignment between source and target domains using adversarial training, and we equip the architecture of the confidence network with multi-scale prediction suitable for semantic segmentation. We show that this strategy produces more accurate pseudo-labels and outperforms strong state-of-the-art baselines at the time of publication on three challenging UDA segmentation benchmarks.

The work described in this chapter is based on the following publication:

- *Charles Corbière, Nicolas Thome, Antoine Saporta, Tuan-Hung Vu, Matthieu Cord, Patrick Pérez. "Confidence Estimation via Auxiliary Models". In IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.*

Contents

4.1 Context

Perception systems in autonomous cars require in-depth understanding of scenes in which they operate. For this reason, semantic segmentation modules are often incorporated to obtain class-label predictions for every scene pixel. While recent advances in deep convolutional networks have significantly improved segmentation performance, their efficacy depends on large quantities of accurately labeled training data. But the labeling process usually requires experts' efforts and the annotation cost limits the operational domains of such systems. On the other hand, a lot of driving scenes data are synthesized by game engines such as GTA5 [152]. Consequently, recent works try to leverage this cheap alternative supervision by training models on these image sources and predicting on real images. But direct transfer is not effective as we observe a drop in performance when evaluating on real images, due to a 'domain gap'.

Unsupervised Domain Adaptation (UDA) is the field of research that aims at reducing this domain gap between source and target domains. In the UDA context, annotated source samples along with some unlabeled target images are available at train time. Most works in this line of research aim at minimizing the distribution discrepancy between the source domain and the target domain, at the feature [153] or prediction level [154, 155], potentially combined with translation methods transforming source images to match the target domain 'style' [156]. Recently, self-training [38, 157, 158] proved its ability to boost adaptation performance significantly. The rationale behind these approaches is to label automatically the most confident target pixels according to current network prediction and to retrain the network accordingly. While this idea is appealing, the presence of noisy or incorrect pseudo-labels could be detrimental to the training of the neural network. As an example, using a ratio of 70% of pseudo-labels in [38] leads to a performance of around 48% mIoU, which is better than 34% with direct transfer (only trained on source domain), but still largely below 63% obtained with the same amount of ground-truth labels. Therefore, defining good measures of confidence to select reliable predictions is of crucial importance towards the development of error-free self-training.

To improve self-training efficiency, we propose to adapt our learning confidence approach developed in the previous chapter for the particular context of unsupervised domain adaptation for semantic segmentation. Using an auxiliary model, we select a pool of pixels with high confidence scores to perform the pseudo-labeling (Section 4.3.1). We propose *ConDA*, a new deep framework for UDA semantic segmentation with self-training. ConDA leverages the general idea of ConfidNet, but includes the following adaptations specifically designed for UDA:

- an adversarial training scheme to prevent drifts in confidence distribution between source and target domains (Section 4.3.2);
- an 'atrous' pyramidal pooling architecture for the confidence network to perform multi-scale confidence estimation (Section 4.3.3).

In Section 4.4, we empirically demonstrate that ConDA brings systematic improvements in performance over self-training based on the standard Maximum Class Probability (MCP), with experiments on various challenging UDA benchmarks with synthetic source datasets and real target datasets and using multiple UDA semantic segmentation methods [38, 155, 159], some of them including multiple modalities (*e.g.* depth [159]).

4.2 Unsupervised domain adaptation

Formally, let us consider the annotated source-domain training set $\mathcal{D}_\mathrm{s} = \{(\boldsymbol{x}_{\mathrm{s},n}, \boldsymbol{y}_{\mathrm{s},n})\}_{n=1}^{N_\mathrm{s}}$, where $\boldsymbol{x}_{\mathrm{s},n}$ is a color image of size (H, W) and $\boldsymbol{y}_{\mathrm{s},n} \in \mathcal{Y}^{H\times W}$ its associated ground-truth segmentation map. A segmentation network F with parameters $\boldsymbol{\theta}$ takes as input an image $\boldsymbol{x}$ and returns a predicted *soft*-segmentation map $F(\boldsymbol{x};\boldsymbol{\theta}) = \mathsf{P}_{\boldsymbol{x}}^{\boldsymbol{\theta}} \in [0,1]^{H\times W\times K}$, where $\mathsf{P}_{\boldsymbol{x}}^{\boldsymbol{\theta}}[h,w,:] = P(Y[h,w] \,|\, \boldsymbol{x};\boldsymbol{\theta}) \in \Delta^{K-1}$, with Δ^{K-1} the probability (K-1)-simplex. The final prediction of the network is the segmentation map $f(\boldsymbol{x})$ defined pixel-wise as $f(\boldsymbol{x})[h,w] = \mathrm{argmax}_{k\in\mathcal{Y}}\, \mathsf{P}_{\boldsymbol{x}}^{\boldsymbol{\theta}}[h,w,k]$. This network is learned over the source domain samples $(\boldsymbol{x}_s, \boldsymbol{y}_s)$ using the cross-entropy segmentation loss:

$$\mathcal{L}_\mathrm{seg}(\boldsymbol{x}_s, \boldsymbol{y}_s) = -\sum_{h=1}^{H}\sum_{w=1}^{W} \log \boldsymbol{P}_{\boldsymbol{x}_s}^{(h,w,k^*)}, \tag{4.1}$$

which is minimized over the parameters θ_F of the network and where k^* is the ground-truth segmentation class for pixel (h,w)[1].

In UDA, the main challenge is to use the unlabeled target set $\mathcal{D}_\mathrm{t} = \{\boldsymbol{x}_{\mathrm{t},n}\}_{n=1}^{N_\mathrm{t}}$ available during training to learn domain-invariant features on which the segmentation model would behave similarly in both domains. Common strategies to perform this task are to minimize the maximum mean discrepancy (MMD) [160], to align the second-order statistics of the distributions (CORAL) [161] or to adopt an adversarial training approach to produce indistinguishable source-target distributions in feature space [153] or output space [154]. For the semantic segmentation task, most recent progresses have been found around the latter. To cite a few methods: CyCADA [156] first stylizes

[1] We omit the location dependence (h,w) on k^* for conciseness.

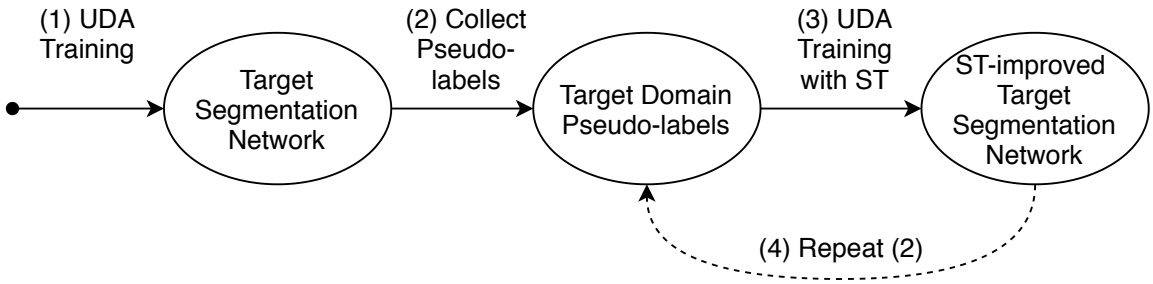


Figure 4.1: **Self-training (ST) for UDA**. A segmentation model is first learned with UDA and used to collect pseudo-labels on target domain images. These automatically annotated data are used to subsequently retrain the model, an operation that can be iterated.

the source images as target-domain images before aligning source and target in the feature space; AdaptSegNet [154] constructs a multi-level adversarial network to perform domain adaptation at different feature levels; AdvEnt [155] aligns the entropy of the pixel-wise predictions with an adversarial loss; BDL [38] learns alternatively an image translation model and a segmentation model that promote each other; DISE [162] disentangles images into domain-invariant structure and domain-specific texture representations, enabling image translation across domains and label transfer to improve segmentation performance.

In the following, we denote $\mathcal{L}_F$ as the objective function of the classifier F, regardless of the method used for UDA. For instance, with adversarial training methods, we would write $\mathcal{L}_F = \mathcal{L}_{\text{seg}} + \mathcal{L}_{\text{adv}}$ where $\mathcal{L}_{\text{adv}}$ is the adversarial term in the objective function.

Self-training Within semi-supervised learning literature [37, 163], self-training with pseudo-labeling showed to be a simple but effective strategy that relies on picking up the current predictions on the unlabeled data and using them as if they were true labels for further training. It is shown in [37] that the effect of pseudo-labeling is equivalent to entropy regularization [163]. In a UDA setting, the idea is to collect pseudo-labels on the unlabeled target-domain samples in order to have an additional supervision loss in the target domain. To select only reliable pseudo-labels, such that the performance of the adapted semantic segmentation network effectively improves, BDL [38] resorts to standard selection with MCP. ESL [164] uses instead the entropy of the prediction as confidence criterion for its pseudo-label selection. CBST [158] proposes an iterative self-training procedure where the pseudo-labels are generated based on a loss minimization. In [158], the authors also propose a way to balance the classes in their pseudo-labels to avoid the dominance of large classes as well as a way to introduce spatial priors. More recently, the CRST framework [157] proposes multiple types of confidence regularization to limit the propagation of errors caused by noisy pseudo-labels.

By attaching pseudo-labels $\hat{\boldsymbol{y}}_t$ to the target-domain images $\boldsymbol{x}_t$, the objective function of F with self-training can be written:

$$\mathcal{L}_{\text{F}}^{*} = \mathcal{L}_{\text{F}} + \frac{\lambda_{\text{ST}}}{|\mathcal{D}_t|} \sum_{\boldsymbol{x}_t \in \mathcal{D}_t} \mathcal{L}_{\text{seg}}(\boldsymbol{x}_t, \hat{\boldsymbol{y}}_t), \tag{4.2}$$

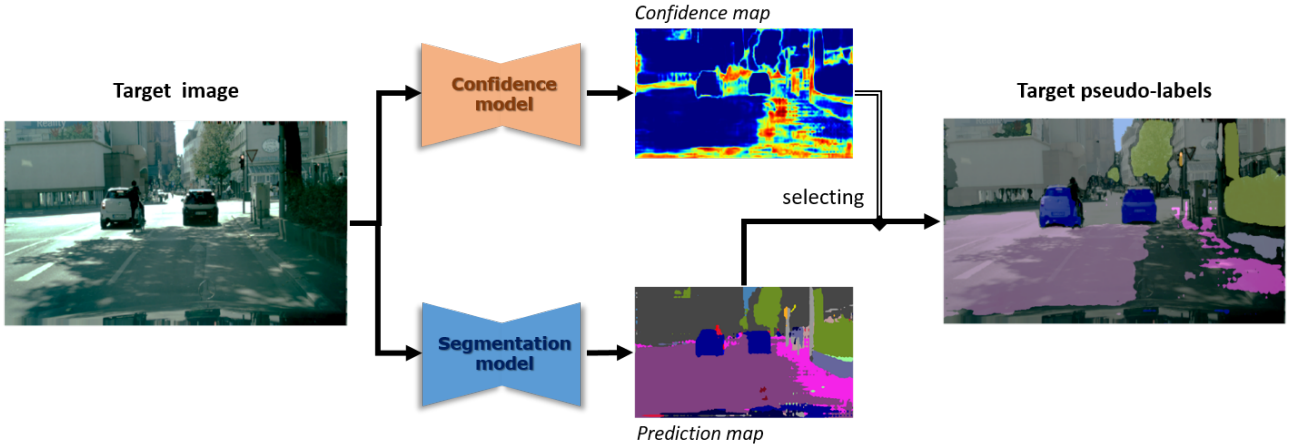


Figure 4.2: **Proposed self-training with learned confidence**. Instead of relying only on the segmentation model to generate pseudo-label maps for target images, we propose to use a confidence model specifically trained to this end. This model outputs a reliable confidence map which helps to improve the quality of the final pseudo-label map.

with a weight λ_{ST} to balance the self-training term.

Self-training in UDA leverages the domain alignment that has already been achieved by the UDA strategy, assuming that the predictions of the current segmentation network F on target domain are relatively accurate. A high-level view of self-training for semantic segmentation with UDA is described in Fig. 4.1:

1. Train a segmentation network for the target domain using a chosen UDA technique;
2. Collect pseudo-labels among the predictions that this network makes on the target-domain training images;
3. Train a new semantic-segmentation network from scratch using the chosen UDA technique in combination with supervised training on target-domain data with pseudo-labels;
4. Possibly, repeat from step 2 by collecting better pseudo-labels after each iteration.

While the general idea of self-training is simple and intuitive, collecting good pseudo-labels is quite tricky. If too many of them correspond to erroneous predictions of the current segmentation network, the performance of the whole UDA can deteriorate. Thus, a measure of confidence should be used in order to only gather reliable predictions as pseudo-labels and to reject the others.

4.3 ConDA: Confidence learning in domain adaptation

Leveraging automatic pseudo-labeling of target-domain training examples is in particular a simple, yet powerful way to further improve UDA performance with self-training. One key ingredient of such

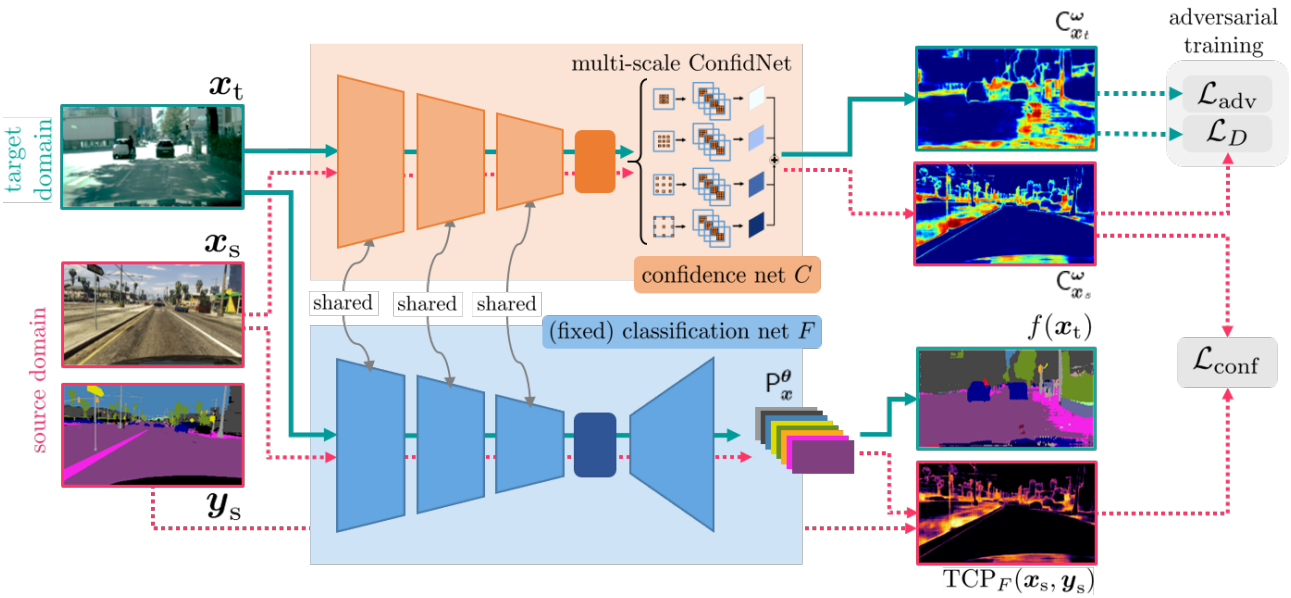


Figure 4.3: **Overview of proposed confidence learning for domain adaptation (ConDA) in semantic segmentation**. Given images in source and target domains, we pass them to the encoder part of the segmentation network F to obtain their feature maps. This network F is fixed during this phase and its weights are not updated. The confidence maps are obtained by feeding these feature maps to the trainable head of the confidence network C, which includes a multi-scale ConfidNet module. For source-domain images, a regression loss $\mathcal{L}_{\text{conf}}$ (Eq. (4.4)) is computed to minimize the distance between $\mathsf{C}^{\boldsymbol{\omega}}_{\boldsymbol{x}_\text{s}}$ and the fixed true-class-probability map $\text{TCP}_F(\boldsymbol{x}_\text{s}, \boldsymbol{y}_\text{s})$. An adversarial training scheme – based on discriminator's loss $\mathcal{L}_D(\boldsymbol{\psi})$ (Eq. (4.6)) and adversarial part $\mathcal{L}_{\text{adv}}(\boldsymbol{\omega})$ of confidence net's loss (Eq. (4.8)) –, is also added to enforce the consistency between the $\mathsf{C}^{\boldsymbol{\omega}}_{\boldsymbol{x}_\text{s}}$'s and $\mathsf{C}^{\boldsymbol{\omega}}_{\boldsymbol{x}_\text{t}}$'s. Dashed arrows stand for paths that are used only at train time.

an approach being the selection of the most promising pseudo-labels, the proposed auxiliary confidence-prediction model lends itself particularly well to this task. In this section, we detail how the proposed approach to confidence prediction can be adapted to semantic segmentation, with application to domain adaptation through self-training.

4.3.1 Selecting pseudo-labels with a confidence model

Following the self-training framework previously described, a confidence network C is learned at step (2) to predict the confidence of the UDA-trained semantic segmentation network F and used to select only trustworthy pseudo-labels on target-domain images as illustrated in Fig. 4.2.

To this end, the framework proposed in Section 3.3 in an image classification setup, and applied to predicting erroneous image classification, needs here to be adapted to the structured output of semantic segmentation, which can be seen as a pixel-wise classification problem. Given a target-domain image $\boldsymbol{x}_\text{t}$, we want to predict both its soft semantic map $F(\boldsymbol{x}_\text{t}; \boldsymbol{\theta})$ and, using an auxiliary

model with trainable parameters $\boldsymbol{\omega}$, its confidence map:

$$C(\boldsymbol{x}_\mathrm{t};\boldsymbol{\omega}) = \mathsf{C}^{\boldsymbol{\omega}}_{\boldsymbol{x}_\mathrm{t}} \in [0,1]^{H\times W}. \tag{4.3}$$

Given a pixel (h,w), if its confidence $\mathsf{C}^{\boldsymbol{\omega}}_{\boldsymbol{x}_\mathrm{t}}[h,w]$ is above a chosen threshold δ, we label it with its predicted class $f(\boldsymbol{x}_\mathrm{t})[h,w] = \mathrm{argmax}_{k\in\mathcal{Y}}\, \mathsf{P}^{\boldsymbol{\theta}}_{\boldsymbol{x}_\mathrm{t}}[h,w,k]$, otherwise it is masked out. Computed over all images in $\mathcal{D}_\mathrm{t}$, these incomplete segmentation maps constitute target pseudo-labels that are used to train a new semantic-segmentation network. Optionally, we may repeat from step (2) and learn alternately a confidence model to collect pseudo-labels and a segmentation network using this self-training.

4.3.2 Confidence training with adversarial loss

To train the confidence network C, we propose to jointly optimize two objectives. Following the approach proposed in Section 3.3, the first one supervises the confidence prediction on annotated source-domain examples using the known true class probabilities for the predictions from F. Specific to semantic segmentation with UDA, the second one is an adversarial loss that aims at reducing the domain gap between source and target. A complete overview of the approach is provided in Fig. 4.3.

Confidence loss. The first objective is a pixel-wise version of the confidence loss Eq. (3.15). On annotated source-domain images, it requires the confidence network C to predict at each pixel the score assigned by the classifier F to the (known) true class:

$$\mathcal{L}_\mathrm{conf}(\boldsymbol{\omega};\mathcal{D}_\mathrm{s}) = \frac{1}{N_\mathrm{s}}\sum_{n=1}^{N_\mathrm{s}} \left\| \mathsf{C}^{\boldsymbol{\omega}}_{\boldsymbol{x}_{\mathrm{s},n}} - \mathrm{TCP}_F(\boldsymbol{x}_{\mathrm{s},n}, \boldsymbol{y}_{\mathrm{s},n}) \right\|^2_\mathrm{F}, \tag{4.4}$$

where $\|\cdot\|_\mathrm{F}$ denotes the Frobenius norm and, for an image $\boldsymbol{x}$ with true segmentation map $\boldsymbol{y}$ and predicted soft one-hot $F(\boldsymbol{x};\hat{\boldsymbol{\theta}})$, we note

$$\mathrm{TCP}_F(\boldsymbol{x},\boldsymbol{y})[h,w] = F(\boldsymbol{x};\hat{\boldsymbol{\theta}})\Big[h,w,\boldsymbol{y}[h,w]\Big] \tag{4.5}$$

at location (h,w). On a new input image, C should predict at each pixel the score that F will assign to the unknown true class, which will serve as a confidence measure.

However, compared to the application in the previous chapter, we have here the additional problem of the gap between source and target domains, an issue that might affect the training of the confidence model as in the training of the segmentation model.

Adversarial loss. The second objective concerns the domain gap. While the confidence network C learns to estimate TCP on source-domain images, its confidence estimation on target-domain images may suffer dramatically from this domain shift. As classically done in UDA, we propose an adversarial learning of our auxiliary model in order to address this problem. More precisely, we want the confidence maps produced by C in the source domain to resemble those obtained in the target domain.

A discriminator $D : [0,1]^{H \times W} \rightarrow \{0,1\}$, with parameters $\boldsymbol{\psi}$, is trained concurrently with C with the aim to recognize the domain (1 for source, 0 for target) of an image given its confidence map. The following loss is minimized w.r.t. $\boldsymbol{\psi}$:

$$\mathcal{L}_D(\boldsymbol{\psi}; \mathcal{D}_\mathrm{s} \cup \mathcal{D}_\mathrm{t}) = \frac{1}{N_\mathrm{s}} \sum_{n=1}^{N_\mathrm{s}} \mathcal{L}_\mathrm{adv}(\boldsymbol{x}_{\mathrm{s},n}, 1) + \frac{1}{N_\mathrm{t}} \sum_{n=1}^{N_\mathrm{t}} \mathcal{L}_\mathrm{adv}(\boldsymbol{x}_{\mathrm{t},n}, 0), \tag{4.6}$$

where $\mathcal{L}_\mathrm{adv}$ denotes the cross-entropy loss of the discriminator based on confidence maps:

$$\mathcal{L}_\mathrm{adv}(\boldsymbol{x}, \lambda) = -\lambda \log \left(D(\mathsf{C}^{\boldsymbol{\omega}}_{\boldsymbol{x}}; \boldsymbol{\psi})\right) - (1-\lambda) \log(1 - D(\mathsf{C}^{\boldsymbol{\omega}}_{\boldsymbol{x}}; \boldsymbol{\psi})), \tag{4.7}$$

for $\lambda \in \{0,1\}$, which is a function of both $\boldsymbol{\psi}$ and $\boldsymbol{\omega}$. In alternation with the training of the discriminator using Eq. (4.6), the adversarial training of the confidence net is conducted by minimizing, w.r.t. $\boldsymbol{\omega}$, the following loss:

$$\mathcal{L}_C(\boldsymbol{\omega}; \mathcal{D}_\mathrm{s} \cup \mathcal{D}_\mathrm{t}) = \mathcal{L}_\mathrm{conf}(\boldsymbol{\omega}; \mathcal{D}_\mathrm{s}) + \frac{\lambda_\mathrm{adv}}{N_\mathrm{t}} \sum_{n=1}^{N_\mathrm{t}} \mathcal{L}_\mathrm{adv}(\boldsymbol{x}_\mathrm{t}, 1), \tag{4.8}$$

where the second term, weighted by $\lambda_\mathrm{adv} > 0$, encourages C to produce maps in the target domain that will confuse the discriminator.

This proposed adversarial confidence learning scheme also acts as a regularizer during training, improving robustness of the unknown TCP target confidence. As the training of the confidence model may actually be unstable, adversarial training provides additional information signal, in particular imposing that confidence estimation should be invariant to domain shifts. We empirically observe that this adversarial confidence learning provides better confidence estimates and improves convergence and stability of the training scheme.

4.3.3 Multi-scale ConfidNet architecture

In semantic segmentation, models consist of fully convolutional networks where hidden representations are 2D feature maps. This is in contrast with the architecture of classification models considered in Chapter 3. Consequently, we replace fully-connected layers in the ConfidNet module by 1×1 convolutional layers with the adequate number of channels.

In many segmentation datasets, the existence of objects at multiple scales may complicate confidence estimation. As in recent works dealing with varying object sizes [7], we further improve our confidence network C by adding a multi-scale architecture based on spatial pyramid pooling. It consists of a computationally efficient scheme to re-sample a feature map at different scales, and then to aggregate the confidence maps. We illustrate the multi-scale architecture for a confidence network in Fig. 4.4.

From a feature map, we apply parallel atrous convolutional layers with 3×3 kernel size and different sampling rates, each of them followed by a series of 4 standard convolutional layers with 3×3 kernel size. In contrast with convolutional layers with large kernels, atrous convolution layers enlarge the field of view of filters and help to incorporate a larger context without increasing the number of parameters and the computation time. Resulting features are then summed before upsampling to the original

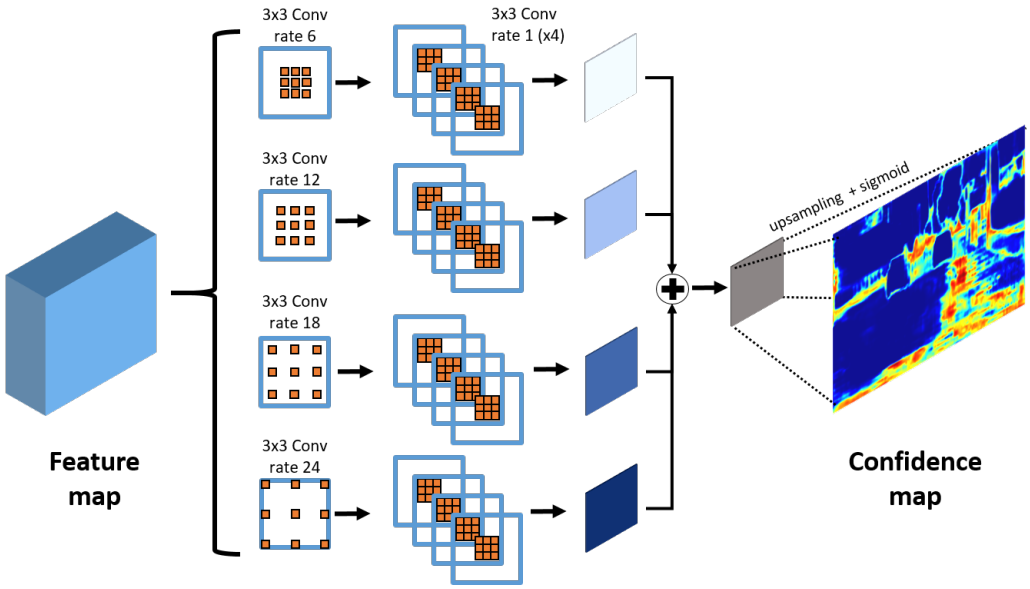


Figure 4.4: **Multi-scale architecture for confidence learning**. Four atrous convolutional layers are applied to a feature map in parallel, each of them is followed by a series of four standard convolutional layers. Confidence maps are obtained by summing the resulting features and upsampling to original image size.

image size of $H \times W$. We apply a final sigmoid activation to output a confidence map with values between 0 and 1.

The whole architecture of the confidence model C is represented in the orange block of Fig. 4.3, along with its training given a fixed segmentation model F (blue block) with which it shares the encoder. Such as in the previous section, fine-tuning the encoder within C is also possible, although we did not explore the option in this semantic segmentation context due to the excessive memory overhead it implies.

4.4 Experiments

In this section, we analyse on several semantic segmentation benchmarks the performance of ConDA, our approach to domain adaptation with confidence-based self-training. We report comparisons with state-of-the-art methods at the time of publication on each benchmark at the time of publication. We also analyse further the quality of ConDA's pseudo-labelling and demonstrate via an ablation study the importance of each of its components.

4.4.1 Experimental setup

Datasets As in many domain adaptation works for semantic segmentation, we consider the specific task of adapting from synthetic to real data in urban scenes. We experiment with two synthetic

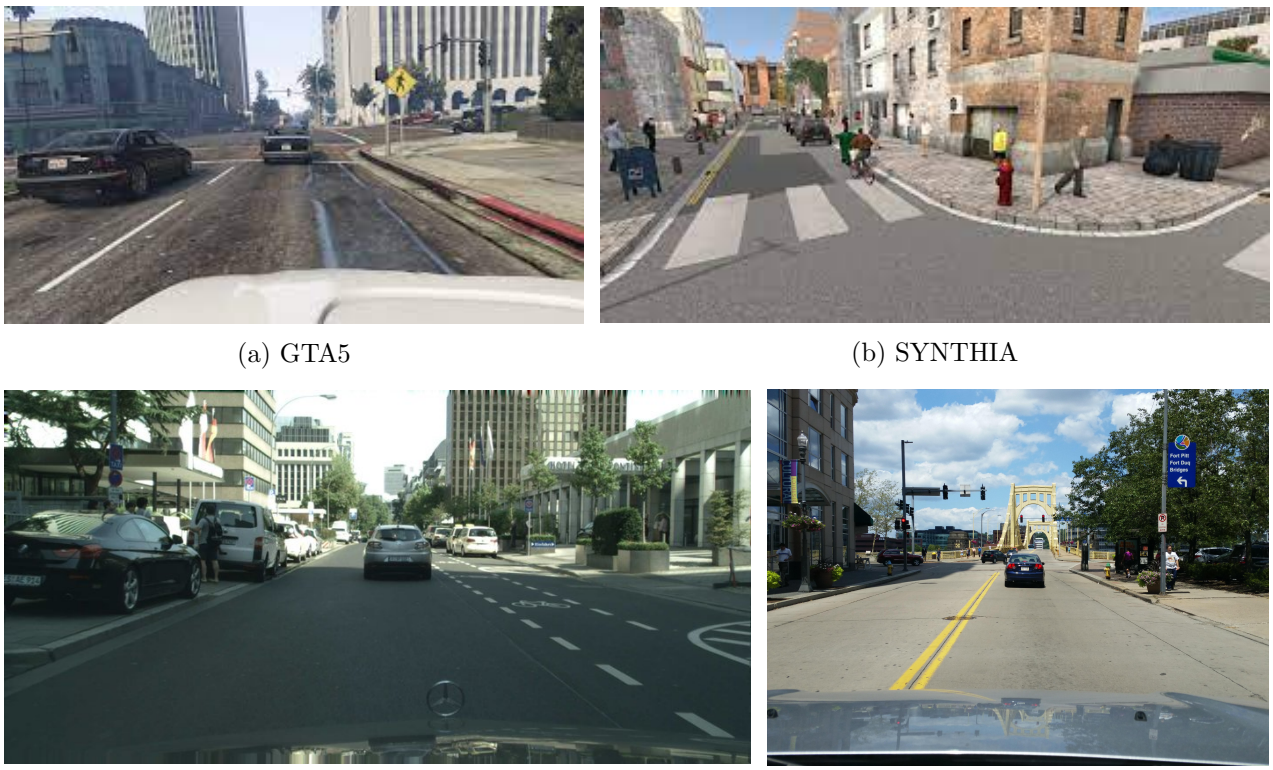
source datasets – SYNTHIA [165] and GTA5 [152] – and two real target datasets – Cityscapes [77] and Mapillary Vistas [14]. More specifically, we use the SYNTHIA-RAND-CITYSCAPES split for SYNTHIA [165], composed of 9,400 color images generated in a simulator, of dimension 1280×760 and annotated for semantic segmentation with 16 classes in common with Cityscapes [77]. As for GTA5 [152], the dataset is composed of 24,966 images extracted from the eponymous game, of dimension 1914×1052, with semantic segmentation annotation with 19 classes in common with Cityscapes [77]. On the other hand, Cityscapes [77] is a dataset of real street-level images. It is split in a training set, a validation set and a test set. For domain adaptation, we use the training set as the target dataset during training. It is composed of 2,975 images of dimension 2048×1024. Since the ground-truth segmentation maps are missing from the testing dataset, we exploit the validation set composed of 500 images for testing purposes. We also validate the approach on Mapillary Vistas [14], another dataset of street-level images. As Cityscapes [77], it is split in a train set, a validation set and a test set, and the ground-truth maps are missing from the testing dataset. For domain adaptation, we use the 18,000 images from the training set as target and the 2,000 images from the validation set for testing. On Mapillary Vistas [14] experiments, we consider 7 'super classes' that include the 19 and 16 classes used in Cityscapes [77] experiments with GTA5 [152] and SYNTHIA [165], respectively. All results are reported in terms of the mean intersection over union (mIoU) metric. The higher this percentage, the better.

Network architectures. We evaluate the proposed self-training method on three state-of-the-art domain adaptation architectures at the time of publication. They all are based on DeepLabV2 [7], a standard semantic segmentation network. The domain alignment modules are nevertheless different:

- *AdaptSegNet* [154] performs adversarial domain adaptation on the output of the semantic segmentation network to align directly the segmentation maps between source and target domains.
- *AdvEnt* [155] proposes another adversarial learning framework for domain adaptation: instead of the softmax output prediction, AdvEnt aligns the entropy of the pixel-wise predictions.
- *DADA* [159] uses depth information on source images as privileged information during segmentation training.

Implementation details The semantic segmentation models are initialized with backbones pretrained on ImageNet [4]. The segmentation network is optimized by Stochastic Gradient Descent with learning rate 2.5×10^{-4}, momentum 0.9 and weight decay 10^{-4}. As for the discriminator, it is optimized by Adam [73] with learning rate 10^{-4}. The hyperparameters λ_{adv} and λ_{ST} are fixed at 10^{-3} and 1, respectively. For each baseline model, we start our self-training procedure from the pre-trained weights given on the author's GitHub[2]. In some experiments, we use translated source images into the target domain. Those translated images are pre-computed using a CycleGAN [166], as provided by [38].

[2] https://github.com/valeoai/ADVENT, https://github.com/liyunsheng13/BDL



(a) GTA5 (b) SYNTHIA

(c) Cityscapes (d) Mapillary Vistas

Figure 4.5: **Images sample from datasets used in UDA experiments.** Synthetic datasets used as source domains are GTA5-dataset (a) and SYNTHIA (b); real datasets used as target domains are Cityscapes (c) and Mapillary Vistas (d).

Regarding confidence training, we use the same training dataset than in segmentation training. As in Chapter 3, hyperparameters are chosen based on validation-set AUPR.

4.4.2 ConDA vs. MCP self-training

We compare the adaptation results using MCP to collect pseudo-labels and using ConDA instead, on two different methods: AdaptSegNet [154] and AdvEnt [155]. Results are available in Table 4.1. On GTA5 ▷ Cityscapes benchmark, we can see that, for both method, ConDA improves over MCP in self-training framework for domain adaptation by adding respectively +1.0 point mIoU improvement on AdaptSegNet and +0.9 point mIoU improvement on AdvEnt, which is our best result on this benchmark. We also compare the best adaptation method (Advent) on two other datasets: as shown in Table 4.1, we observe a systematic improvement over MCP: +0.5 point mIoU on SYNTHIA ▷ Cityscapes and +1.3 point mIoU on 'SYNTHIA ▷ Mapillary'. Finally, we also extend to methods dealing with multiple modalities (*e.g.* depth) such as DADA [159] and we observe similarly that ConDA outperforms MCP by +1.1 point. These results demonstrate the relevance of our method by selecting better pseudo-labels to improve adaptation, regardless of the segmentation method or the

Table 4.1: **Comparison on mIoU of MCP-based vs. ConDA-based self-training** over multiple architectures and benchmarks. For DADA* architecture, segmentation models are trained using depth as privileged information.

	GTA5 ▷ Cityscapes		SYNTHIA ▷ Cityscapes	SYNTHIA ▷ Mapillary	
Self-Training	AdaptSegNet	AdvEnt	AdvEnt	AdvEnt	DADA*
MCP	48.3	49.0	45.5	65.1	70.9
ConDA	**49.3**	**49.9**	**46.0**	**66.4**	**72.0**

UDA segmentation benchmark used.

4.4.3 Comparison with UDA baselines

In this section, ConDA results correspond to applying our self-training approach on AdvEnt domain adaptation method.

GTA5 ▷ Cityscapes. Results for semantic segmentation on the Cityscapes validation set using GTA5 as source domain are available in Table 4.2 in the following page. We first notice that self-training based methods from the literature improved performance on this benchmark up to 48.5% mIoU with BDL [38]. ConDA outperforms all those methods on this framework by reaching 49.9% mIoU. Note also that combining AdvEnt with MCP self-training already achieved 49.0% mIoU.

SYNTHIA ▷ Cityscapes. We extend experiments by using another source domain dataset. We report in a consistent way adaptation results for the task 'SYNTHIA ▷ Cityscapes' in Table 4.3. Following relevant literature on this dataset, mIoU results for 16 categories and for 13 categories are available. Again, ConDA achieves state-of-the-art performance on this benchmark at the time of publication with 46.0% mIoU .

SYNTHIA ▷ Mapillary. Along with results on Cityscapes, we further study domain adaptation on another target dataset, namely Mapillary Vistas. Table 4.4 presents semantic segmentation performance using SYNTHIA as source dataset. This benchmark has also been used in other recent works, such as in AdvEnt [155] and DADA [159]. ConDA outperforms the baseline method with 66.4% mIoU compared to 65.2% mIoU in AdvEnt. When using depth from SYNTHIA as privileged information such as in DADA, proposed method (ConDA*) further increases performance from 67.6% mIoU to 72.0% mIoU.

Table 4.2: **Comparative performance on semantic segmentation with synth-to-real unsupervised domain adaptation.** Results in per-class IoU and class-averaged mIoU on GTA5 ▷ Cityscapes. All methods are based on a DeepLabv2 backbone.

		GTA5 ▷ Cityscapes																			
Method	Self-Train.	road	sidewalk	building	wall	fence	pole	light	sign	veg	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIoU
AdaptSegNet [154]		86.5	25.9	79.8	22.1	20.0	23.6	33.1	21.8	81.8	25.9	75.9	57.3	26.2	76.3	29.8	32.1	7.2	**29.5**	32.5	41.4
CyCADA [156]		86.7	35.6	80.1	19.8	17.5	**38.0**	**39.9**	**41.5**	82.7	27.9	73.6	**64.9**	19.0	65.0	12.0	28.6	4.5	31.1	42.0	42.7
DISE [162]		91.5	47.5	82.5	31.3	25.6	33.0	33.7	25.8	82.7	28.8	82.7	62.4	30.8	85.2	27.7	34.5	6.4	25.2	24.4	45.4
AdvEnt [155]		89.4	33.1	81.0	26.6	26.8	27.2	33.5	24.7	83.9	36.7	78.8	58.7	30.5	84.8	38.5	44.5	1.7	31.6	32.4	45.5
CBST [158]	✓	91.8	53.5	80.5	32.7	21.0	34.0	28.9	20.4	83.9	34.2	80.9	53.1	24.0	82.7	30.3	35.9	16.0	25.9	**42.8**	45.9
MRKLD [157]	✓	91.0	55.4	80.0	33.7	21.4	37.3	32.9	24.5	85.0	34.1	80.8	57.7	24.6	84.1	27.8	30.1	**26.9**	26.0	42.3	47.1
BDL [38]	✓	91.0	44.7	84.2	34.6	**27.5**	30.2	36.0	36.0	85.0	**43.6**	83.0	58.6	**31.6**	83.3	35.3	49.7	3.3	28.8	35.6	48.5
ESL [164]	✓	90.2	43.9	84.7	35.9	28.5	31.2	37.9	34.0	84.5	42.2	83.9	59.0	32.2	81.8	36.7	49.4	1.8	30.6	34.1	48.6
ConDA	✓	**93.5**	**56.9**	**85.3**	**38.6**	26.1	34.3	36.9	29.9	**85.3**	40.6	**88.3**	58.1	30.3	**85.8**	**39.8**	**51.0**	0.0	28.9	37.8	**49.9**

Table 4.3: **Comparison in mIoU for SYNTHIA ▷ Cityscapes experiments**. mIoU* is the 13-class setup (excluding the classes 'wall', 'fence' and 'pole') as used in earlier works. All methods reported are based on a DeepLabv2 backbone.

		SYNTHIA ▷ Cityscapes																
Method	Self-Train.	road	sidewalk	building	wall	fence	pole	light	sign	veg	sky	person	rider	car	bus	mbike	bike	mIoU	mIoU*
AdaptSegNet [154]		84.3	42.7	77.5	-	-	-	4.7	7.0	77.9	82.5	54.3	21.0	72.3	32.2	18.9	32.3	-	46.7
DISE [162]		**91.7**	**53.5**	77.1	2.5	0.2	27.1	6.2	7.6	78.4	81.2	55.8	19.2	82.3	30.3	17.1	34.3	41.5	48.8
AdvEnt [155]		85.6	42.2	79.7	8.7	0.4	25.9	5.4	8.1	80.4	84.1	57.9	23.8	73.3	36.4	14.2	33.0	41.2	48.0
CBST [158]	✓	68.0	29.9	76.3	10.8	1.4	33.9	**22.8**	29.5	77.6	78.3	60.6	28.3	81.6	23.5	18.8	39.8	42.6	48.9
MRKLD [157]	✓	67.7	32.2	73.9	10.7	**1.6**	**37.4**	22.2	**31.2**	80.8	80.5	**60.8**	**29.1**	**82.8**	25.0	19.4	45.3	43.8	50.1
BDL [38]	✓	83.9	43.7	80.2	**12.9**	0.5	30.1	18.0	17.3	79.7	83.5	52.2	25.8	72.5	35.5	**25.8**	45.4	44.2	51.0
ESL [164]	✓	84.3	39.7	79.0	9.4	0.7	27.7	16.0	14.3	78.3	83.8	59.1	26.6	72.7	35.8	23.6	45.8	43.5	50.7
ConDA (Ours)	✓	88.1	46.7	**81.1**	10.6	1.1	31.3	22.6	19.6	**81.3**	**84.3**	53.9	21.7	79.8	**42.9**	24.2	**46.8**	**46.0**	**53.3**

Table 4.4: **Comparison in mIoU for SYNTHIA ▷ Mapillary experiments**. DADA* and ConDA* are trained using depth as privileged information.

SYNTHIA ▷ Mapillary									
Method	Self-Train.	flat	constr.	object	nature	sky	human	vehicle	mIoU
AdvEnt [155]		86.9	58.8	30.5	74.1	85.1	48.3	72.5	65.2
ESL [164]	✓	88.4	55.7	**32.0**	**75.4**	84.3	43.5	**76.2**	65.4
ConDA (Ours)	✓	**89.1**	**63.5**	28.3	72.7	**88.2**	**49.7**	73.0	**66.4**
DADA* [159]		86.7	62.1	34.9	75.9	88.6	51.1	73.8	67.6
ConDA* (Ours)	✓	**87.8**	**67.5**	**40.5**	**76.8**	**92.3**	**60.7**	**78.5**	**72.0**

4.4.4 Ablation study

To study the effect of the adversarial training and of the multi-scale confidence architecture on the confidence model, we perform an ablation study on the GTA5 ▷ Cityscapes benchmark. The results on domain adaptation after re-training the segmentation network using collected pseudo-labels are reported in Table 4.5. In this table, "ConfidNet" refers to the simple network architecture defined in Section 3.3 (adapted to segmentation by replacing the fully connected layers by 1×1 convolutions of suitable width); "Adv. ConfidNet" denotes the same architecture but with the adversarial loss from Section 4.3.2 added to its learning scheme; "Multi-scale ConfidNet" stands for the architecture introduced in Section 4.3.3; Finally, the full method, "ConDA" amounts to having both this architecture and the adversarial loss. We notice that adding the adversarial learning achieves significantly better performance, for both ConfidNet and multi-scale ConfidNet, with respectively +1.4 and +0.8 point increase. Multi-scale ConfidNet (resp. adv. multi-Scale ConfidNet) also improves performance up to +0.9 point (resp. +0.3) from their ConfidNet architecture counterpart. These results stress the importance of both components of the proposed confidence model.

Model	Multi-Scale.	Adv	mIoU
ConfidNet			47.6
Multi-Scale ConfidNet	✓		48.5
Adv. ConfidNet		✓	49.0
ConDA (Adv. Multi-scale ConfidNet)	✓	✓	**49.9**

Table 4.5: **Ablation study on semantic segmentation with pseudo-labelling-based adaptation.** Full-fledged ConDA approach is compared on GTA5 ▷ Cityscapes to stripped-down variants (with/without multi-scale architecture in ConfidNet, with/without adversarial learning).

4.4.5 Quality of pseudo-labels

We analyze the effectiveness of MCP and ConDA as confidence measures to select relevant pseudo-labels in the target domain. For a given fraction of retained pseudo-labels (coverage) on target-domain training images, we compare in Fig. 4.6 the precision of each method. Here, precision means the ratio between the number of correct predictions and the total number of collected pseudo-labels, *i.e.* accuracy. We vary the coverage between 70% and 90%[3].

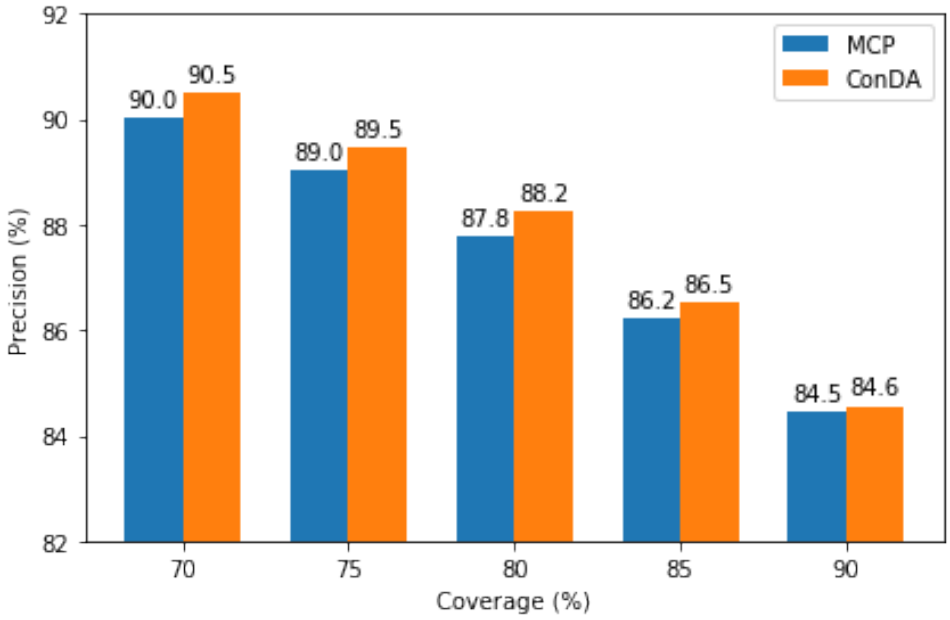


Figure 4.6: **Comparative quality of selected pseudo-labels**. Proportion of correct pseudo-labels (precision) for different coverages on GTA5 ▷ Cityscapes, for MCP and ConDA.

ConDA outperforms MCP for all coverage levels, meaning it selects significantly fewer erroneous predictions for the next round of segmentation-model training. Along with the segmentation adaptation improvements presented earlier, these coverage results demonstrate that reducing the amount of noise in the pseudo-labels is key to learning a better segmentation adaptation model. Fig. 4.7 presents qualitative results of those pseudo-labels methods. We find again that MCP and ConDA seem to select around the same amount of correct predictions in their pseudo-labels, but with ConDA picking out a lot fewer erroneous ones.

Computational Cost. Composed of only four atrous convolutional layers in parallel, our multi-scale confidence network remains light. When collecting pseudo-labels, the overhead cost induced by our method is minor when estimating confidence, only adding roughly 10% time increase compared to MCP. In our setting, a forward pass using ConDA takes 43ms on average. Note that segmentation

[3]For instance, in our previous experiment with AdvEnt on 'GTA5 ▷ Cityscapes', 80.5% pixels were kept using MCP and 82.0% using ConDA.

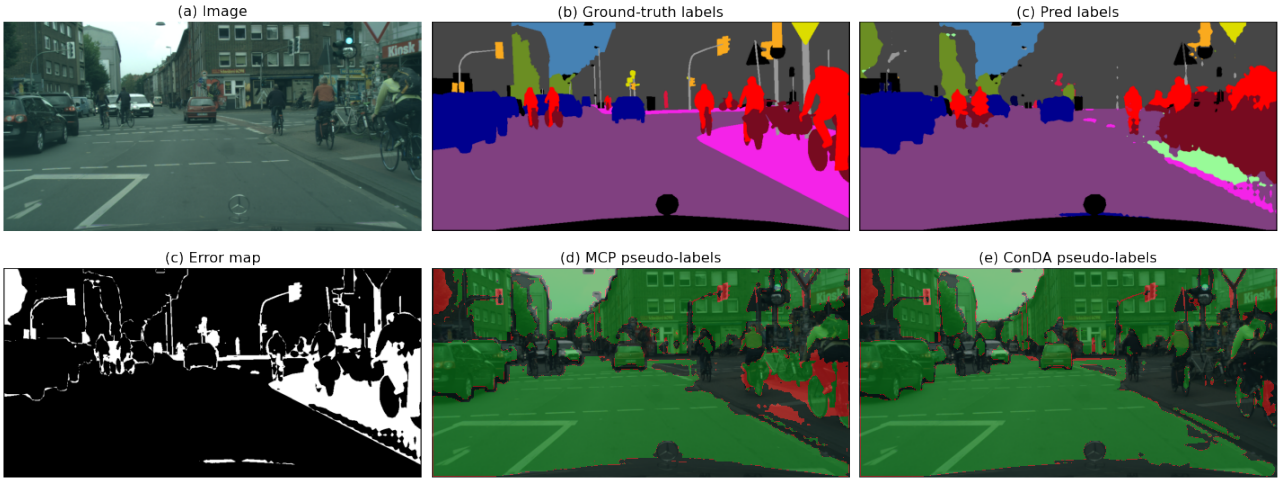


Figure 4.7: **Qualitative result of pseudo-label selection for semantic-segmentation adaptation.** The three first panels present a target-domain image of the GTA5 ▷ Cityscapes benchmark (a) along with its ground-truth segmentation maps (b) and the predicted map before self-training (c). The error map associated with the predicted segmentation is shown in (d), with erroneous predictions flagged in white. We compare pseudo-labels collected with MCP (e) and with ConDA (f). Green (resp. red) pixels are correct (resp. erroneous) predictions selected by the method and black pixels are discarded predictions. ConDA retains fewer errors while preserving approximately the same amount of correct predictions.

training and inference are not changed, which makes ConDA suitable for real-time purposes.

4.5 Conclusion

In this chapter, we show that applied to self-training with pseudo-labels, using an auxiliary model dedicated to estimate the confidence of predictions help to better select relevant pixels for pseudo-labeling. Our learning approach brings systematic improvements in performance over self-training based on the standard MCP. We reach state-of-the-art results at the time of publication on three synthetic-to-real unsupervised-domain-adaptation benchmarks (GTA5 ▷ Cityscapes, SYNTHIA ▷Cityscapes and SYNTHIA ▷Mapillary Vistas). To achieve these results, we equipped the auxiliary model with a multi-scale confidence architecture and supplemented the confidence loss with an adversarial training scheme to enforce alignment between confidence maps in source and target domains. In particular, a clear benefit of our learning approach is to be compatible with any models which use domain adaptation, without adding substantial overhead cost (only 10% time increase compared to MCP in our experiments).

Thus far, we focused on detecting errors to reject them or to alternatively select only correct predictions. However, in the wild, a ML system may also encounter data that is unlike its training data. In addition to in-distribution errors, we will also consider the detection of out-of-distribution samples in the next chapter to be robust to any kind of hazardous predictions.

Chapter 5

Simultaneous Detection of Misclassifications and Out-of-Distribution Samples with Evidential Models

Chapter Abstract

A safe deployment of ML systems should include an accurate monitoring of errors but also unusual inputs where predicting might be hazardous. In this chapter, we address the task of jointly detecting errors and anomalies in classification tasks. Evidential models are a Bayesian approach which provides a sampling-free way of deriving second-order uncertainty measures on the simplex, i.e. measures on the distribution over probabilities, that estimate different sources of uncertainty. In this chapter, we leverage the second-order representation provided by evidential models and we introduce KLoS*, a Kullback–Leibler divergence criterion defined on the class-probability simplex. By keeping the full distributional information, KLoS captures in-distribution and out-of-distribution (OOD) uncertainties in a single score. A crucial property of KLoS is to be a class-wise divergence measure built from in-distribution samples and to not require OOD training data, in contrast to current uncertainty measure used with evidential models. We further design an auxiliary neural network,* KLoSNet*, to learn a refined criterion directly aligned with the evidential training objective. In the realistic context where no OOD data is available during training, our experiments show that KLoSNet outperforms every other uncertainty measures to simultaneously detect misclassifications and OOD samples. When training with OOD samples, we also observe that existing measures are brittle to the choice of the OOD dataset, whereas KLoS remains more robust.*

The work described in this chapter is based on the following publication:

- *Charles Corbière, Marc Lafon, Nicolas Thome, Matthieu Cord, Patrick Pérez. "Beyond First-Order Estimation with Evidential Models for Open-World Recognition". ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning.*

Contents

5.1 Context

Machine learning models commonly rely on the closed-set assumption that source and target data are independent and identically distributed (*i.i.d.*). Yet in practice, distribution shifts arise naturally in many real-world scenarios. For instance, self-driving cars struggle to perform well under conditions different to those of training, such as variations in weather [167], light [168], and object poses [169]. Worse, models can be exposed to inputs from unseen classes which they will attempt to predict anyway. These failures may remain unnoticed as they do not result in explicit errors in the model.

While previous works address separately misclassification detection and OOD detection, we argue it is necessary for a recognition system to be able to identify both in-distribution misclassifications and unknown/unseen inputs at test time for a safe deployment in open-world settings [110]. We illustrate this task in Fig. 5.1. In particular, we find in Section 5.5 that all previous approaches do not perform equally well on both detection tasks, which mitigates their ability on the joint detection task.

To address the task of simultaneous detection of misclassifications and OOD samples, an uncertainty measure should discriminate between correct predictions and erroneous predictions for in-distribution samples while increasing for inputs far from distribution. Consequently, it should capture both aleatoric and epistemic uncertainty. Bayesian approaches [40, 52] and ensembles [53, 54] are principled methods which induce a more accurate estimation of epistemic uncertainty. These techniques produce a probability density over the predictive categorical distribution $p(y|\boldsymbol{x}, \mathcal{D})$ obtained from sampling as shown in Chapter 2 (top row of Fig. 2.7). But this comes at the expense of an increased computational cost.

A recent class of models, coined *evidential* [57, 56], proposes instead to explicitly learn the concentration parameters of a Dirichlet distribution over probabilities. Based on the subjective logic framework [96], evidential models enrich uncertainty representation with evidence information and enable a model to represent different sources of uncertainty (bottom row of Fig. 2.7). *Conflicting*

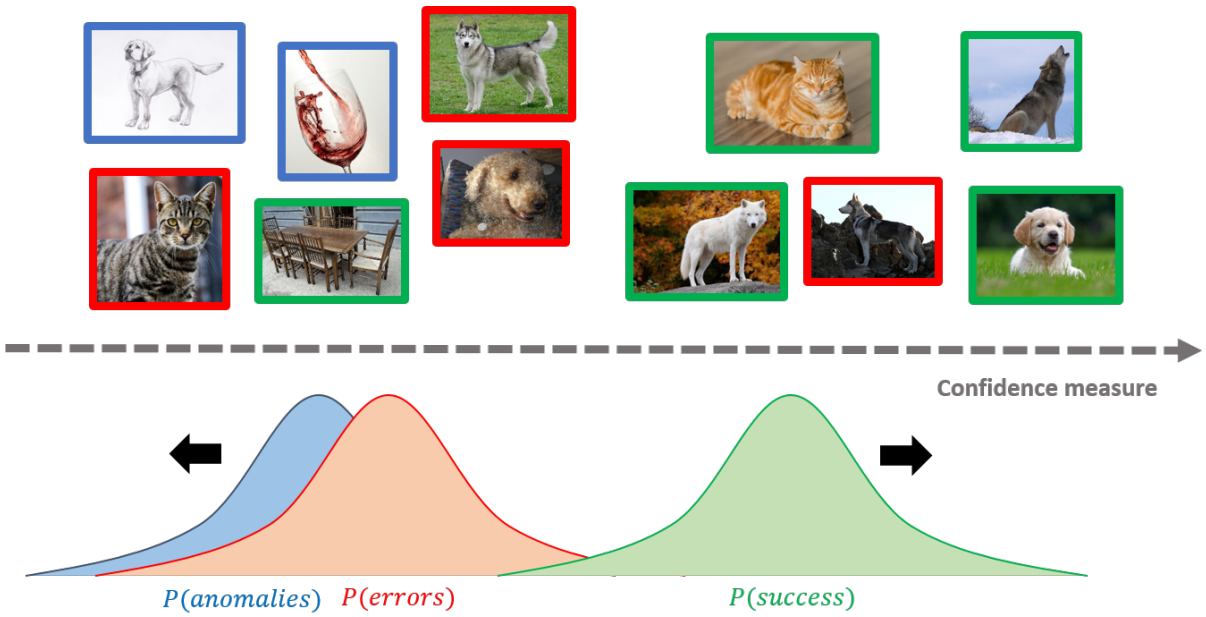


Figure 5.1: **Simultaneous detection of misclassifications and OOD samples.** When ranking samples according to their confidence value, correct predictions (in green) should have higher scores on average than misclassifications (in red) *and* OOD samples (in blue) to enable the model to distinguish them.

evidence, *e.g.*, noise or class confusion, is characterized by the expectation of the second-order Dirichlet distribution while the distribution spread on the simplex expresses the *amount of evidence* in a prediction [101]. These sources of uncertainty correspond respectively to aleatoric uncertainty and epistemic uncertainty in the machine learning literature [135]. Evidential models have been shown to improve generalisation [170], OOD detection [171] and adversarial attack detection [135].

In this chapter, we leverage the second-order uncertainty representation that evidential models provide and we introduce *KLoS*, a measure that accounts for both in-distribution and out-of-distribution sources of uncertainty and that is effective even without having access to auxiliary OOD data at train time. KLoS computes the Kullback–Leibler (KL) divergence between the model's predicted Dirichlet distribution and a specifically designed class-wise prototype Dirichlet distribution. Prototype distributions are designed with concentration parameters shared with in-distribution training data, which enables a model to detect OOD samples without assuming any restrictive behavior, *e.g.*, having low precision α_0. KLoS naturally reflects the training objective used in evidential models and we propose to learn an auxiliary model, named *KLoSNet*, to regress the values of a refined objective for training samples and to improve uncertainty estimation. To assess the quality of uncertainty estimates in open-world recognition, we design the new task of simultaneous detection of misclassifications and OOD samples. Extensive experiments show the benefits of KLoSNet on various image datasets and model architectures. In presence of OOD training data, we also found that our proposed measure is more robust to the choice of OOD samples while previous measures may perform poorly. Finally, we show that KLoS can be successfully combined with ensembling to improve performance.

5.2 Evidential neural networks

In this section, we partially remind the reader the framework of evidential models introduced in Chapter 2 and focus on the derivation of the training objective to put in perspective the link with our proposed uncertainty measure afterwards. The training dataset $\mathcal{D}$ consists of N *i.i.d.* samples $(\boldsymbol{x}, y)$ drawn from an unknown joint distribution $P(X, Y)$. We denote $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_K)$ the random variable over categorical probabilities, where $\sum_{k=1}^{K} \pi_k = 1$, and which lives on the (K-1)-dimensional simplex $\triangle^{K-1}$.

Evidential Neural Networks (ENNs) propose to model explicitly the posterior distribution over categorical probabilities $p(\boldsymbol{\pi}|\boldsymbol{x}, y)$ by a variational Dirichlet distribution,

$$q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{\theta})) = \frac{\Gamma(\alpha_0(\boldsymbol{x}, \boldsymbol{\theta}))}{\prod_{k=1}^{K} \Gamma(\alpha_k(\boldsymbol{x}, \boldsymbol{\theta}))} \prod_{k=1}^{K} \pi_k^{\alpha_k(\boldsymbol{x}, \boldsymbol{\theta})-1}, \tag{5.1}$$

whose concentration parameters $\boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{\theta}) = \exp f(\boldsymbol{x}, \boldsymbol{\theta})$ are output by a neural network f with parameters $\boldsymbol{\theta}$; Γ is the Gamma function and $\alpha_0(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{k=1}^{K} \alpha_k(\boldsymbol{x}, \boldsymbol{\theta})$ with $\alpha_k = \exp f_k(\boldsymbol{x}, \boldsymbol{\theta})$ indexing the k^{th} element of the vector of all K concentration parameters $\boldsymbol{\alpha}$. Precision α_0 controls the sharpness of the density with more mass concentrating around the mean as α_0 grows. By conjugate property, the predictive distribution for a new point $\boldsymbol{x}^*$ is

$$P(Y = k \mid \boldsymbol{x}^*, \mathcal{D}) \approx \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x}^*)}[\pi_k] = \frac{\exp f_k(\boldsymbol{x}^*, \boldsymbol{\theta})}{\sum_{j=1}^{K} \exp f_j(\boldsymbol{x}^*, \boldsymbol{\theta})}, \tag{5.2}$$

which is the usual output of a network f with softmax activation.

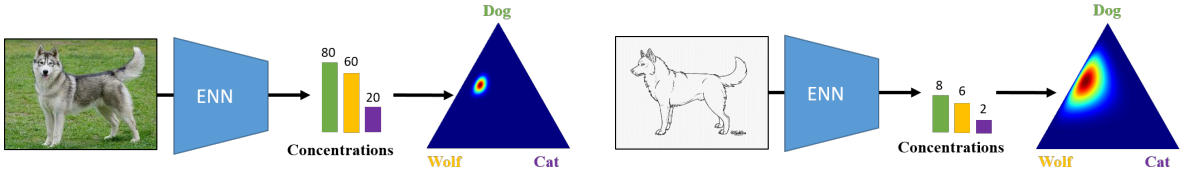
The concentration parameters $\boldsymbol{\alpha}$ can be interpreted as pseudo-counts representing the amount of evidence in each class. For instance, in Fig. 5.2a, the $\boldsymbol{\alpha}$'s output by the ENN indicate that the image is almost equally likely to be classified as *wolf* or as *dog*. More interestingly, it also distinguishes these in-distribution images from the OOD sample in Fig. 5.2b via the total amount of evidence α_0.

Training Objective The ENN training is formulated as a variational approximation to minimize the KL divergence between the distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x})$ and the true posterior distribution $p(\boldsymbol{\pi}|\boldsymbol{x}, y)$:

$$\begin{aligned}
\mathcal{L}_{\text{var}}(\boldsymbol{\theta}; \mathcal{D}) &= \mathbb{E}_{(\boldsymbol{x},y)\sim P(X,Y)}\big[\mathbb{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x}) \parallel p(\boldsymbol{\pi}|\boldsymbol{x}, y))\big] && (5.3)\\
&= \frac{1}{N} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \Big[\int q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x}) \log \frac{q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x})}{p(\boldsymbol{\pi}|\boldsymbol{x}, y)} \Big] && (5.4)\\
&= \frac{1}{N} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \Big[\int q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x}) \log \frac{q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x}) p(y|\boldsymbol{x})}{p(y|\boldsymbol{\pi}, \boldsymbol{x}) p(\boldsymbol{\pi}|\boldsymbol{x})} \Big] && (5.5)\\
&= \frac{1}{N} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \Big[\mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x})}\big[-\log p(y|\boldsymbol{\pi}, \boldsymbol{x})\big] + \mathbb{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x}) \parallel p(\boldsymbol{\pi}|\boldsymbol{x})) + \log p(y|\boldsymbol{x}) \Big], && (5.6)
\end{aligned}$$

where $N = \mathrm{card}(\mathcal{D})$. As the log-likelihood $\log p(y|\boldsymbol{x})$ does not depend on parameters $\boldsymbol{\theta}$,

$$\min_{\theta} \mathcal{L}_{\text{var}}(\boldsymbol{\theta}; \mathcal{D}) = \min_{\theta} \frac{1}{N} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \Big[\mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x})}\big[-\log p(y|\boldsymbol{\pi}, \boldsymbol{x})\big] + \mathbb{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x}) \parallel p(\boldsymbol{\pi}|\boldsymbol{x})) \Big]. \tag{5.7}$$



(a) In-distribution image
MCP = 0.50 , entropy = 0.97, KLoS = 97.85

(b) Outlier with same class confusion
MCP = 0.50 , entropy = 0.97, KLoS = 104.71

Figure 5.2: **Limitations of first-order uncertainty measures and their handling with KLoS**. (a) An in-distribution image with conflicting evidence between *dog* and *wolf*. (b) An outlier with the same class confusion but a lower amount of evidence. An evidential neural network (ENN) outputs class-wise evidence information as concentration parameters of a Dirichlet density (visualized on the simplex) over 3-class distributions. Although this density is flatter for the second input, the predictive entropy and MCP, only based on first-order statistics, are equal for both inputs. In contrast, the proposed measure, KLoS, captures both class confusion and lack of evidence, hence correctly reflecting the larger uncertainty for the latter sample.

For conciseness, we denote $\alpha_k = \alpha_k(\boldsymbol{x}, \boldsymbol{\theta}), \forall k \in \mathcal{Y}$ hereafter. For a sample $(\boldsymbol{x}, y)$, the reverse cross-entropy term amounts to $\mathbb{E}_{\boldsymbol{\pi} \sim q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x})}[-\log p(y|\boldsymbol{\pi}, \boldsymbol{x})] = -(\psi(\alpha_y) - \psi(\alpha_0))$ where ψ is the digamma function. Hence, the optimization objective is written as:

$$\min_{\theta} \mathcal{L}_{\text{var}}(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{N} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} -(\psi(\alpha_y) - \psi(\alpha_0)) + \mathbb{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x}) \parallel p(\boldsymbol{\pi}|\boldsymbol{x})). \tag{5.8}$$

Considering that most of the training inputs $\boldsymbol{x}$ are associated with only one observation y in $\mathcal{D}$, we should choose small concentration parameters $\boldsymbol{\beta}$ for the prior $p(\boldsymbol{\pi}|\boldsymbol{x}) = \text{Dir}(\boldsymbol{\pi}|\boldsymbol{\beta})$ to prevent the resulting posterior distribution $p(\boldsymbol{\pi}|\boldsymbol{x}) = \text{Dir}(\boldsymbol{\pi}|\beta_1, ..., \beta_y + 1, ..., \beta_K)$ from being dominated by the prior. However, this causes gradients to be very large in small-value regimes due to the digamma function, *e.g.* $\psi'(0.01) > 10^{-4}$.

To stabilize the optimization, we follow [170] and use the non-informative uniform prior distribution $p(\boldsymbol{\pi}|\boldsymbol{x}) = \text{Dir}(\boldsymbol{\pi}|\mathbf{1})$ where $\mathbf{1}$ is the all-one uniform vector, and we weight the KL divergence term with $\lambda > 0$:

$$\mathcal{L}^{\lambda}_{\text{var}}(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{N} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} -(\psi(\alpha_y) - \psi(\alpha_0)) + \lambda \mathbb{KL}(\text{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{\theta}))||\text{Dir}(\boldsymbol{\pi}|\mathbf{1})). \tag{5.9}$$

In particular, minimizing loss Eq. (5.9) enforces training sample's precision α_0 to remain close to the value $K + 1/\lambda$ [135].

While $\mathcal{L}^{\lambda}_{\text{var}}(\boldsymbol{\theta}; \mathcal{D})$ slightly differs from $\mathcal{L}_{\text{var}}(\boldsymbol{\theta}; \mathcal{D})$, both functions lead to the same optima. Indeed, by considering their gradient, we can show that a local optimum of $\mathcal{L}_{\text{var}}(\boldsymbol{\theta}; \mathcal{D})$ is achieved for a sample $\boldsymbol{x}$ when $\boldsymbol{\alpha}^* = (\beta_1, ..., \beta_y + 1, ..., \beta_K)$ and a local optimum of $\mathcal{L}^{\lambda}_{\text{var}}(\boldsymbol{\theta}; \mathcal{D})$ is $\boldsymbol{\alpha}^\bullet = (1, \cdots, 1 + 1/\lambda, \cdots, 1)$. Hence, their ratio between each element is equal:

$$\forall i, j \in [\![1, K]\!], \frac{\alpha^*_i}{\alpha^*_j} = \frac{\alpha^\bullet_i}{\alpha^\bullet_j}. \tag{5.10}$$

5.3 Capturing in-distribution and out-of-distribution uncertainties

In this section, we present the limits of current uncertainty measures used in evidential models (Section 5.3.1) and we introduce our measure to effectively capture class confusion and lack of evidence with evidential models (Section 5.3.2). We further propose a confidence learning approach to enhance in-distribution uncertainty estimation in Section 5.3.3.

5.3.1 Limits of current uncertainty measures with evidential models

For open-world recognition, a model should be equipped with an uncertainty measure which accounts both for first-order and second-order uncertainties to detect misclassifications and out-of-distribution samples. However, current uncertainty measures do not leverage the distribution over output probabilities on the simplex to derive such a joint measure of the two sources of uncertainty. The predictive entropy $\mathbb{H}[Y|\boldsymbol{x}, \mathcal{D}]$ and the *maximum class probability* (MCP) targeting total uncertainty actually reduce distributions of probabilities on the simplex to their expected value and compute first-order uncertainty measure from point-wise probabilities [56, 170]. This significantly limits the expressiveness of the resulting measures. In particular, they are invariant to the spread of the distribution over probabilities. This causes a significant loss of information. Given similar conflicting evidence, first-order uncertainty measures assign the same value to in-distribution and out-of-distribution samples as shown in Fig. 5.2, which is undesirable. With the goal of obtaining accurate estimates, measures should allow uncertainty caused by class confusion and lack of evidence to be cumulative, a property naturally fulfilled by the predictive variance in Bayesian regression [98].

Other uncertainty measures used with evidential models include second-order uncertainty measures which quantify the distribution dispersion on the simplex, *e.g.*, precision α_0 or mutual information. While adequate to detect OOD samples, these measures are not suited to estimate aleatoric uncertainty, which is characterized by the expectation of the Dirichlet distribution.

In addition, the success of these measures rely on the assumption that the Dirichlet distribution spread is larger for OOD than for in-distribution (ID) samples. Consequently, some previous works [57, 135, 171] propose to use auxiliary OOD data during training to enforce higher distribution spread on OOD inputs, which may be unrealistic in many applications. This has been debated recently by [116]. Finally, this assumption is not always fulfilled in absence of OOD training data [116, 117]. As shown in Fig. 5.3 for a model trained on CIFAR-10, α_0 values largely overlap between IDs and OODs when no OOD training data is used, limiting the effectiveness of existing second-order uncertainty measures. Consequently, neither current first-order nor second-order uncertainty measures appear to be suited for open-world settings.

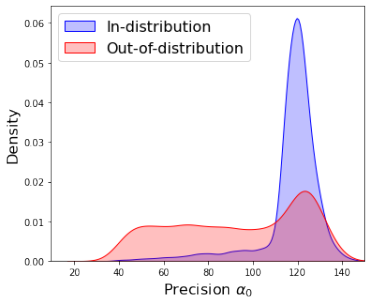


Figure 5.3: Precision densities for ID (CIFAR-10) and OOD (TinyImageNet) samples when no OOD training data is used.

5.3.2 KLoS: a Kullback-Leibler divergence measure on the simplex

By explicitly learning a distribution of the categorical probabilities $\boldsymbol{\pi}$, evidential models provide a second-order uncertainty representation where the expectation of the Dirichlet distribution relates to class confusion and its spread to the amount of evidence. While originally used to measure the total uncertainty, the predictive entropy $\mathbb{H}[y|\boldsymbol{x},\boldsymbol{\theta}]$ and the maximum class probability $\text{MCP}(\boldsymbol{x},\boldsymbol{\theta}) = \max_k P(\mathbf{y}=k|\boldsymbol{x},\boldsymbol{\theta})$ only account for the position on the simplex. These measures are invariant to the dispersion of the Dirichlet distribution that generates the categorical probabilities. This can be problematic, as illustrated in Fig. 5.2. To capture uncertainties due to class confusion *and* lack of evidence, an effective measure should account for the sharpness of the Dirichlet distribution and its location on the simplex.

We introduce a novel measure, named *KLoS* for "KL on Simplex", that computes the KL divergence between the model's output and a class-wise prototype Dirichlet distribution with concentrations $\boldsymbol{\gamma}_{\hat{y}}$ focused on the *predicted* class $\hat{y}$.

Definition 5.1 (KLoS)**.** *For any admissible input $\boldsymbol{x} \in \mathcal{X}$, its KLoS measure is defined as:*

$$KLoS(\boldsymbol{x}) \triangleq \mathbb{KL}\Big(Dir(\boldsymbol{\pi}|\boldsymbol{\alpha}(\boldsymbol{x},\boldsymbol{\theta})) \parallel Dir(\boldsymbol{\pi}|\boldsymbol{\gamma}_{\hat{y}})\Big), \tag{5.11}$$

where $\boldsymbol{\alpha}(\boldsymbol{x},\boldsymbol{\theta}) = \exp f(\boldsymbol{x},\boldsymbol{\theta})$ is the model's output and $\boldsymbol{\gamma}_{\hat{y}} = (1,\dots,1,\tau,1,\dots,1)$ are uniform concentration parameters except for the predicted class with concentration τ.

The lower KLoS is, the more certain the prediction is. Correct predictions will have Dirichlet distributions similar to the prototype Dirichlet distribution $\boldsymbol{\gamma}_{\hat{y}}$ and will thus be associated with a low uncertainty score (Fig. 5.4a). Samples with high class confusion will present an expected probability distributions closer to simplex's center than the expected class-wise prototype $p^*_{\hat{y}} = (\frac{1}{K-1+\tau},\cdots,\frac{\tau}{K-1+\tau},\dots,\frac{1}{K-1+\tau})$, resulting in a higher KLoS score (Fig. 5.4b). Similarly, KLoS also penalizes samples having a different precision α_0 than the precision $\alpha^*_0 = \tau + K - 1$ of the prototype $\boldsymbol{\gamma}_{\hat{y}}$. Samples with smaller (Fig. 5.4c) and higher (Fig. 5.4d) amount of evidence than α^*_0 receive a larger KLoS score.

Effective measure without OOD training data. Since in-distribution samples are enforced to have precision close to α^*_0 during training, the class-wise prototypes are fine estimates of the concentration parameters of training data for each class. Hence, KLoS is a divergence-based metric, which only needs in-distribution data during training to compute its prototypes. This behavior is illustrated in Section 5.5.1. The proposed measure will be effective to detect various types of OOD samples whose precision is far from α^*_0. In contrast, second-order uncertainty measures, *e.g.*, mutual information, assume that OOD samples have smaller α_0, a property difficult to fulfill for models trained only with in-distribution samples (see Fig. 5.3). In Section 5.5.5, we explore more in-depth the impact of the choice of OOD training data on the actual α_0 values for OOD samples.

Decomposition of KLoS Even though our method targets the simultaneous detection of misclassifications and OOD samples, one can detect the source of uncertainty in KLoS scores by

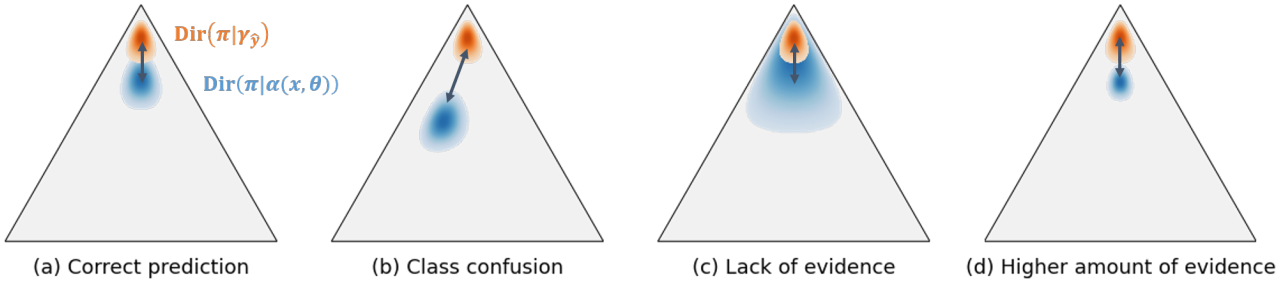


Figure 5.4: **KLoS on the probability simplex**. Given the input sample, the blue region represents the distribution predicted by the evidential model and the orange region represents the prototype Dirichlet distribution with parameters $\boldsymbol{\gamma}_{\hat{y}} = (1, \cdots, 1, \tau, 1, \ldots, 1)$ focused on the predicted class $\hat{y}$. Illustration of the behavior of KLoS in absence of uncertainty (a), in case of class confusion (b) and in case of a different amount of evidence, either lower (c) or higher (d).

using the following decomposition:

Proposition 5.1. *By approximating the digamma function ψ, KLoS can be decomposed as:*

$$KLoS(\boldsymbol{x}) \approx -(\tau-1)\log\left(\frac{\alpha_{\hat{y}}}{\alpha_0}\right) + \Big(-(\tau-1)\left(\frac{1}{2\alpha_0} - \frac{1}{2\alpha_{\hat{y}}}\right) + \mathbb{KL}\big(Dir(\boldsymbol{\pi}|\boldsymbol{\alpha}) \parallel Dir(\boldsymbol{\pi}|\mathbf{1})\big)\Big) + r, \quad (5.12)$$

where $r = -\big(\log\Gamma(\tau) - \log\Gamma(K-1+\tau) - \log\Gamma(K)\big)$ does not depend on the model parameters $\boldsymbol{\theta}$ nor on the input $\boldsymbol{x}$.

Proof. The KL divergence between two Dirichlet distributions can be obtained in closed form and KLoS can be calculated as:

$$\begin{aligned} \mathrm{KLoS}(\boldsymbol{x}) &= \mathbb{KL}\Big(\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \parallel \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\gamma}_{\hat{y}})\Big) && (5.13)\\ &= \log\Gamma(\alpha_0) - \log\Gamma(K-1+\tau) + \log\Gamma(\tau) - \sum_{k=1}^{K}\log\Gamma(\alpha_k) \\ &\quad + \sum_{k\neq y}(\alpha_k - 1)\big(\psi(\alpha_k) - \psi(\alpha_0)\big) + (\alpha_{\hat{y}} - \tau)\big(\psi(\alpha_{\hat{y}}) - \psi(\alpha_0)\big). && (5.14)\end{aligned}$$

On the other hand, the KL divergence between the model's output and an uniform Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\pi}|\mathbf{1})$ reads:

$$\mathrm{KL}\Big(\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \parallel \mathrm{Dir}(\boldsymbol{\pi}|\mathbf{1})\Big) = \log\Gamma(\alpha_0) - \log\Gamma(K) - \sum_{k=1}^{K}\log\Gamma(\alpha_k) + \sum_{k=1}^{K}(\alpha_k - 1)\big(\psi(\alpha_k) - \psi(\alpha_0)\big). \quad (5.15)$$

Hence, KLoS can be written as:

$$\begin{aligned} \mathrm{KLoS}(\boldsymbol{x}) &= -(\tau-1)\big(\psi(\alpha_{\hat{y}}) - \psi(\alpha_0)\big) + \mathbb{KL}(\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \parallel \mathrm{Dir}(\boldsymbol{\pi}|\mathbf{1})) \\ &\quad + \big(\log\Gamma(\tau) - \log\Gamma(K-1+\tau) - \log\Gamma(K)\big). && (5.16)\end{aligned}$$

By considering the asymptotic series approximation to the digamma function, $\psi(x) = \log x - \frac{1}{2x} + \mathcal{O}(\frac{1}{x^2})$, the previous expression can be approximated by:

$$\mathrm{KLoS}(\boldsymbol{x}) \approx -(\tau-1)\log(\frac{\alpha_{\hat{y}}}{\alpha_0}) + \Big(-(\tau-1)(\frac{1}{2\alpha_0} - \frac{1}{2\alpha_{\hat{y}}})) + \mathbb{KL}(\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \parallel \mathrm{Dir}(\boldsymbol{\pi}|\mathbf{1}))\Big) + r, \quad (5.17)$$

where $r = -\big(\log\Gamma(\tau) - \log\Gamma(K-1+\tau) - \log\Gamma(K)\big)$. □

The first term is the standard log-likelihood and relates only to expected probabilities, hence to the class confusion. The ratio $\alpha_{\hat{y}}/\alpha_0$ makes it invariant to any scaling of the concentration parameters vector $\boldsymbol{\alpha}$. The second term takes into account the spread of the distribution by measuring how close α_0 is to $(\tau + K - 1)$, and measures the amount of evidence.

5.3.3 Improving uncertainty estimation with confidence learning

When the model misclassifies an example, *i.e.*, the predicted class $\hat{y}$ differs from the ground truth y, KLoS measures the distance between the ENN's output and the wrongly estimated posterior $p(\boldsymbol{\pi}|\boldsymbol{x}, \hat{y})$. This may result in an arbitrarily high confidence / low KL divergence value. Measuring instead the distance to the true posterior distribution $p(\pi|\boldsymbol{x}, y)$ (Fig. 5.5) more likely yield a greater value, reflecting the fact that the classifier made an error.

Thus, a better measure for misclassification detection would be:

$$\mathrm{KLoS}^*(\boldsymbol{x}, y) \triangleq \mathbb{KL}\Big(\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}(\boldsymbol{x},\boldsymbol{\theta})) \parallel \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\gamma}_y)\Big), \quad (5.18)$$

where $\boldsymbol{\gamma}_y$ corresponds to the uniform concentrations except for the *true* class y with $\tau = 1 + \lambda^{-1}$.

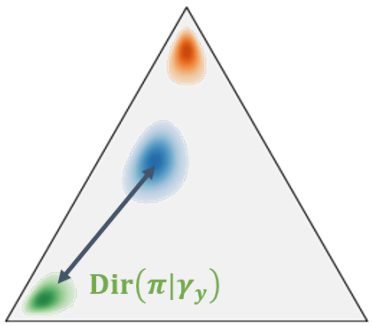


Figure 5.5: KLoS* measures the distance to the prototype Dirichlet distribution centered on the true class y (green).

Connecting KLoS* with Evidential Training Objective Interestingly, the following proposition that choosing such value for τ results in KLoS* matching the objective function in Eq. (5.9).

Proposition 5.2. *If $\tau = 1 + \lambda^{-1}$, then minimizing the evidential training objective $\mathcal{L}^{\lambda}_{var}(\boldsymbol{\theta}; \mathcal{D})$ is equivalent to minimizing the KLoS* value of each training point $\boldsymbol{x}$.*

Proof. Let us decompose $\mathcal{L}^{\lambda}_{\mathrm{var}}(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{N}\sum_{(\boldsymbol{x},y)\in\mathcal{D}} l^{\lambda}_{\mathrm{var}}(\boldsymbol{x}, y, \boldsymbol{\theta})$.

By deriving KLoS* in a similar way than Eq. (5.16), we can observe that:

$$\mathrm{KLoS}^*(\boldsymbol{x}) = l^{\lambda}_{\mathrm{var}}(\boldsymbol{x}, y, \boldsymbol{\theta}) + r, \quad (5.19)$$

where $r = -\big(\log\Gamma(1+1/\lambda) - \log\Gamma(K-1+1/\lambda) - \log\Gamma(K)\big)$ does not depend on the model parameters $\boldsymbol{\theta}$.

Hence, minimizing the evidential training objective $\mathcal{L}^{\lambda}_{\text{var}}(\boldsymbol{\theta}; \mathcal{D})$ is equivalent to minimizing the KLoS* value of each training point $\boldsymbol{x}$. □

This means that KLoS* is explicitly minimized by the evidential model during training for in-distribution samples. By mimicking the evidential training objective, we reflect the fact that the model is confident about its prediction if KLoS* is close to zero. In addition, minimizing the KL divergence between the variational distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x})$ and the posterior $p(\boldsymbol{\pi}|\boldsymbol{x}, y)$ is equivalent to maximizing the evidence lower bound (ELBO) [98]. Hence, a small KLoS* value corresponds to a high ELBO, which is coherent with the common assumption in variational inference that higher ELBO corresponds to "better" models [172].

Obviously, the true class of an output is not available when estimating confidence on test samples. We propose to learn KLoS* by introducing an auxiliary confidence neural network, *KLoSNet*, with parameters $\boldsymbol{\omega}$, which outputs a confidence prediction $C(\boldsymbol{x}, \boldsymbol{\omega})$. KLoSNet consists in a small decoder, composed of several dense layers attached to the penultimate layer of the original classification network. During training, we seek $\boldsymbol{\omega}$ such that $C(\boldsymbol{x}, \boldsymbol{\omega})$ is close to KLoS$^*(\boldsymbol{x}, y)$, by minimizing

$$\mathcal{L}_{\text{KLoSNet}}(\boldsymbol{\omega}; \mathcal{D}) = \frac{1}{N} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \left\| C(\boldsymbol{x}, \boldsymbol{\omega}) - \text{KLoS}^*(\boldsymbol{x}, y) \right\|^2. \tag{5.20}$$

KLoSNet can be further improved by endowing it with its own feature extractor. Initialized with the encoder of the classification network, which must remain untouched for not affecting its performance, the encoder of KLoSNet can be fine-tuned along with its regression head. This amounts to minimizing Eq. (5.20) with respect to both sets of parameters.

The training set for confidence learning is the one used for classification training. In the experiments, we observe a slight performance drop when using a validation set instead. Indeed, when dealing with models with high predictive performance and small validation sets, we end up with fewer misclassification examples than in the train set. At test time, we now directly use KLoSNet's scalar output $C(\boldsymbol{x}, \boldsymbol{\omega}')$ as our uncertainty estimate. As previously, the lower the output value, the more confident the prediction.

5.4 Related work

We detail here related work on OOD detection used in the following experiments. For misclassification detection, we refer to the related work in Section 3.4 and the presentation of the task in the background chapter (Section 2.3.2).

In the literature, a range of methods aim to detect anomalies in the form of out-of-distribution (OOD) samples. Applied on a pre-trained model, ODIN [112] mitigates over-confidence by post-processing logits with temperature scaling and by adding inverse adversarial perturbations. [113] proposes a confidence score based on the class-conditional Mahalanobis distance, with the assumption of tied covariance. Although effective, both approaches need OOD data to tune hyperparameters, which might not generalize to other OOD datasets [173]. Finally, Liu *et al.* [124] interpret a pre-trained

NN as an energy-based model and compute the energy score to detect OOD samples. Interestingly, this score corresponds to the log precision $\log \alpha_0$, which is similar to the EPKL measure [135] used in ENNs.

5.5 Experiments

We evaluate our approach against: first-order uncertainty metrics (Maximum Class probability (*MCP*) and predictive entropy (*Entropy*)), second-order metrics (mutual information (*Mut. Inf.*), differential entropy (*Diff. Ent.*), expected pairwise KL divergence (*EPKL*) and *dissonance*), post-training methods for OOD detection (*ODIN* and *Mahalanobis*) and for misclassification detection (*ConfidNet*). Except in Section 5.5.5, we consider setups where no OOD data is available for training. Consequently, the results reported for ODIN and Mahalanobis are obtained without adversarial perturbations, which is also the best configuration for the considered tasks. We indeed show in Section 5.5.4 that these perturbations degrade misclassification detection.

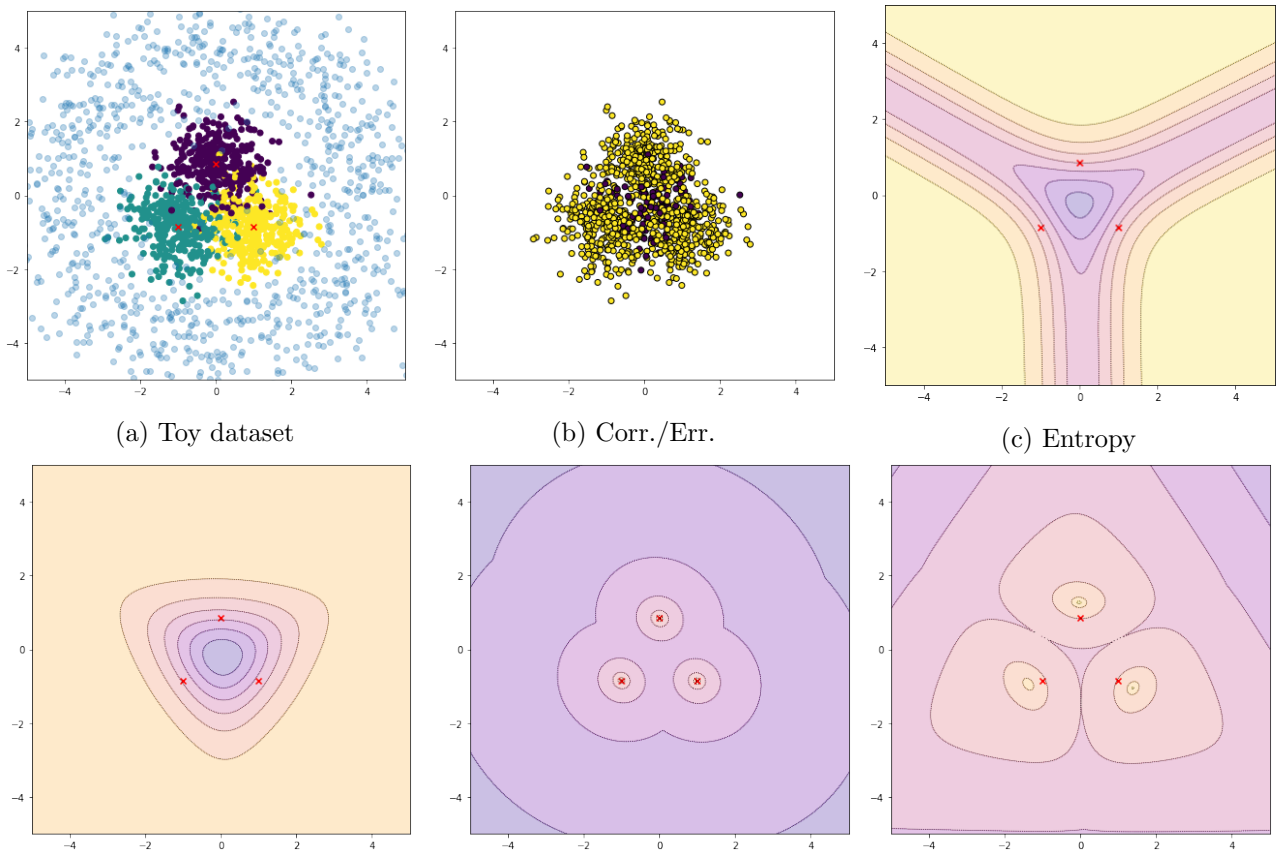
5.5.1 Synthetic experiment

We analyse the behavior of the KLoS measure and the limitations of existing first- and second-order uncertainty metrics on a 2D synthetic dataset composed of three Gaussian-distributed classes with equidistant means and identical isotropic variance (Fig. 5.6):

$$p(X = \boldsymbol{x}, Y = y) = \frac{1}{3} \cdot \mathcal{N}(X = \boldsymbol{x} \mid \boldsymbol{\mu}_y, \sigma^2 \mathrm{I}_{2\times 2}), \tag{5.21}$$

where $\boldsymbol{\mu}_1 = (0, \sqrt{3}/2)$, $\boldsymbol{\mu}_2 = (-1, -\sqrt{3}/2)$, $\boldsymbol{\mu}_3 = (1, -\sqrt{3}/2)$ and $\sigma = 4$. The marginal distribution of $\mathbf{x}$ is a Gaussian mixture with three equally weighted components having equidistant centers and equal spherical covariance matrices. The test dataset consists of 1,000 other samples from this distribution. Finally, we construct an out-of-distribution (OOD) dataset following [135], by sampling 100 points in $\mathbb{R}^2$ such that they form a 'ring' with large noise around the training points. Some OOD samples will be close to the in-distribution while others will be very far (see Fig. 5.6). The number of OOD samples has been chosen so that it amounts approximately to the number of test points misclassified by the classifier. The classification is performed by a simple logistic regression. A set of five models is trained for 200 epochs using the evidential training objective with regularization parameter $\lambda =$ 5e-2 and Adam optimizer with learning rate 0.02. Uncertainty metrics – MCP, Entropy, Mut. Inf., Malahanobis and KLoS – are computed from these models. This constitutes a scenario with high first-order uncertainty due to class overlap. OOD samples are drawn from a ring around the in-distribution dataset and are only used for evaluation.

Fig. 5.6c shows that Entropy correctly assigns large uncertainty along decision boundaries, which is convenient to detect misclassifications, but yields low uncertainty for points far from the distribution. Mut. Inf. (Fig. 5.6d) has the opposite behavior than desired by decreasing when moving away from the training data. This is due to the linear nature of the toy dataset where models assign higher concentration parameters far from decision boundaries, hence smaller spread on the simplex, as also



(a) Toy dataset (b) Corr./Err. (c) Entropy

(d) Mut. Inf. (e) Mahalanobis (f) KLoS

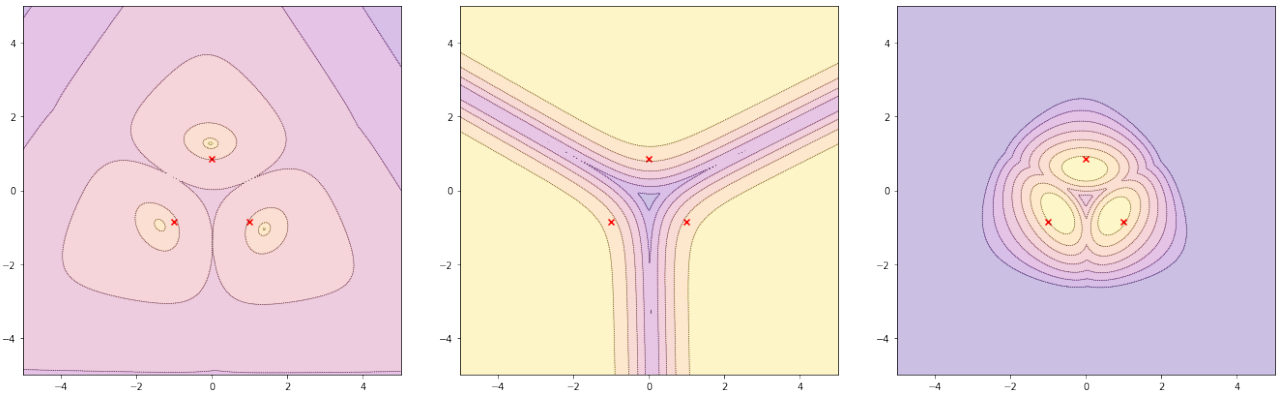
Figure 5.6: **Comparison of various uncertainty measures for a given evidential classifier on a toy dataset.** (a) Training samples from 3 input Gaussian distributions with large overlap (hence class confusion) and OOD test samples (blue); (b) Correct (yellow) and erroneous (red) class predictions on in-domain test samples; (c-f) Visualisation of different uncertainty measures derived from the evidential model trained on the toy dataset. Yellow (resp. purple) indicates high (resp. low) certainty.

noted in [116]. Additionally, Mut. Inf. does not reflect the uncertainty caused by class confusion along decision boundaries. Neither Entropy nor Mut. Inf. is suitable to detect OOD samples in this synthetic experiment. In contrast, KLoS allows discriminating both misclassifications and OOD samples from correct predictions as uncertainty increases far from in-distribution samples for each class (Fig. 5.6f). KLoS measures a distance between the model's output and a class-wise prototype distribution. Here, we can observe that it acts as a divergence-based measure for each class.

We extend the comparison to include Mahalanobis (Fig. 5.6e), which is a distance-based measure by assuming Gaussian class conditionals on latent representations, here in the input space. However, Mahalanobis does not discriminate points close to the decision boundaries from points with a similar distance to the origin. Hence, it may be less suited to detect misclassifications than KLoS. Additionally, KLoS does not assume Gaussian distributions in the latent space nor tied covariance, which may be a strong assumption when dealing with a high-dimension latent space. In Section B.2.1, a complementary

quantitative evaluation on this toy problem confirms our findings regarding the inadequacy of first-order uncertainty measures such as MCP and Entropy, and the improvement provided by KLoS over Mahalanobis on misclassification detection.

Decomposition of KLoS. To gain further intuition about the decomposition, we provide illustrations of the first term (negative log-likelihood, NLL) and the second term in Fig. 5.7. We observe that the NLL term, which is equivalent to MCP measure, helps to detect misclassifications while the second term denotes increasing uncertainty as we move far away from training data. Hence, by using either the NLL term or the second term, one could distinguish the source of uncertainty if needed.



(a) KLoS (b) NLL term (c) Second term

Figure 5.7: **Visualisation of the decomposition of KLoS on the toy dataset.**

5.5.2 Simultaneous detection of errors and out-of-distribution samples

The task of detecting both in-distribution misclassifications and OOD samples gives the opportunity to jointly evaluate in-distribution and out-of-distribution uncertainty representations of a method. In this binary classification problem, correct predictions are considered as positive samples while misclassified inputs and OOD examples constitute negative samples. Following standard practices [45], we use the area under the ROC curve (AUROC) to evaluate threshold-independent performance.

The models used in the experiments present high predictive performances. Most often, there are much fewer misclassifications in the test set than considered OOD samples. Hence, joint detection performances might be dominated by the evaluation of the quality of OOD detection. To mitigate this unbalance, we propose to consider the following scheme based on oversampling. Let $\mathcal{A}_{\text{M}}$ be the subset of in-distribution test examples that are misclassified by the observed model and $\mathcal{A}_{\text{O}}$ the set of OOD test samples. We randomly sample $\kappa|\mathcal{A}_{\text{O}}|$ points in $\mathcal{A}_{\text{M}}$, with $\kappa = 1$. Supposing $|\mathcal{A}_{\text{O}}| \geq |\mathcal{A}_{\text{M}}|$, this corresponds to oversampling the set of misclassifications. This over-sampled set is then added to the OOD set to form the negative examples for detection training. The set of correct predictions remains

(a) CIFAR-10 (b) TinyImageNet (c) LSUN (d) STL10

Figure 5.8: **Images samples from OOD datasets** (b,c,d) used in experiments and compared to in-distribution CIFAR-10 (a).

the same. We observed that the variance in AUROC due to this sampling is negligible and we report only the mean hereafter.

In the following, uncertainty measures are derived from an evidential model (Eq. (5.9)) with $\lambda = 10^{-2}$, except for second-order metrics where we found that setting $\lambda = 10^{-3}$ improves performance. We rely on the learned classifier to train our auxiliary confidence model KLoSNet, using the same training set and following loss Eq. (5.20). Experiments are conducted with VGG-16 [74] and ResNet-18 [76] architectures on CIFAR-10 (Fig. 5.8a) and CIFAR-100 datasets [35]. The OOD datasets used for evaluation are presented in Fig. 5.8: TinyImageNet[1] – a subset of ImageNet (10,000 test images with 200 classes) –, LSUN [174] – a scene classification dataset (10,000 test images of 10 scenes) –, STL-10 – a dataset similar to CIFAR-10 but with different classes, and SVHN [36] – an RGB dataset of 28×28 house-number images (73,257 training and 26,032 test images with 10 digits) –. We downsample each image of TinyImageNet, LSUN and STL-10 to size 32×32. All details about architectures, training algorithms and datasets are available in Chapter B.

Along with simultaneous detection results, we provide separate results for misclassifications detection and OOD detection respectively in Table 5.1. On OOD detection, Mahalanobis and KLoSNet outperform other methods, including second-order measures. ODIN also fails to deliver here as logits are small due to regularization in the evidential training objective. Mut. Inf. and other spread-based second-order uncertainty measures fall short to detect correctly OOD. Indeed, for settings where OOD training data is not available, there is no guarantee that every OOD sample will result in lower predicted concentration parameters as previously shown by the density plot of precision α_0 in Fig. 5.3. This stresses the importance of class-wise divergence-based measure.

While Mahalanobis may sometimes be slightly better than KLoSNet for OOD detection, it performs significantly less well in misclassification detection, in line with the behavior shown in synthetic experiments. As a result, KLoSNet appears to be the best measure in every simultaneous detection benchmark. For instance, for CIFAR-10/STL-10 with VGG-16, KLoSNet achieves 81.8% AUROC while the second best, Mahalanobis, scores 78.8%.

[1] https://tiny-imagenet.herokuapp.com/

Table 5.1: **Comparative experiments on CIFAR-10 and CIFAR-100.** Misclassification (Mis.), out-of-distribution (OOD) and simultaneous (Mis+OOD) detection results (mean % AUROC and std. over 5 runs). Bold type indicates significantly best performance ($p<0.05$) according to paired t-test.

			LSUN		**TinyImageNet**		**STL-10**	
	Method	Mis.	OOD	Mis+OOD	OOD	Mis+OOD	OOD	Mis+OOD
CIFAR-10 VGG-16	MCP	87.6 ±1.6	79.7 ±1.1	84.9 ±1.1	80.3 ±1.5	85.2 ±1.5	60.3 ±1.2	75.2 ±1.4
	Entropy	83.5 ±2.4	83.8 ±0.3	87.9 ±0.2	82.3 ±0.4	87.2 ±0.4	60.1 ±1.2	75.0 ±1.4
	ConfidNet	90.2 ±0.8	82.1 ±1.5	87.6 ±1.1	83.5 ±0.6	88.3 ±0.7	61.5 ±1.6	77.2 ±1.1
	Dissonance	91.9 ±0.2	84.8 ±0.3	90.1 ±0.1	84.2 ±0.2	89.7 ±0.1	64.1 ±0.1	79.6 ±0.1
	Mut. Inf.	84.1 ±1.5	84.6 ±0.6	85.1 ±1.0	80.6 ±0.8	83.4 ±1.1	61.3 ±0.8	65.0 ±2.5
	Diff. Ent.	86.8 ±1.0	85.6 ±0.5	87.2 ±0.7	82.7 ±0.7	85.8 ±0.8	62.0 ±1.0	75.4 ±1.3
	EPKL	83.9 ±1.5	84.5 ±0.7	85.1 ±1.0	80.4 ±0.8	83.2 ±1.2	61.3 ±0.8	73.8 ±1.1
	Diss.+Mut. Inf.	92.0 ±0.2	86.5 ±0.3	89.8 ±0.2	83.6 ±0.3	89.5 ±0.3	63.6 ±0.5	79.4 ±0.4
	ODIN	86.0 ±2.0	79.5 ±1.2	83.8 ±1.5	79.6 ±1.9	84.0 ±2.0	54.7 ±1.5	65.0 ±2.6
	Mahalanobis	91.2 ±0.3	**88.9** ±0.2	**91.3** ±0.1	**86.4** ±0.2	90.2 ±0.1	63.4 ±0.2	78.8 ±0.3
	KLoSNet (Ours)	**92.5** ±0.6	87.6 ±0.9	**91.7** ±0.9	**86.6** ±0.9	**91.2** ±0.8	**67.7** ±1.4	**81.8** ±0.9
CIFAR-10 ResNet-18	MCP	84.9 ±0.8	79.6 ±1.0	83.0 ±0.9	77.2 ±0.7	81.8 ±0.7	58.5 ±1.2	72.5 ±0.4
	Entropy	84.6 ±0.8	79.6 ±1.1	82.8 ±0.9	77.2 ±0.7	81.6 ±0.7	58.4 ±1.2	72.2 ±0.4
	ConfidNet	90.7 ±0.4	84.6 ±1.1	88.6 ±0.6	83.5 ±1.1	88.0 ±0.6	63.2 ±1.2	77.9 ±0.5
	Dissonance	92.9 ±0.4	90.3 ±0.4	92.7 ±0.4	87.7 ±0.3	91.4 ±0.3	67.3 ±0.5	81.2 ±0.4
	Mut. Inf	80.6 ±0.6	77.0 ±1.2	79.4 ±0.9	74.3 ±0.8	78.0 ±0.7	56.4 ±1.0	69.1 ±0.2
	Diff. Ent	82.7 ±0.6	78.3 ±1.2	81.1 ±0.9	75.9 ±0.8	79.9 ±0.7	57.5 ±1.1	70.8 ±0.3
	EPKL	80.2 ±0.6	76.8 ±1.3	79.0 ±0.9	74.1 ±0.8	77.7 ±0.7	56.2 ±1.0	68.9 ±0.3
	Diss.+Mut. Inf.	92.4 ±0.5	86.7 ±1.0	90.1 ±0.8	84.3 ±0.5	88.8 ±0.6	65.2 ±0.7	80.3 ±0.4
	ODIN	83.7 ±0.7	78.9 ±1.0	81.9 ±0.9	76.5 ±0.7	80.7 ±0.7	57.9 ±1.2	71.5 ±0.4
	Mahalanobis	91.2 ±0.4	90.7 ±0.4	91.8 ±0.3	87.6 ±0.4	90.3 ±0.4	66.8 ±0.5	80.0 ±0.3
	KLoSNet (Ours)	**93.9** ±0.4	**93.1** ±1.1	**94.4** ±0.3	**90.6** ±0.6	**93.2** ±0.2	**68.5** ±0.3	**82.3** ±0.2
CIFAR-100 VGG-16	MCP	82.9 ±0.8	62.8 ±1.3	77.6 ±0.9	72.0 ±0.5	81.8 ±0.7	69.7 ±0.7	80.9 ±0.7
	Entropy	82.2 ±0.8	63.2 ±1.4	77.2 ±1.0	72.5 ±0.6	81.5 ±0.8	70.1 ±0.8	80.6 ±0.7
	ConfidNet	84.4 ±0.6	65.3 ±2.0	80.0 ±1.3	73.8 ±0.6	83.7 ±0.7	71.5 ±0.6	82.7 ±0.3
	Dissonance	84.1 ±0.4	62.5 ±1.4	78.7 ±0.8	70.3 ±0.4	82.5 ±0.4	69.3 ±0.4	82.2 ±0.4
	Mut. Inf.	78.9 ±0.8	65.6 ±0.7	76.2 ±0.9	71.8 ±0.2	79.1 ±0.4	70.1 ±0.6	78.5 ±0.6
	Diff. Ent.	80.2 ±0.8	65.6 ±0.9	77.2 ±0.8	72.7 ±0.3	80.4 ±0.4	71.0 ±0.5	79.7 ±0.5
	EPKL	78.8 ±0.8	65.2 ±1.0	76.1 ±0.9	71.6 ±0.2	78.9 ±0.4	70.0 ±0.6	78.3 ±0.6
	Diss.+Mut. Inf.	84.2 ±0.6	65.1 ±0.3	80.1 ±0.4	70.1 ±0.3	82.5 ±0.5	69.5 ±0.3	82.3 ±0.5
	ODIN	82.1 ±0.8	62.9 ±1.4	77.1 ±1.0	71.9 ±0.6	81.3 ±0.8	69.6 ±0.8	80.3 ±0.7
	Mahalanobis	84.0 ±0.2	**71.1** ±1.0	82.4 ±0.5	**77.0** ±0.5	84.9 ±0.3	**75.4** ±0.3	84.3 ±0.5
	KLoSNet (Ours)	**86.7** ±0.4	68.4 ±1.1	**83.0** ±0.6	76.4 ±0.4	**86.4** ±0.4	**75.0** ±0.5	**86.0** ±0.4
CIFAR-100 ResNet-18	MCP	84.0 ±0.4	70.4 ±0.9	81.0 ±0.3	76.6 ±0.5	83.6 ±0.4	75.4 ±0.5	83.1 ±0.2
	Entropy	83.7 ±0.4	70.4 ±0.9	80.8 ±0.3	76.9 ±0.5	83.5 ±0.3	75.7 ±0.5	83.0 ±0.3
	ConfidNet	**87.1** ±0.2	73.0 ±1.4	**84.5** ±0.6	79.1 ±0.3	86.8 ±0.3	**78.5** ±0.8	**86.6** ±0.5
	Dissonance	86.7 ±0.4	72.3 ±0.4	84.0 ±0.2	75.0 ±0.4	85.3 ±0.4	74.7 ±0.3	85.2 ±0.2
	Mut. Inf	82.6 ±0.4	70.2 ±1.1	80.0 ±0.4	76.4 ±0.6	82.6 ±0.3	75.1 ±0.5	82.1 ±0.3
	Diff. Ent	83.0 ±0.4	70.1 ±1.1	80.2 ±0.4	76.8 ±0.5	83.0 ±0.3	75.6 ±0.5	82.5 ±0.3
	EPKL	82.5 ±0.4	70.2 ±1.1	80.0 ±0.4	76.3 ±0.6	82.5 ±0.3	75.0 ±0.5	82.0 ±0.2
	Diss.+Mut. Inf.	86.5 ±0.4	71.8 ±0.8	83.6 ±0.5	76.1 ±0.3	84.7 ±0.4	75.2 ±0.5	84.6 ±0.3
	ODIN	83.7 ±0.4	70.3 ±0.9	80.8 ±0.3	76.6 ±0.5	83.5 ±0.3	75.4 ±0.5	83.0 ±0.3
	Mahalanobis	85.9 ±0.4	**75.2** ±0.6	**84.5** ±0.1	78.4 ±0.5	85.9 ±0.3	77.5 ±0.4	85.6 ±0.3
	KLoSNet (Ours)	86.9 ±0.3	73.1 ±0.4	**84.4** ±0.1	**80.8** ±0.2	**87.3** ±0.2	**79.0** ±0.2	**86.7** ±0.3

Table 5.2: **Impact of confidence learning.** Comparison of detection performances (% AUROC) between KLoS and KLoSNet for CIFAR-10 and CIFAR-100 experiments with VGG-16 architecture.

			LSUN		TinyImageNet		STL-10	
	Method	Mis.	OOD	Mis+OOD	OOD	Mis+OOD	OOD	Mis+OOD
CIFAR-10 VGG-16	KLoS	92.1 ±0.3	86.5 ±0.3	91.2 ±0.2	85.4 ±0.3	90.4 ±0.2	64.1 ±0.3	79.6 ±0.3
	KLoSNet	**92.5** ±0.6	**87.6** ±0.9	**91.7** ±0.9	**86.6** ±0.9	**91.2** ±0.8	**67.7** ±1.4	**81.8** ±0.9
CIFAR-100 VGG-16	KLoS	85.4 ±0.2	65.1 ±1.1	81.3 ±0.6	74.5 ±0.4	85.4 ±0.4	72.7 ±0.3	84.8 ±0.4
	KLoSNet	**86.7** ±0.4	**68.4** ±1.1	**83.0** ±0.6	**76.4** ±0.4	**86.4** ±0.4	**75.0** ±0.5	**86.0** ±0.4

We also observe that KLoSNet significantly improves misclassification detection, even compared to dedicated methods such as ConfidNet or second-order measures related to class confusion, *e.g.*, dissonance. Another baseline could be to combine two measures specialized respectively for class confusion and lack of evidence, such as Dissonance+Mut.Inf. But it still performs less well than KLoSNet.

Impact of Confidence Learning. To evaluate the effect of the uncertainty measure KLoS and of the auxiliary confidence network KLoSNet, we report a detailed ablation study in Table 5.2. We can notice that KLoSNet improves misclassification over KLoS but also OOD detection in every benchmark. We intuit that learning to improve misclassification detection also helps to spot some OOD inputs that share similar characteristics.

Oversampling Factor. When deploying a model in the wild, it is difficult to know beforehand the proportions of misclassifications and OOD samples the model will have to handle. Until now, we assumed an equal proportion in order to evaluate equally the capacity to detect both kinds of inputs. In Fig. 5.9, we vary the oversampling factor κ in $[0.01, 100]$ for CIFAR10/TinyImageNet to assess the robustness of tested methods and our approach. The higher the oversampling factor is, the more misclassifications will be sampled in the test set, hence giving importance to misclassification detection, and vice versa. Results show that regardless of the value chosen for oversampling, KLoSNet consistently outperforms all other measures, with a larger gain when κ increases.

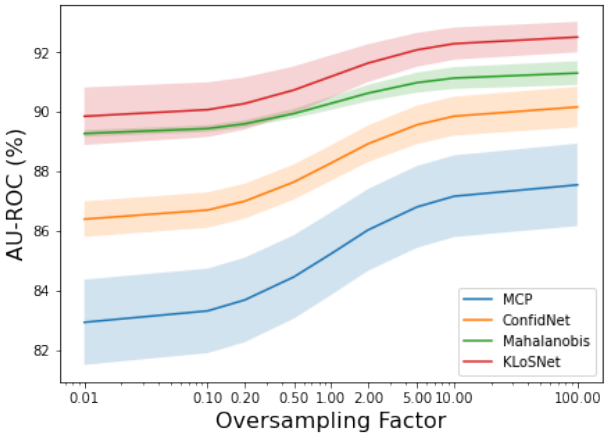
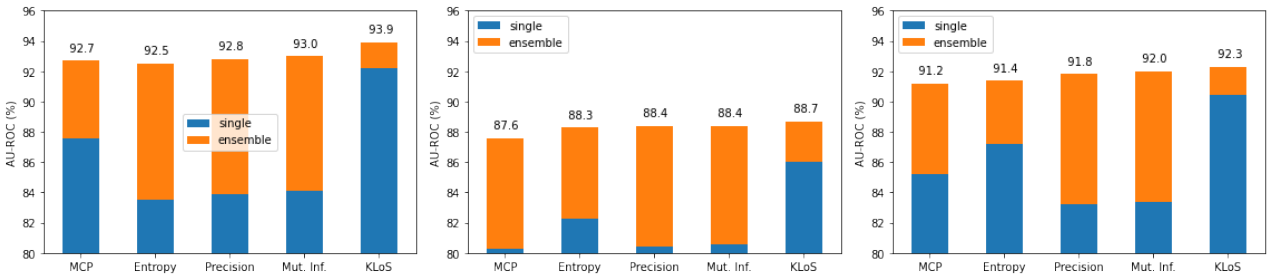


Figure 5.9: **Impact of the oversampling factor κ** (CIFAR-10/TinyImageNet).

Combining KLoS with Ensembling. Aggregating predictions from an ensemble of neural networks not only improves generalization [53, 175] but also uncertainty estimation [54]. We train an ensemble of ten evidential models on CIFAR-10 and evaluate the performance of various uncertainty measures – MCP,



(a) Mis. detection (b) OOD detection (c) Mis.+OOD detection

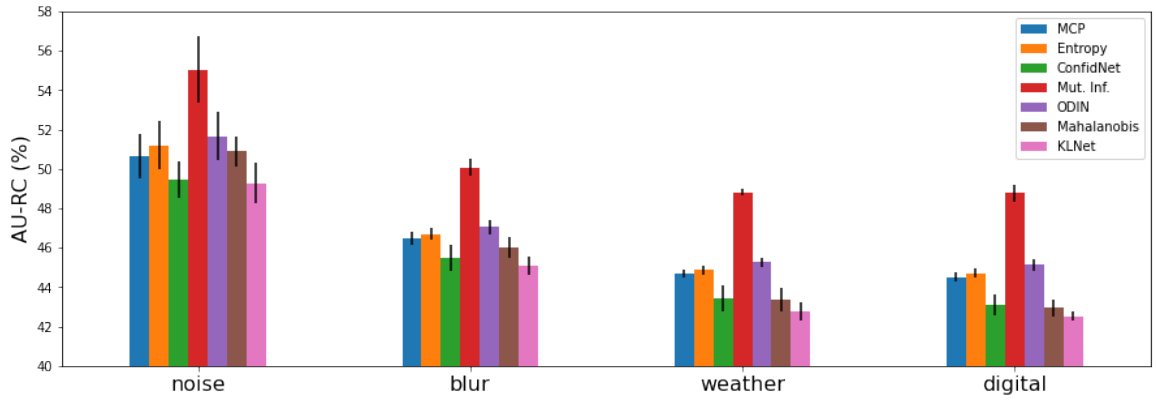
Figure 5.10: **Comparative gain with ensembling for each detection task on CIFAR10 vs. TinyImageNet.** Ensembling improves performances with every tested method, in particular in OOD detection (Fig. 5.10b). KLoS remains the best method when combined with ensembling.

predictive entropy, precision, mutual information, and KLoS – obtained from averaged concentration parameters. Results for each detection task with CIFAR-10/TinyImageNet benchmark are available in Fig. 5.10. As expected, every method obtains improved performance when computed from an ensemble of models. The gain is particularly pronounced for OOD detection: for instance performance with precision α_0 is improved by +8.0 and with mutual information by +7.8 points. These gains are due to the diversity in predictions provided by ensembling which helps to better capture epistemic uncertainty, as explained in Chapter 2. While improvements with KLoS are less significant, KLoS remains the best measure in each detection task with respectively 93.8% AU-ROC in misclassification detection, 88.7% in OOD detection and 92.3% in the joint detection. One possible explanation is that KLoS was already capturing effectively epistemic uncertainty and the improvement with ensembling may consequently be less significant.

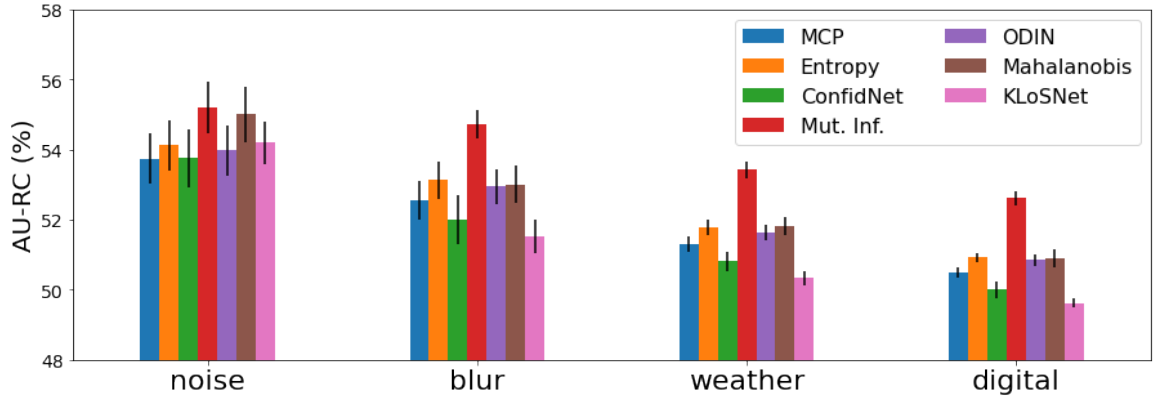
5.5.3 Selective classification in presence of distribution shifts

Classification with a reject option, also known as *selective classification* [44], consists in a scenario where a classifier can abstain on samples where its confidence is below a certain threshold. This is appropriate for applications where the system can hand over to human experts or users. Performance can be measured on *risk-coverage* curves. We recall evaluation metrics in the following and refer the reader to Section 2.3 for a more detail description. The *coverage* is the probability mass of the non-rejected region in $\mathcal{X}$ and can be empirically estimated by the percentage of the non-rejected samples. The *risk* of a selective classifier is the average loss on the accepted samples. Given a chosen coverage, good selective classifiers correlate with low risk. Averaged performances are evaluated on risk-coverage curves with a threshold-independent area-under-curve metric, denoted here AURC. The lower the AURC, the better the selective classifier.

Previous works evaluate the performance on in-distribution data. However, a classifier may encounter data drawn from a different distribution when deployed in the wild. Following [26], we extend selective classification by penalizing non-rejected OOD samples. If a sample is drawn from the in-distribution, we compute the 0/1 cost function as usual. For OOD samples, we apply the maximum



(a) CIFAR-10-C



(b) CIFAR-100-C

Figure 5.11: **Aggregated results for selective classification on CIFAR-10-C and CIFAR-100-C**. Comparative performance in AURC (%) of classification with the option to reject misclassified test samples and samples from shifted distributions. Results are averaged on 5 runs (mean ± std.).

cost of 1, whatever the prediction. As for simultaneous detection, we rely on oversampling to mitigate the unbalance between misclassifications and OOD samples.

Experiments are conducted with previously trained VGG-16 networks on CIFAR-100. We measure their selective classification when subject to distribution shifts by considering CIFAR-100C [176] as OOD dataset. This dataset is constructed by corrupting the original CIFAR-100 test set. There is a total of 15 types of corruptions, which can be grouped into four families, namely *noise*, *blur*, *weather* and *digital*. Each corruption comes with five different levels of severity. While this dataset is commonly used to measure robustness to distribution shift, we focus here on models' ability to reject these samples along with misclassifications made on the original CIFAR-100 test set.

The results are reported by corruption families (noise, blur, weather and digital) in Section 5.5.3 and further detailed in Section B.2.3. One common observation regardless of the criterion is that selective classification is harder when subject to noise perturbations than other types of perturbation. In each case, KLoSNet and ConfidNet obtain the best performances. For instance, for weather

perturbations on CIFAR-10-C, KloSNet achieves 42.7% AURC and ConfidNet 43.4% AURC. In particular, KloSNet outperforms every other method for blur, weather and digital perturbations of CIFAR-100-C. Hence, when subject to an unforeseen distribution shift, a model equipped with KLoSNet provides more accurate uncertainty estimates without sacrificing predictive performances. Note that for noise corruptions, the results depend widely on the run, which makes interpretation more difficult.

5.5.4 Impact of Adversarial Perturbations

In the original papers, ODIN and Mahalanobis preprocess inputs by adding small inverse adversarial perturbations to reinforce networks in their prediction; this has also the observed benefit to make in-distribution and out-of-distribution samples more separable. The tuning of the adversarial noise's magnitude depends on the evaluated OOD data.

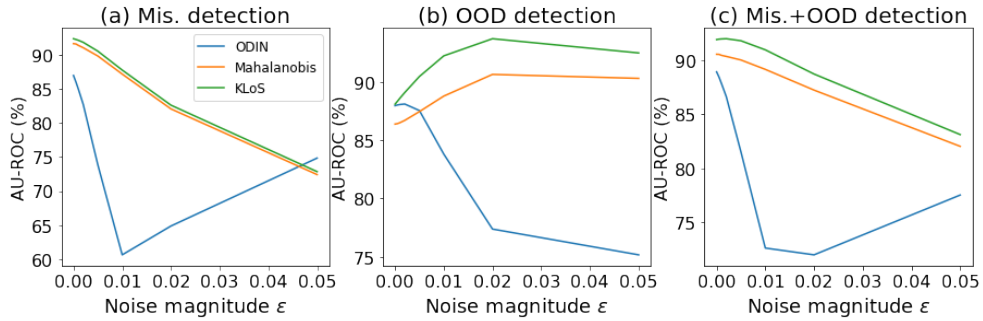
In Fig. 5.12a, we plot the AUC of each detection task with different values of perturbation magnitude ε with ODIN, Mahalanobis and KLoS, using SVHN as OOD dataset. Even though there exists a particular noise value for improved OOD detection (Fig. 5.12a, middle), increasing noise magnitude deteriorates performances in misclassification detection (Fig. 5.12a, left) for each method. The best results on the simultaneous detection task (Fig. 5.12a, right) correspond to $\varepsilon = 0$, as done in experiments presented in previous sections.

Except with SVHN, adversarial perturbations are detrimental even to OOD detection. We report the AUC results of varying adversarial perturbations on CIFAR-10 dataset when using LSUN (Fig. 5.12b), TinyImageNet (Fig. 5.12c) and STL-10 (Fig. 5.12d) as OOD datasets. The best results on each considered task correspond to $\varepsilon = 0$ and KLoS outperforms both Mahalanobis and ODIN. As opposed to results with SVHN as OOD dataset, we did not observe improvements on any method (ODIN, Mahalanobis and KLoS) when using inverse adversarial perturbations for OOD detection with LSUN, TinyImageNet and STL-10 datasets. Similar results are observed in [112] (Appendix B, Fig. 8) when using WideResNet architectures.

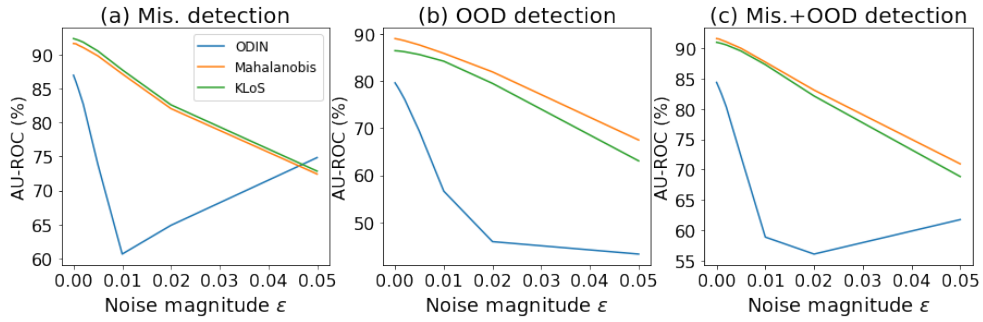
5.5.5 Effect of training with out-of-distribution samples

Previous results demonstrate that simultaneous detection of misclassifications and OOD samples can be significantly improved by KLoSNet. We now investigate settings where OOD samples are available. We train an evidential model to minimize the reverse KL divergence [135] between the model output and a sharp Dirichlet distribution focused on the predicted class for in-distribution samples, and between the model output and a uniform Dirichlet distribution for OOD samples. This loss induces low concentration parameters for OOD data and improves second-order uncertainty measures such as Mut. Inf

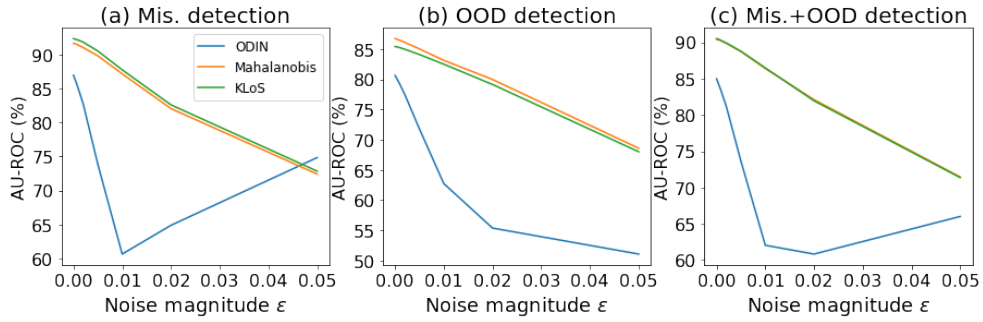
The literature on evidential models only deals with an OOD training set somewhat related to the in-distribution dataset, *e.g.* CIFAR-100 for models trained on CIFAR-10. Despite semantic differences, CIFAR-10 and CIFAR-100 images were collected the same way, which might explain the generalisation to other OOD samples in evaluation.



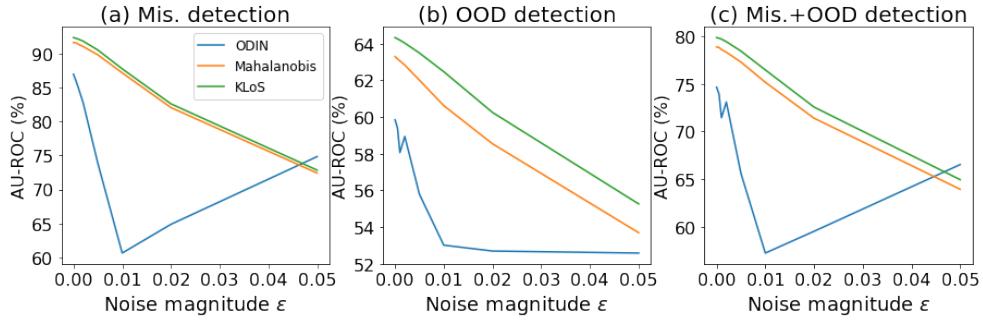
(a) CIFAR-10 / SVHN



(b) CIFAR-10 / LSUN



(c) CIFAR-10 / TinyImageNet



(d) CIFAR-10 / STL-10

Figure 5.12: Effect of inverse adversarial perturbations on OOD-designed measures and KLoS for misclassification detection, OOD detection and simultaneous detection with VGG-16 architecture.

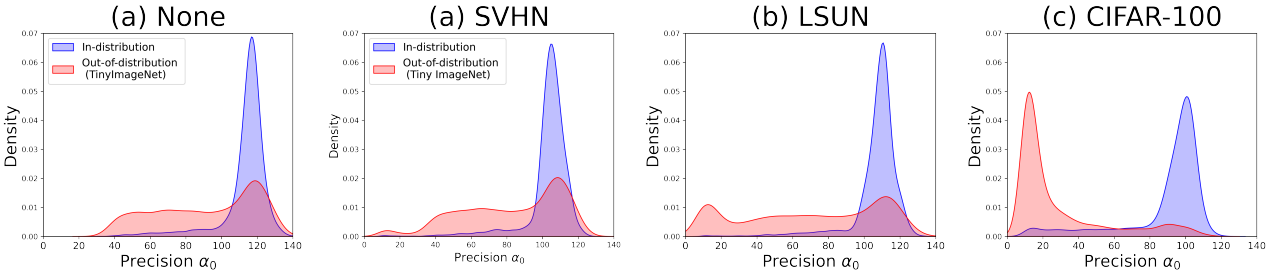
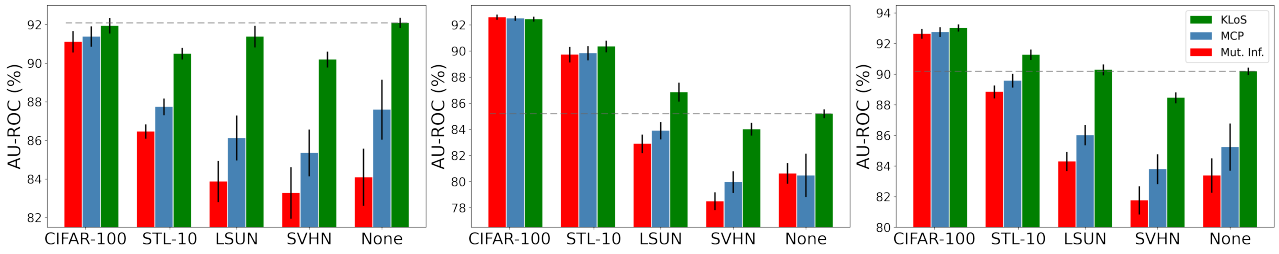


Figure 5.13: **Effect of OOD training data on precision α_0.** Density plots for CIFAR-10/TinyImageNet benchmark: (a) without OOD training data, (b,c) with inappropriate OOD samples (SVHN, LSUN); (d) with close OOD samples (CIFAR-100).

Contrarily, CIFAR-10 objects and SVHN street-view numbers differ more for instance. In Fig. 5.14, we vary the OOD training set and compare the uncertainty metrics taken from the resulting models.



(a) Mis. detection (b) OOD detection (c) Mis.+OOD detection

Figure 5.14: **Comparative detection results with different OOD training datasets.** While using OOD samples in training improves performance in general, the gain varies widely, sometimes even being negative for inappropriate OOD samples *e.g.*, SVHN. KLoS remains the best measure in every setting. Experiment with VGG-16 architecture on CIFAR-10 dataset.

As expected, using CIFAR-100 as training OOD data improves performance for every measure (MCP, Mut. Inf. and KLoS). However, the boost provided by training with OOD samples depends highly on the chosen dataset: The performance of Mut. Inf. decreases from 92.6% AUC with CIFAR-100 to 82.9% when switching to LSUN, and even becomes worse with SVHN (78.5%) compared to using no OOD data (80.6%). Indeed, Fig. 5.13 shows that only the CIFAR-100 dataset seems to be effective to enforce low α_0 on unseen OOD samples.

We also note that KLoS outperforms or is on par with MCP and Mut. Inf. in every setting. These results confirm the adequateness of KLoS for simultaneous detection and extend our findings to settings where OOD data is available at train time. Most importantly, using KLoS on models without OOD training data yields better detection performance than other measures taken from models trained with inappropriate OOD samples, here being every OOD dataset other than CIFAR-100.

5.6 Conclusion

Based on evidential models, we define *KLoS*, a Kullback–Leibler divergence criterion defined on the class-probability simplex. By design, KLoS encompasses both class confusion and evidence information, which is necessary for open-world recognition. We adapt our learning confidence approach to evidential models and proposed KLoSNet, an auxiliary model to estimate the uncertainty of a classifier for both in-domain and out-of-domain inputs. KLoSNet is trained to predict the KLoS* value of a prediction. Our experiments extensively demonstrate its effectiveness across various architectures, datasets and configurations, and reveal its class-wise divergence-based behavior. We also show that, far from being the panacea, using training OOD samples depends critically on the choice of these samples for existing uncertainty measures. KLoS, on the other hand, is more robust to this choice and can alleviate their use altogether.

Chapter 6

Conclusion and Perspectives

We first summarize the contributions that we proposed in this thesis before discussing interesting directions for future work.

6.1 Summary of contributions

The main contribution of this thesis is to use an auxiliary confidence model to learn prediction confidence from deep neural networks in classification. Given a trained classification model, the confidence model learns from data to estimate an adequate criterion derived from the classifier, such as the true class probability for standard neural networks and KLoS for evidential models. At test time, we directly use the confidence model's output as our uncertainty estimate. One major benefit of this method is to be architecture-agnostic: in our experiments, we successfully improve uncertainty estimation for classification models with different deep learning architectures (MLP, LeNet, VGGs, ResNets). We applied our approach on three tasks: failure prediction (Chapter 3), unsupervised domain adaptation for semantic segmentation (Chapter 4), and simultaneous detection of in-distribution errors and out-of-distribution samples (Chapter 5). For each task, there are two main challenges to address: (1) which criterion should we use, and (2) how to efficiently train the confidence model.

Failure prediction with learned confidence. Chapter 3 starts by detailing the fundamental limit of maximum class probability (MCP), which yields over-confident uncertainty estimates for misclassifications. We define the true class probability (TCP) as an alternative measure which provides a better ranking between correct predictions and misclassifications than MCP. As the true class is unknown at test time, we introduce *ConfidNet*, an auxiliary confidence neural network trained to learn TCP from data. ConfidNet consists in a small decoder neural network composed of several dense layers and initially sharing the same ConvNet encoder as the classification model. ConfidNet's learning scheme consists in training the auxiliary network to regress TCP values and then enabling the fine-tuning of its encoder by decoupling it from the classification' encoder. We were able to improve the capacity of the model to distinguish correct from erroneous samples and to achieve better selective

classification with many different architectures and for each image classification experiment. In the long history of classifiers with a reject option, our contribution on learning a model's confidence can be seen as a specific case of selective classification, where the selection function is based on an independent neural network to define the underpinning confidence-rate.

Selection of confident pseudo-labels for domain adaptation. Chapter 4 shows that reliable confidence estimates are key to improve self-training approaches in domain adaptation. We transpose the idea of learning confidence via an auxiliary model and we select relevant pixels for pseudo-labeling based on confidence estimates output by this auxiliary model. In a manner analogous to ConfidNet, we learn to regress to TCP from training data. The proposed adaptation of our original approach to this new context, termed *ConDA*, involves an 'atrous' pyramidal pooling architecture with structured output to perform multi-scale confidence estimation and we adopt an adversarial learning scheme which enforces alignment between confidence maps in source and target domains. Results showed significant improvements from strong baselines in each benchmark.

Detecting errors and out-of-distribution samples with evidential models. Finally, we extend our learning confidence via auxiliary models to the context of simultaneous detection of in-distribution errors and out-of-distribution samples. It first requires defining a criterion that captures aleatoric and epistemic uncertainty in a single score. As a Bayesian approach, evidential models enrich uncertainty representation with evidence information and allows one to fulfill the previous requirement by deriving second-order measures on the class-probability simplex. Consequently, we defined *KLoS*, a KL divergence criterion between a model's output and a class-wise prototype Dirichlet distribution focused on the predicted class. By design, KLoS encompasses both class confusion and evidence information, thanks to its class-wise divergence-based behavior. An auxiliary model, *KLoSNet*, is then trained to predict a refined criterion, KLoS*, measuring KL divergence with a prototype based on the true class of an input. Across various architectures, datasets and configurations, KLoSNet improves performance on the joint detection and reveals itself to be more robust to the type of OOD samples in scenarios allowing this type of auxiliary training data.

6.2 Perspectives for future work

Let us now discuss interesting directions that could be addressed in future work in relation to our contributions.

6.2.1 Error data generation to ease confidence learning

Confidence learning showed significant improvements over strong baselines in uncertainty estimation for each considered work. Nevertheless, the training of the auxiliary model depends on the quality of the dataset, *i.e.* the number of errors available. Modern neural networks are over-parameterized and tend to over-fit training data, hence achieving high accuracy on training sets and

leaving only a small fraction of misclassified samples. We believe this data imbalance issue mitigates performance in confidence learning. In Chapter 3, we experimented training ConfidNet on a hold-out dataset. We observed a general performance drop when using a validation set for training TCP confidence. The drop is especially pronounced for small datasets (MNIST). Consequently, with a high accuracy and a small validation set, we do not get a larger absolute number of errors using the validation set compared to the train set. In preliminary experiments, we also tried a weighted MSE loss (and a weighted binary cross-entropy) where the cost of wrong TCP estimates were higher for misclassifications than for correct predictions. But it did not result in improved performance either.

To mitigate the imbalance issue, another solution would be to artificially generate errors. Adversarial perturbations [114, 131] are small perturbations to the input that are almost imperceptible to humans but which fool a neural network, hence switching a correct prediction to an erroneous one. A combined set of genuine and adversarial inputs would help to re-balanced training. Mix-up [177], and more generally aggressive data augmentation techniques such as AugMix [178] and CutMix [179] have been shown to improve robustness and could be also applied here to generate samples with mixed probabilities, hence providing a larger range of TCP values for confidence training.

6.2.2 Confidence learning of an ensemble

As a simple alternative to fully Bayesian methods, ensembles have been a popular research topic within probabilistic methods [180, 181, 53]. With deep neural networks, not only they improved generalization but they also outperformed other Bayesian methods and a single model in uncertainty estimation [54]. In particular, diversity between individual NN's predictions allows an ensemble to better capture epistemic uncertainty: the more diverse predictions are, the more the model is uncertain about this input. Measures such as mutual information can be derived to evaluate this diversity. Yet, when it comes to failure prediction, previous methods rely on averaging the predictions into a single probability vector and deriving usual measures such as MCP and entropy.

Chapter 3 proposed to improve failure prediction only on a single model as the auxiliary confidence model should be initially attached to an intermediate representation of the classifier. An interesting direction should be to combine the idea of confidence learning to the context of ensembles. The straightforward solution would be to train an auxiliary model to regress the TCP value of the average probability vector. But preliminary experiments showed difficulties in converging to regress such averaged values as each individual model behaves differently and we cannot rely on initialized weights from one arbitrary model. One could try to attach an auxiliary model to each member of the ensemble and train them separately before averaging outputs of all confidence models. A clever approach would imply leveraging the diversity in predictions to refine the criterion to estimate during confidence learning.

6.2.3 Generative models for out-of-distribution detection

In Chapter 5, we highlighted the class-wise density estimator behaviour of KLoS, which is a crucial property in the absence of OOD training data to improve the simultaneous detection of

misclassifications and OOD samples. Along with this contribution, our experiments also revealed that while performance of existing uncertainty measures are considerably improved by using training OOD samples, it also critically depends on the choice of these samples (Section 5.5.5).

Among methods using OOD samples when training deep classifier, Hendrycks *et al.* [115] propose to learn to classify in-distribution samples while producing high predictive entropy for OOD samples:

$$\mathcal{L}_{\text{OE}}(\boldsymbol{\theta}, \mathcal{D}) = \mathbb{E}_{(\boldsymbol{x},y)\sim p_{\text{in}}}\big[\log p(y|\boldsymbol{x}, \boldsymbol{\theta})\big] + \lambda \mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_{\text{out}}}\big[\mathbb{H}[p(y|\tilde{\boldsymbol{x}}, \boldsymbol{\theta})]\big]. \tag{6.1}$$

Accordingly, they use predictive entropy to discriminate between in-distribution samples and OOD samples. It relies on the availability of a large OOD dataset, for instance 80 Million Tiny Images[1] with CIFAR-10 or CIFAR-100 as in-distribution dataset. With evidential models, multiple works [57, 135, 171] proposed a similar approach by enforcing OOD samples to have low precision α_0:

$$\mathcal{L}_{\text{RKL}}(\boldsymbol{\theta}; \mathcal{D}) = \mathbb{E}_{(\boldsymbol{x},y)\sim p_{\text{in}}}\big[\text{KL}(\text{Dir}(\boldsymbol{\pi}|\alpha(\boldsymbol{x}, \boldsymbol{\theta}) \parallel \text{Dir}(\boldsymbol{\pi}|\beta_{\text{in}}))\big] + \lambda \mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_{\text{out}}}\big[\text{KL}(\text{Dir}(\boldsymbol{\pi}|\alpha(\boldsymbol{x}, \boldsymbol{\theta}) \parallel \text{Dir}(\boldsymbol{\pi}|\mathbf{1}))\big]. \tag{6.2}$$

The uncertainty measure used for OOD detection in that case is a dispersion measure, such as mutual information or precision.

Previous methods have been shown to be really effective to improve OOD detection. But while finding suitable OOD samples may be easy for some academic datasets, it may turn more problematic in real-world applications [116, 117], with the risk of degrading performance with an inappropriate choice.

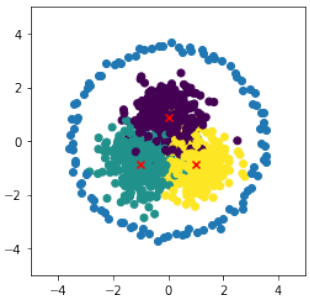


Figure 6.1: Illustration of ideal OOD training data on a toy dataset.

Building a suitable OOD training set for real tasks is an open research perspective which could alleviate the need for real but hard-to-find OOD samples. To ensure good generalization to other types of anomalies, the main challenge is to produce samples which are close to the in-distribution, even at the boundary such as for the toy dataset shown in Fig. 6.1. Generative models such as proposed in [182, 117, 183] could be an interesting solution to fulfill this problem.

6.2.4 Further applications of confidence learning

In an analogous way to Chapter 4, the confidence learning approach could be applied to new contexts where the quality of uncertainty estimates is crucial.

Application to semi-supervised learning. Unfortunately, the efficacy of deep neural networks depends on large quantities of accurately labeled training data. But the labeling process usually requires arduous and expensive efforts, which is one of the major limitations to train a fully-supervised deep neural network. If only a few labeled samples are available, it is challenging to build a successful ML system. In contrast, unlabeled data is usually abundant and can be easily or inexpensively obtained.

[1] https://groups.csail.mit.edu/vision/TinyImages/

Semi-supervised learning (SSL) [184] is a learning paradigm that aims to improve learning performance from labeled data by using additional unlabeled instances. A family of approaches for SSL [37, 163] proposed to infer pseudo-labels on unlabeled data and to re-train a network using these pseudo-labels. The key is to select the most confident labels. Obviously, due to the close connection between domain adaptation and semi-supervised learning methods, a natural extension to ConDA is to apply it in the latter context.

Application to active learning. An alternative to semi-supervised learning is active learning in which data are actively sampled to be labeled by human oracles with the goal of maximizing model performance while minimizing labeling costs. Various sampling strategies have been proposed for active learning over the years coming from different perspectives, e.g. uncertainty [39] and representativeness [185]. Uncertainty-based methods are based on measures derived from the probability vector, such as entropy, MCP or margin sampling. Confidence learning could improve the selection of useful samples, such as done in a related work where the authors aim to learn the loss value [186].

Application to multi-modal fusion. The joint operation of several types of sensors is a key part of autonomous driving systems. Currently, the majority of systems are based on a late fusion due to safety reasons such as redundancy but also due to technical (vehicle network architecture) or commercial (use of several suppliers) constraints. As mentioned in Chapter 1, early multi-modal fusion could benefit from reliable uncertainty estimates. A system could rely more on a certain sensor or discard predictions from other sensors due to a low confidence estimate. On a related topic, Kendall *et al.* [187] showed that multi-task learning can be improved by weighting each task by a task-dependent uncertainty estimate. While this approach was developed for multi-output, it could be adapted in the scenario of multi-input and with confidence learning.

6.2.5 From uncertainty estimation to robustness

In this thesis, we focused on deriving reliable uncertainty estimates to ensure proper monitoring of deployed ML systems in real-world tasks. Alternatively, a whole part of AI safety literature aims to improve the robustness of deep neural networks to distribution shift. Mismatches in data distributions in test time compared to training time can cause a surprisingly large drop in predictive performance [26]. Robustness to distribution shifts has recently been an increasingly popular topic among machine learning researchers [188, 178, 189, 190]. For instance, ImageNet-trained models have been shown to lack robustness against common corruptions [176], adversarial inputs [114], change in renditions [191] (*e.g.* painting, embroidery, etc.). In contrast to the out-of-distribution samples studied in Chapter 5, all of these input manipulations do not change the semantic content of the input, and thus, machine learning models should not change their decision-making behavior in their presence. Consequently, the field is also sometimes referred to *out-of-distribution generalization* [192]. A long-term perspective for this thesis is to leverage the tools developed for uncertainty estimation to improve out-of-distribution generalization.

Bibliography

[1] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, pp. 386–408, 1958. 1

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 30, 2017. 1

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015. 1

[4] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 25, 2012. 1, 2, 14, 64, 117

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 117

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision (ECCV)*, 2016. 1, 117

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 834–848, 2018. 1, 15, 62, 64, 117, 123

[8] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model." in *Proc. Interspeech*, 2010. 1, 117

[9] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019. 1, 117

[10] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, 2012. 1, 117

[11] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, "Deep speech: Scaling up end-to-end speech recognition," 2014. 1, 117

[12] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 2016. 1, 117

[13] J. Hui. Real-time object detection with yolo, yolov2 and now yolov3. Medium. [Online]. Available: https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088 2

[14] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017. 2, 64, 123

[15] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *Arxiv*, 2016. 2

[16] M. García-Martínez, L. Barrault, and F. Bougares, "Factored neural machine translation," 2016. 2

[17] G. Kohs. Alphago - the movie. DeepMind. [Online]. Available: https://www.youtube.com/watch?v=WXuK6gekU1Y 2

[18] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art." *Foundations and Trends in Computer Graphics and Vision*, 2017. 1

[19] A. Ouaknine, A. Newson, P. Pérez, F. Tupin, and J. Rebut, "Multi-view radar semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1

[20] G. Puy, A. Boulch, and R. Marlet, "FLOT: Scene Flow on Point Clouds Guided by Optimal Transport," in *European Conference on Computer Vision (ECCV)*, 2020. 1

[21] J. Varghese and R. G. Boone, "Overview of autonomous vehicle sensors and systems," in *Proceedings of the International Conference on Operations Excellence and Service Engineering (IEOM)*, 2015. 1

[22] M. Toromanoff, É. Wirbel, and F. Moutarde, "End-to-end model-free reinforcement learning for urban driving using implicit affordances," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[23] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *CoRR*, vol. abs/1606.06565, 2016. 2

[24] T. G. J. Rudner and H. Toner, "Key concepts in ai safety: An overview," Center for Security and Emerging Technology, Tech. Rep., 2021. 2

[25] D. Hendrycks, N. Carlini, J. Schulman, and J. Steinhardt, "Unsolved problems in ml safety," *ArXiv*, vol. abs/2109.13916, 2021. 2

[26] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, S. Beery, J. Leskovec, A. Kundaje, E. Pierson, S. Levine, C. Finn, and P. Liang, "Wilds: A benchmark of in-the-wild distribution shifts," in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021. 2, 5, 11, 89, 99, 117

[27] D. Anguelov. Taming the long tail of autonomous driving challenges. Waymo. [Online]. Available: https://www.youtube.com/watch?v=Q0nGo2-y0xY 2

[28] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 31, 2017. 2

[29] Y. Li, X. Xu, J. Xiao, S. Li, and H. T. Shen, "Adaptive square attack: Fooling autonomous cars with adversarial traffic signs," *IEEE Internet of Things Journal*, vol. 8, pp. 6337–6347, 2021. 2

[30] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, vol. 2, pp. 665–673, 2020. 3, 5, 117

[31] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316 3

[32] N. T. S. Board, "Collision between a car operating with automated vehicle control systems and a tractor-semitrailer truck near williston, florida, may 7, 2016," National Highway Traffic Safety Administration, Tech. Rep., 2017. [Online]. Available: https://www.ntsb.gov/investigations/accidentreports/reports/har1702.pdf 3, 117

[33] ——, "Collision between vehicle controlled by development automated driving system and pedrestrain, tempe, arizona, march 18, 2018," National Highway Traffic Safety Administration, Tech. Rep., 2019. [Online]. Available: https://www.ntsb.gov/investigations/accidentreports/reports/har1903.pdf 3

[34] R. McAllister, Y. Gal, A. Kendall, M. van der Wilk, A. Shah, R. Cipolla, and A. Weller, "Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017. 3, 117

[35] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009. 5, 43, 44, 86, 126, 135

[36] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 5, 37, 43, 44, 86, 118, 120, 135, 136

[37] D.-H. Lee, "Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks," *ICML Workshop on Challenges in Representation Learning*, 2013. 3, 58, 99, 129

[38] Y. Li, L. Yuan, and N. Vasconcelos, "Bidirectional learning for domain adaptation of semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 56, 57, 58, 64, 66, 67, 68, 121

[39] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017. 4, 11, 99, 129

[40] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016. 4, 5, 17, 20, 22, 42, 44, 46, 74, 117, 120, 124

[41] C. Chow, "An optimum character recognition system using decision functions," *IRE Trans. Electron. Comput.*, 1957. 4, 24, 34, 42, 117, 118

[42] H. Zaragoza and d. Buc, "Confidence measures for neural network classifiers," in *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, 1998. 4, 42, 117

[43] J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, and N. Ueffing, "Confidence estimation for machine translation," in *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, 2004. 4, 42, 117

[44] R. El-Yaniv and Y. Wiener, "On the foundations of noise-free selective classification," *Journal of Machine Learning Research*, vol. 11, no. 53, pp. 1605–1641, 2010. 4, 24, 34, 36, 42, 46, 89, 117, 118

[45] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *International Conference on Learning Representations (ICLR)*, 2017. 4, 25, 26, 34, 42, 44, 46, 85, 117, 120

[46] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4, 5, 42, 117, 118

[47] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2017. 4, 15, 44, 117, 120

[48] M. Hein, M. Andriushchenko, and J. Bitterwolf, "Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 4, 5, 42, 117

[49] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017. 4, 26, 27, 43, 50, 117

[50] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations (ICLR)*, 2014. 5, 118

[51] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, pp. 452–459, 2015. 5

[52] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, "A simple baseline for Bayesian uncertainty in deep learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019. 5, 42, 74, 124

[53] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 30, 2017. 5, 17, 27, 42, 74, 88, 97, 124, 128

[54] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019. 5, 17, 74, 88, 97, 124

[55] A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov, "Pitfalls of in-domain uncertainty estimation and ensembling in deep learning," in *International Conference on Learning Representations (ICLR)*, 2020. 5, 17, 26

[56] M. Sensoy, L. Kaplan, and M. Kandemir, "Evidential deep learning to quantify classification uncertainty," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018. 6, 18, 74, 78, 124, 125

[57] A. Malinin and M. Gales, "Predictive uncertainty estimation via prior networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018. 6, 10, 18, 74, 78, 98

[58] D. V. Lindley, *The Philosophy of Statistics*. Journal of the Royal Statistical Society: Series D (The Statistician), 2000. 10

[59] F. H. Knight, *Risk, Uncertainty and Profit.* Houghton Mifflin Co, 1921. 10

[60] J. R. Benjamin and C. A. Cornell, *Probability, statistics, and decision for civil engineers.* McGraw-Hill,, 1970. 10

[61] A. D. Kiureghian and O. Ditlevsen, "Aleatory or epistemic? does it matter?" *Structural Safety*, vol. 31, pp. 105–112, 2009. 10

[62] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in Neural Information Processing Systems*, vol. 30, 2017. 10, 42

[63] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods," *Machine Learning*, vol. 110, pp. 457–506, 2021. 10, 12, 16

[64] R. Senge, S. Bösner, K. Dembczyński, J. Haasenritter, O. Hirsch, N. Donner-Banzhoff, and E. Hüllermeier, "Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty," *Information Sciences*, vol. 255, pp. 16–29, 2014. 10

[65] O. Zendel, K. Honauer, M. Murschitz, D. Steininger, and G. F. Dominguez, "Wilddash - creating hazard-aware benchmarks," in *European Conference on Computer Vision (ECCV)*, 2018. 11

[66] M. Kull and P. A. Flach, "Reliability maps: A tool to enhance probability estimates and improve classification accuracy," in *Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, T. Calders, F. Esposito, E. Hüllermeier, and R. Meo, Eds., 2014. 10

[67] K. Hornik, M. B. Stinchcombe, and H. L. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989. 13

[68] W. Mcculloch and W. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 127–147, 1943. 13

[69] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986. 14

[70] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *International Conference on Computational Statistics (COMPSTAT)*, 2010. 14

[71] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research (JMLR)*, vol. 12, p. 2121–2159, 2011. 14

[72] M. D. Zeiler, "Adadelta: An adaptive learning rate method," 2012. 14

[73] J. B. Diederik P. Kingma, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015. 14, 64

[74] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015. 14, 15, 44, 86, 126, 135

[75] T. Durand. Deep architectures in latex. [Online]. Available: https://github.com/durandtibo/deep_archi_latex 14

[76] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 15, 86, 126

[77] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 15, 64, 123

[78] D. J. MacKay, "Bayesian methods for adaptive models," Ph.D. dissertation, California Institute of Technology, 1992. 16, 17

[79] R. M. Neal, *Bayesian Learning for Neural Networks*. Springer-Verlag, 1996. 16, 42

[80] ——, "Mcmc using hamiltonian dynamics," in *Handbook of Markov Chain Monte Carlo*, 2011. 17

[81] J. M. Hernandez-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015. 17, 42

[82] P. Jylänki, A. Nummenmaa, and A. Vehtari, "Expectation propagation for neural networks with sparsity-promoting priors," *Journal of Machine Learning Research*, vol. 15, pp. 1849–1901, 2014. 17

[83] G. E. Hinton and D. van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proceedings of the Sixth Annual Conference on Computational Learning Theory (COLT)*, 1993. 17

[84] A. Graves, "Practical variational inference for neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 24, 2011. 17

[85] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015. 17, 42

[86] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. 17

[87] P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. Wilson, "What are bayesian neural network posteriors really like?" in *Proceedings of the 38th International Conference on Machine Learning (ICML*, 2021. 17

[88] F. Wenzel, K. Roth, B. Veeling, J. Swiatkowski, L. Tran, S. Mandt, J. Snoek, T. Salimans, R. Jenatton, and S. Nowozin, "How good is the Bayes posterior in deep neural networks really?" in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020. 17

[89] I. Osband, "Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout," in *NIPS Workshop on Bayesian Deep Learning*, 2016. 17

[90] Y. Liu, M. Pagliardini, T. Chavdarova, and S. U. Stich, "The peril of popular deep learning uncertainty estimation methods," in *NeurIPS Workshop on Bayesian Deep Learning*, 2021. 17

[91] A. G. Wilson and P. Izmailov, "Bayesian deep learning and a probabilistic perspective of generalization," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 18

[92] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get M for free," in *International Conference on Learning Representations (ICLR)*, 2017. 18

[93] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, "A simple baseline for bayesian uncertainty in deep learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019. 18

[94] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2005. 18

[95] G. W. Benton, W. J. Maddox, J. P. Salkey, J. Albinati, and A. G. Wilson, "Function-space distributions over kernels," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019. 18

[96] A. Josang, *Subjective Logic: A Formalism for Reasoning Under Uncertainty*. Springer, 2016. 18, 74, 124

[97] A. P. Dempster, *A Generalization of Bayesian Inference*. Springer, 2008. 18

[98] K. P. Murphy, *Machine learning : A Probabilistic Perspective*. MIT Press, 2012. 18, 78, 82, 125

[99] S. Depeweg, J.-M. Hernandez-Lobato, F. Doshi-Velez, and S. Udluft, "Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018. 22

[100] A. Josang, J.-H. Cho, and F. Chen, "Uncertainty characteristics of subjective opinions," in *International Conference on Information Fusion (FUSION)*, 2018. 23

[101] W. Shi, X. Zhao, F. Chen, and Q. Yu, "Multifaceted uncertainty estimation for label-efficient deep learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020. 23, 75, 125

[102] Y. Geifman, G. Uziel, and R. El-Yaniv, "Bias-reduced uncertainty estimation for deep neural classifiers," in *International Conference on Learning Representations (ICLR)*, 2019. 24, 45

[103] Y. Geifman and R. El-Yaniv, "Selective classification for deep neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 30, 2017. 25, 36, 42, 46

[104] C. Cortes, G. DeSalvo, and M. Mohri, "Learning with rejection," in *Proceedings of The 27th International Conference on Algorithmic Learning Theory (ALT)*, 2016. 25, 42

[105] Y. Geifman and R. El-Yaniv, "Selectivenet: A deep neural network with an integrated reject option," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019. 25, 42

[106] S. Hecker, D. Dai, and L. V. Gool, "Failure prediction for autonomous driving," in *IEEE Intelligent Vehicles Symposium (IV)*, 2018. 25

[107] X. Jiang, M. Osl, J. Kim, and L. Ohno-Machado, "Calibrating predictive model estimates to support personalized medicine," *Journal of the American Medical Informatics Association (JAMIA)*, vol. 19, 2012. 26

[108] J. Vaicenavicius, D. Widmann, C. Andersson, F. Lindsten, J. Roll, and T. Schön, "Evaluating model calibration in classification," in *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019. 26

[109] M. P. Naeini, G. F. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, 2015. 26

[110] A. Bendale and T. Boult, "Towards open world recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 27, 74, 124

[111] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," *arXiv preprint arXiv:2110.11334*, 2021. 27, 28

[112] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *International Conference on Learning Representations (ICLR)*, 2018. 28, 43, 82, 91, 126

[113] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 31, 2018. 29, 82, 126

[114] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015. 29, 30, 97, 99

[115] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," in *International Conference on Learning Representations (ICLR)*, 2019. 29, 98, 128

[116] B. Charpentier, D. Zügner, and S. Günnemann, "Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020. 29, 78, 84, 98, 125, 128, 135

[117] M. Sensoy, L. Kaplan, F. Cerutti, and M. Saleki, "Uncertainty-aware deep classifiers using generative models," in *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020. 29, 78, 98, 125, 128

[118] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations (ICLR)*, 2014. 29

[119] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?" in *International Conference on Learning Representations (ICLR)*, 2019. 29

[120] L. Dinh, D. Krueger, and Y. Bengio, "NICE: non-linear independent components estimation," in *International Conference on Learning Representations (ICLR)*, 2015. 29

[121] A. van den Oord, N. Kalchbrenner, L. Espeholt, k. kavukcuoglu, O. Vinyals, and A. Graves, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 29, 2016. 29

[122] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications," in *International Conference on Learning Representations (ICLR)*, 2017. 29

[123] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky, "Your classifier is secretly an energy based model and you should treat it like one," in *International Conference on Learning Representations (ICLR)*, 2020. 29

[124] W. Liu, X. Wang, J. Owens, and Y. Li, "Energy-based out-of-distribution detection," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020. 29, 82

[125] Y. LeCun, S. Chopra, R. Hadsell, F. J. Huang, and et al., "A tutorial on energy-based learning," in *Predicting structured data*, 2006. 29

[126] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics." in *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2011. 29

[127] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *IEEE European Symposium on Security and Privacy*, 2016. 30

[128] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," 2017. 30

[129] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Sympsosium on Security and Privacy*, 2017. 30

[130] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 30

[131] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*, 2018. 30, 97

[132] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2018. 30

[133] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," in *International Conference on Learning Representations (ICLR)*, 2019. 30

[134] N. Carlini and D. A. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (ACM CCS)*, 2017. 30

[135] A. Malinin and M. Gales, "Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019. 30, 75, 77, 78, 83, 91, 98, 124, 125, 126, 135

[136] J. Gilmer and D. Hendrycks, "A discussion of 'adversarial examples are not bugs, they are features': Adversarial example researchers need to expand what is meant by 'robustness'," *Distill*, 2019. 31

[137] J. Nam, S. Park, E. J. Hwang, J. Lee, K.-N. Jin, K. Lim, T. Vu, J. Sohn, S. Hwang, J. M. Goo, and C. M. Park, "Development and validation of deep learning–based automatic detection algorithm for malignant pulmonary nodules on chest radiographs," *Radiology*, 2018. 34

[138] O. Linda, T. Vollmer, and M. Manic, "Neural network based intrusion detection system for critical infrastructures," in *International Joint Conference on Neural Networks (IJCNN)*, 2009. 34

[139] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 1948. 35

[140] P. Mohapatra, M. Rolínek, C. Jawahar, V. Kolmogorov, and M. Pawan Kumar, "Efficient optimization for rank-based loss functions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 41

[141] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 41, 48

[142] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Workshop on Deep Learning and Representation Learning*, 2015. 41

[143] P. L. Bartlett and M. H. Wegkamp, "Classification with a reject option using a hinge loss," *Journal of Machine Learning Research*, vol. 9, no. 59, pp. 1823–1840, 2008. 42

[144] C. Cortes, G. DeSalvo, and M. Mohri, "Boosting with abstention," in *Advances in Neural Information Processing Systems (NIPS)*, 2016. 42

[145] L. P. Cordella, C. De Stefano, F. Tortorella, and M. Vento, "A method for improving classification reliability of multilayer perceptrons," *IEEE Transactions on Neural Networks*, 1995. 42

[146] H. Jiang, B. Kim, M. Guan, and M. Gupta, "To trust or not to trust a classifier," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 31, 2018. 42, 120

[147] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" in *International Conference on Database Theory (ICDT)*, 1999. 43

[148] T. DeVries and G. W. Taylor, "Learning confidence for out-of-distribution detection in neural networks," 2018. 43

[149] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998. 43, 44, 120

[150] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *European Conference on Computer Vision (ECCV)*, 2008. 43, 120

[151] H. Jiang, B. Kim, M. Guan, and M. Gupta, "To trust or not to trust a classifier," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, vol. 31. 44, 46

[152] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision (ECCV)*, 2016. 56, 64, 121, 123

[153] J. Hoffman, D. Wang, F. Yu, and T. Darrell, "Fcns in the wild: Pixel-level adversarial and constraint-based adaptation," *CoRR*, vol. abs/1612.02649, 2016. [Online]. Available: http://arxiv.org/abs/1612.02649 56, 57, 121

[154] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker, "Learning to adapt structured output space for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 56, 57, 58, 64, 65, 67, 68, 123

[155] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, "Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 56, 57, 58, 64, 65, 66, 67, 68, 69, 123

[156] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, "Cycada: Cycle consistent adversarial domain adaptation," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018. 56, 57, 67, 121

[157] Y. Zou, Z. Yu, X. Liu, B. V. Kumar, and J. Wang, "Confidence regularized self-training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 56, 58, 67, 68, 121

[158] Y. Zou, Z. Yu, B. Vijaya Kumar, and J. Wang, "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *European Conference on Computer Vision (ECCV)*, 2018. 56, 58, 67, 68, 121

[159] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Perez, "Dada: Depth-aware domain adaptation in semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. 57, 64, 65, 66, 69, 123

[160] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015. 57

[161] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *European Conference on Computer Vision (ECCV)*, 2016. 57

[162] W.-L. Chang, H.-P. Wang, W.-H. Peng, and W.-C. Chiu, "All about structure: Adapting structural information across domains for boosting semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 58, 67, 68

[163] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 17, 2004. 58, 99, 129

[164] A. Saporta, T.-H. Vu, M. Cord, and P. Pérez, "Esl: Entropy-guided self-supervised learning for domain adaptation in semantic segmentation," in *CVPR Workshop on Scalability in Autonomous Driving*, 2020. 58, 67, 68, 69

[165] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 64, 123

[166] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 64

[167] G. Volk, S. Müller, A. v. Bernuth, D. Hospach, and O. Bringmann, "Towards robust cnn-based object detection through augmentation with synthetic rain variations," in *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019. 74, 124

[168] D. Dai and L. V. Gool, "Dark model adaptation: Semantic image segmentation from daytime to nighttime," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018. 74, 124

[169] M. A. Alcorn, Q. Li, Z. Gong, C. Wang, L. Mai, W.-S. Ku, and A. Nguyen, "Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 74, 124

[170] T. Joo, U. Chung, and M.-G. Seo, "Being bayesian about categorical probability," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020. 75, 77, 78, 124

[171] J. Nandy, W. Hsu, and M. Lee, "Towards maximizing the representation gap between in-domain and out-of-distribution examples," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020. 75, 78, 98, 124, 135

[172] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, University of Cambridge, 2016. 82

[173] A. Shafaei, M. Schmidt, and J. J. Little, "A less biased evaluation of out-of-distribution sample detectors," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2019. 82

[174] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015. 86, 135

[175] A. Rame and M. Cord, "Dice: Diversity in deep ensembles via conditional redundancy adversarial estimation," in *International Conference on Learning Representations (ICLR)*, 2021. 88

[176] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *International Conference on Learning Representations (ICLR)*, 2019. 90, 99

[177] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018. 97, 128

[178] D. Hendrycks*, N. Mu*, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, "Augmix: A simple method to improve robustness and uncertainty under data shift," in *International Conference on Learning Representations*, 2020. 97, 99, 128

[179] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. 97, 128

[180] T. G. Dietterich, "Ensemble methods in machine learning," in *International Workshop on Multiple Classifier Systems*, 2000. 97, 128

[181] L. Rokach, "Ensemble-based classifiers," *Artif. Intell. Rev.*, vol. 33, pp. 1–39, 2010. 97, 128

[182] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," in *International Conference on Learning Representations (ICLR)*, 2018. 98, 128

[183] S. Vernekar, A. Gaurav, V. Abdelzad, T. Denouden, R. Salay, and K. Czarnecki, "Out-of-distribution detection in classifiers via generation," in *NeurIPS Workshop on Safety and Robustness in Decision Making*, 2019. 98, 128

[184] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*. The MIT Press, 2006. 99, 129

[185] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," in *International Conference on Learning Representations*, 2018. 99, 129

[186] D. Yoo and I. S. Kweon, "Learning loss for active learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 99, 129

[187] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 99

[188] E. Rusak, L. Schott, R. Zimmermann, J. Bitterwolf, O. Bringmann, M. Bethge, and W. Brendel, "A simple way to make neural networks robust against diverse image corruptions," in *European Conference on Computer Vision (ECCV)*, 2020. 99

[189] C. Burns and J. Steinhardt, "Limitations of Post-Hoc Feature Alignment for Robustness," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 99

[190] S. Bhojanapalli, A. Chakrabarti, D. Glasner, D. Li, T. Unterthiner, and A. Veit, "Understanding robustness of transformers for image classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 99

[191] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, D. Song, J. Steinhardt, and J. Gilmer, "The many faces of robustness: A critical analysis of out-of-distribution generalization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 99

[192] M. Arjovsky, "Out of distribution generalization in machine learning," Ph.D. dissertation, New York University, 2021. 99

[193] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019. 135

Résumé de la Thèse

Introduction

Depuis la victoire éclatante d'AlexNet [4], une architecture de réseau de neurone convolutif, au Large Scale Visual Recognition Challenge (LSVRC) en 2012, l'apprentissage profond est omniprésent dans les domaines de la vision par ordinateur [5, 6, 7], du traitement du langage naturel [8, 9], de la reconnaissance vocale [10, 11] et de l'apprentissage par renforcement [12]. Les récentes percées en vision par ordinateur grâce à l'apprentissage profond expliquent aussi en grande partie le spectaculaire renouveau de la conduite autonome avec des acteurs technologiques majeurs comme Waymo, Tesla, Baidu et Yandex investissant dans des programmes de voitures autonomes. En tant qu'un des leaders mondiaux des capteurs automobiles, Valeo, qui finance cette thèse, se positionne au cœur de cette révolution actuelle, en développant des LiDAR de haute qualité avec leur technologie SCALA®.

Si ces progrès sont indéniables, les robotaxis ne sont toujours pas déployés au moment de la rédaction de ce manuscrit et de nombreux défis doivent encore être résolus pour commercialiser la voiture autonome à grande échelle, notamment ceux liés à la sécurité. Les accidents survenant avec des voitures autonomes sont des exemples typiques où les répercussions peuvent être catastrophiques. Un exemple frappant est l'incident tragique qui s'est produit le 7 mai 2016 près de Williston (Floride, États-Unis) et qui a entraîné le premier décès causé par une voiture à assistance de conduite hautement automatisée [32]. Le constructeur automobile Tesla a déclaré qu'une des origines de l'accident était liée au système de vision qui avait incorrectement classé le côté blanc d'un camion-remorque comme un ciel éblouissant[2]. Le contrôle et la correct évaluation de la confiance du système dans ses prédictions semblent être plus que nécessaires pour déployer en toute sécurité des modèles d'apprentissage dans des environnements à fort enjeux [34].

L'estimation de la confiance a une longue histoire en apprentissage automatique [41, 42, 43, 44]. Pourtant, une série de travaux récents ont montré que les réseaux de neurones (NNs) modernes souffrent de plusieurs inconvénients conceptuels qui les rendent peu fiables [45, 46, 40, 47, 48]. En classification, ils peinent à détecter les prédictions erronées [45] et produisent des probabilités non calibrées [49]. Les NNs sont également connus pour être fragiles aux changements de distribution [26], leurs performances de prédiction diminuant sévèrement car ils ont tendance à s'appuyer sur des corrélations parasites [30]. Enfin, de nombreux travaux ont montré que les NNs fournissent des prédictions trop confiantes pour des échantillons loins des données d'entraînement [48], y compris des

[2] https://www.tesla.com/blog/tragic-loss

images trompeuses [46] ou adverses [50].

Dans cette thèse, nous relevons le défi de fournir des estimations d'incertitude fiables pour les prédictions des réseaux de neurones profonds avec application en conduite autonome. En particulier, nous cherchons à améliorer la détection des prédictions erronées au moment du test en les distinguant des prédictions correctes. Les erreurs peuvent être de différentes natures et les contributions suivantes aborderont tout d'abord la tâche de détection d'erreur de classification. En plus de la détection de ces exemples au moment du test, nous élaborons également sur l'utilisation de l'approche proposée dans le cas de l'adaptation au domaine, où les approches d'auto-formation s'appuient sur les estimations d'incertitude pour sélectionner les échantillons dans la phase de ré-étiquetage. Enfin, nous considérons la présence d'anomalies et proposons de détecter simultanément les erreurs et les échantillons hors distribution à l'aide d'une seule mesure.

Apprentissage de confiance via un modèle auxiliaire

La prédiction d'échec consiste à prédire à l'exécution si un modèle entraîné a pris une décision correcte ou non pour une entrée donnée. En détectant une prédiction erronée, un système peut décider de s'en tenir à la prédiction ou, au contraire, de la transmettre à un humain ou à un système de secours avec d'autres capteurs, ou simplement de déclencher une alarme. Étroitement liée à la prédiction d'échec, la classification avec option de rejet [41], également connue sous le nom de *classification sélective*. [44], consiste en un scénario où le classifieur a la possibilité de rejeter une instance au lieu de prédire son étiquette. Ces deux tâches renvoient au même problème de *classement ordinal*, qui vise à estimer les valeurs de confiance dont le classement des échantillons est efficace pour distinguer les prédictions correctes des prédictions incorrectes (voir Fig. 3.1).

Une méthode de référence largement utilisée avec les classifieurs de type réseaux de neurones consiste à prendre la valeur de la probabilité de la classe prédite, à savoir la *probabilité de classe maximale* (MCP), donnée par la sortie de la couche softmax :

$$\mathrm{MCP}_F(\boldsymbol{x}) = \max_{k \in \mathcal{Y}} P(Y = k|\boldsymbol{x}, \hat{\boldsymbol{\theta}}) = \max_{k \in \mathcal{Y}} F(\boldsymbol{x}; \hat{\boldsymbol{\theta}})[k]. \tag{6.3}$$

Cependant, en prenant la plus grande probabilité softmax comme estimation de confiance, MCP conduit à des valeurs de confiance élevées à la fois pour les prédictions correctes et erronées, ce qui rend difficile de leur distinction, comme le montre Fig. 3.3a. Prendre l'entropie prédictive comme mesure d'incertitude n'est également pas adéquat. Dans Fig. 3.2, nous montrons côte à côte deux échantillons présentant une entropie similaire, issus d'un petit réseau convolutif entraîné sur SVHN, un jeu de données de numérotation urbaines[36].

Probabilité de la Vrai Classe. Lorsque le modèle classifie mal un exemple, la probabilité associée à la vraie classe y est inférieure à la probabilité maximale et risque d'être petite. Sur la base de cette simple observation, nous proposons de considérer plutôt cette *probabilité de la vrai classe* comme une mesure de confiance appropriée.

$$\mathrm{TCP}_F(\boldsymbol{x},\, y) = P(Y = y|\, \boldsymbol{x}, \hat{\boldsymbol{\theta}}) = F(\boldsymbol{x}; \hat{\boldsymbol{\theta}})[y]. \tag{6.4}$$

Dans Fig. 3.3, nous pouvons observer que TCP permet une bien meilleure séparation que MCP. En particulier, TCP offre les intéressantes garanties suivantes concernant sa capacité à caractériser les prédictions correctes ou erronées d'un modèle.

- $\text{TCP}_F(\boldsymbol{x}, y) > 1/2 \;\Rightarrow f(\boldsymbol{x}) = y$, *i.e.* l'exemple est correctement classé par le modèle ;
- $\text{TCP}_F(\boldsymbol{x}, y) < 1/K \Rightarrow f(\boldsymbol{x}) \neq y$, *i.e.* l'exemple est mal classé par le modèle,

Dans l'intervalle $[1/K, 1/2]$, rien ne garantit que les prédictions correctes et incorrectes ne se chevauchent pas en termes de TCP. Cependant, nous observons qu'avec des réseaux neuronaux profonds, la zone de chevauchement réelle est extrêmement petite en pratique, comme l'illustre Fig. 3.3b sur le jeu de données CIFAR-10. Une explication possible vient du fait que les réseaux neuronaux profonds modernes produisent des prédictions trop confiantes et donc des probabilités non calibrées.

ConfidNet. L'utilisation du TCP comme mesure de confiance sur la sortie d'un modèle serait d'une grande aide lorsqu'il s'agit d'estimer de manière fiable sa confiance. Cependant, la vraie classe y n'est évidemment pas disponible lors de l'estimation de la confiance sur les entrées de test. Nous proposons donc d'*apprendre la confiance TCP à partir des données*. À cette fin, nous introduisons un *modèle auxiliaire* C, avec des paramètres $\boldsymbol{\omega}$, qui est destiné à prédire TCP_F et à agir comme une mesure de confiance pour la fonction de sélection g. Un aperçu de l'approche proposée est disponible dans Fig. 3.4. Ce modèle est entraîné de telle sorte qu'au moment de l'exécution, pour une entrée $\boldsymbol{x} \in \mathcal{X}$ avec l'étiquette vraie (inconnue) y, nous avons :

$$C(\boldsymbol{x}; \boldsymbol{\omega}) \approx \text{TCP}_F(\boldsymbol{x}, y). \tag{6.5}$$

En pratique, ce modèle auxiliaire C est un réseau de neurone entraîné sous supervision complète sur $\mathcal{D}$ pour produire cette estimation de confiance. Pour concevoir ce réseau, nous pouvons transférer les connaissances du réseau de classification déjà entraîné. Nous construisons ConfidNet sur une représentation intermédiaire tardive de F. ConfidNet est conçu comme un petit perceptron multicouche composé d'une succession de couches denses avec une activation sigmoïde finale qui produit $C(\boldsymbol{x}; \boldsymbol{\omega}) \in [0, 1]$. Comme nous voulons régresser un score entre 0 et 1, nous utilisons une fonction de perte d'erreur quadratique moyenne pour entraîner le modèle de confiance :

$$\mathcal{L}_{\text{conf}}(\boldsymbol{\omega}; \mathcal{D}) = \frac{1}{N} \sum_{n=1}^{N} \left(C(\boldsymbol{x}_n; \boldsymbol{\omega}) - \text{TCP}_F(\boldsymbol{x}_n, y_n) \right)^2. \tag{6.6}$$

Nous décomposons les paramètres du réseau de classification F en $\boldsymbol{\theta} = (\boldsymbol{\theta}_E, \boldsymbol{\theta}_{\text{cls}})$, où $\boldsymbol{\theta}_E$ désigne les poids de son encodeur et $\boldsymbol{\theta}_{\text{cls}}$ les poids de ses dernières couches de classification. Comme dans l'apprentissage par transfert, l'apprentissage du réseau de confiance C commence par fixer l'encodeur partagé et n'entraîner que les poids $\boldsymbol{\varphi}$ de ConfidNet. Dans cette phase, la fonction de perte Eq. (3.15) est donc minimisée uniquement par rapport à $\boldsymbol{\omega} = \boldsymbol{\varphi}$. Dans une deuxième phase, nous affinons le réseau complet C, y compris son encodeur qui est maintenant délié de l'encodeur de classification

E (le modèle de classification principal doit rester inchangé, par définition du problème traité). En désignant par E' cet encodeur désormais indépendant, et par $\boldsymbol{\theta}_{E'}$ ses poids, cette seconde phase d'apprentissage optimise Eq. (3.15) en fonction de $\boldsymbol{\omega} = (\boldsymbol{\theta}_{E'}, \boldsymbol{\varphi})$ avec $\boldsymbol{\theta}_{E'}$ initialement fixé à $\boldsymbol{\theta}_E$. Nous désactivons également les couches de dropout dans cette dernière phase d'apprentissage et réduisons la vitesse d'entrainement afin d'atténuer les effets stochastiques qui pourraient amener le nouvel encodeur à trop s'écarter de l'encodeur original utilisé pour la classification. L'augmentation des données peut donc encore être utilisée. ConfidNet peut être entraîné en utilisant soit l'ensemble d'entraînement original, soit un ensemble de validation.

Expériences. Pour démontrer l'efficacité de notre méthode, nous avons implémenté et comparé des approches concurrentes d'estimation de la confiance et de l'incertitude, notamment la probabilité de classe maximale (MCP) comme méthode de référence [45], TrustScore [146] et Monte-Carlo Dropout (MC-Dropout) [40]. Les comparaisons sont effectués sur des jeux de données d'images d'échelle et de complexité variables : MNIST [149], SVHN [36], CIFAR-10 et CIFAR-100. Nous présentons également des expériences de segmentation sémantique sur CamVid [150], un jeu de données standard de scènes de conduite. Les architectures profondes de classification suivent les architectures standarement utilisées en classification d'images, telles que les architectures de perceptron multicouche (MLP), LeNet et VGG16. Pour CamVid, nous avons implémenté un modèle de segmentation sémantique SegNet, suivant [47]. Enfin, nous mesurons la qualité de la prédiction des défaillances en suivant les métriques utilisées dans la littérature [45] : AUPR-Error, AUPR-Success, FPR at 95% TPR et AUROC. Nous nous concentrerons principalement sur l'AUPR-Error, qui calcule l'aire sous la courbe Précision-Rappel en utilisant les erreurs comme classe positive.

Les résultats comparatifs montrent que notre approche surpasse les méthodes usuelles dans toutes les configurations, avec un écart significatif sur les petits modèles et jeux de données. Cela confirme à la fois que le TCP est un critère de confiance adéquat pour la prédiction des défaillances et que notre approche ConfidNet est capable de l'apprendre. Nous fournissons une illustration sur CamVid (Fig. 3.9) pour mieux comprendre notre approche pour la prédiction de défaillance. Par rapport à la méthode de base MCP, notre approche produit des scores de confiance plus élevés pour les prédictions de pixels corrects et plus faibles pour les pixels mal étiquetés, ce qui permet à l'utilisateur de mieux détecter les zones d'erreurs.

Auto-apprentissage avec confiance apprise pour l'adaptation de domaine

Les systèmes de perception des voitures autonomes nécessitent une compréhension approfondie des scènes dans lesquelles ils évoluent. Pour cette raison, des modules de segmentation sémantique sont souvent incorporés afin d'obtenir des prédictions d'étiquettes de classe pour chaque pixel de la scène. Bien que les récents progrès des réseaux convolutifs profonds aient considérablement amélioré les performances de segmentation, leur efficacité dépend de grandes quantités de données d'entraînement étiquetées avec précision. Mais le processus d'étiquetage nécessite généralement

l'intervention d'experts et le coût de l'annotation limite les domaines opérationnels de ces systèmes. D'un autre côté, de nombreuses données de scènes de conduite sont synthétisées par des moteurs de jeux tels que GTA5 [152]. Par conséquent, des travaux récents tentent d'exploiter cette supervision alternative bon marché en entraînant des modèles sur ces sources d'images et en prédisant sur des images réelles. Mais le transfert n'est pas directement efficace car on observe une baisse de performance lors de l'évaluation sur des images réelles, due à un gap entre les domaines.

L'adaptation de domaine non supervisée (UDA) est le domaine de recherche qui vise à réduire cet écart de domaine entre les domaines source et cible. Dans le contexte de l'UDA, des échantillons sources annotés et des images cibles non étiquetées sont disponibles au moment de l'entrainement. La plupart des travaux de cette ligne de recherche visent à minimiser l'écart de distribution entre le domaine source et le domaine cible, au niveau des features extraites ou de la prédiction [153], potentiellement combiné à des méthodes de translation transformant les images sources pour qu'elles correspondent au 'style' [156] du domaine cible. Récemment, l'auto-formation [38, 157, 158] a prouvé sa capacité à augmenter les performances d'adaptation de manière significative. Le principe de ces approches est d'étiqueter automatiquement les pixels cibles les plus confiants selon la prédiction actuelle du réseau et de ré-entrainer le réseau en conséquence. Bien que cette idée soit séduisante, la présence de pseudo-étiquettes avec du bruit ou incorrectes pourrait nuire à l'entrainement du réseau de neurones. À titre d'exemple, l'utilisation d'un ratio de 70% de pseudo-étiquettes dans [38] conduit à une performance d'environ 48% mIoU, ce qui est mieux que 34% avec le transfert direct (uniquement entrainé sur le domaine source), mais toujours largement inférieur aux 63% obtenus avec la même quantité d'étiquettes de terrain. Par conséquent, la définition de bonnes mesures de confiance pour sélectionner des prédictions fiables est d'une importance cruciale pour le développement d'un auto-apprentissage sans erreur.

Pour améliorer l'efficacité de l'auto-formation, nous proposons d'adapter notre approche d'apprentissage de confiance développée dans le chapitre précédent au contexte particulier de l'adaptation non supervisée de domaine pour la segmentation sémantique. Un réseau de confiance C est appris pour prédire la confiance du réseau de segmentation sémantique F entraîné par UDA et utilisé pour sélectionner uniquement les pseudo-étiquettes jugée confiantes sur les images du domaine cible, comme illustré dans Fig. 4.2. À cette fin, le cadre proposé dans Section 3.3 dans une configuration de classification d'images, et appliqué à la prédiction de classification d'images erronées, doit ici être adapté à la sortie structurée de la segmentation sémantique, qui peut être vue comme un problème de classification par pixels. Étant donné une image du domaine cible $\boldsymbol{x}_\mathrm{t}$, nous voulons prédire à la fois sa carte sémantique $F(\boldsymbol{x}_\mathrm{t};\boldsymbol{\theta})$ et, en utilisant un modèle auxiliaire avec des paramètres entraînables $\boldsymbol{\omega}$, sa carte de confiance :

$$C(\boldsymbol{x}_\mathrm{t};\boldsymbol{\omega}) = \mathsf{C}^{\boldsymbol{\omega}}_{\boldsymbol{x}_\mathrm{t}} \in [0,1]^{H\times W}. \tag{6.7}$$

Étant donné un pixel (h,w), si sa confiance $\mathsf{C}^{\boldsymbol{\omega}}_{\boldsymbol{x}_\mathrm{t}}[h,w]$ est supérieure à un seuil choisi δ, nous l'étiquetons avec sa classe prédite $f(\boldsymbol{x}_\mathrm{t})[h,w] = \mathrm{argmax}_{k\in\mathcal{Y}}\, \mathsf{P}^{\boldsymbol{\theta}}_{\boldsymbol{x}_\mathrm{t}}[h,w,k]$, sinon elle est masquée. Calculées sur toutes les images de $\mathcal{D}_\mathrm{t}$, ces cartes de segmentation incomplètes constituent des pseudo-étiquettes cibles qui sont utilisées pour entraîner un nouveau réseau de segmentation sémantique.

Entraînement. Pour entraîner le réseau de confiance C, nous proposons d'optimiser conjointement deux objectifs. Le premier objectif est une version pixel-à-pixel de la perte de confiance Eq. (6.6). Sur des images annotées du domaine source, il exige que le réseau de confiance C prédise à chaque pixel le score attribué par le classifieur F à la vraie classe (connue) :

$$\mathcal{L}_{\text{conf}}(\boldsymbol{\omega}; \mathcal{D}_{\text{s}}) = \frac{1}{N_{\text{s}}} \sum_{n=1}^{N_{\text{s}}} \left\| \mathsf{C}^{\boldsymbol{\omega}}_{\boldsymbol{x}_{\text{s},n}} - \text{TCP}_F(\boldsymbol{x}_{\text{s},n}, \boldsymbol{y}_{\text{s},n}) \right\|_{\text{F}}^2, \tag{6.8}$$

où $\| \cdot \|_{\text{F}}$ désigne la norme de Frobenius et, pour une image $\boldsymbol{x}$ avec une carte de segmentation vraie $\boldsymbol{y}$ et une carte de segmentation prédite $F(\boldsymbol{x}; \hat{\boldsymbol{\theta}})$, on note

$$\text{TCP}_F(\boldsymbol{x}, \boldsymbol{y})[h, w] = F(\boldsymbol{x}; \hat{\boldsymbol{\theta}}) \Big[h, w, \boldsymbol{y}[h, w] \Big] \tag{6.9}$$

à l'emplacement (h, w). Sur une nouvelle image d'entrée, C doit prédire à chaque pixel le score que F attribuera à la vraie classe inconnue, qui servira de mesure de confiance.

Cependant, par rapport à l'application du chapitre précédent, nous avons ici le problème supplémentaire du gap entre les domaines source et cible, un problème qui pourrait affecter l'entraînement du modèle de confiance comme dans l'entraînement du modèle de segmentation. Le deuxième objectif concerne donc le gap entre les domaines. Alors que le réseau de confiance C apprend à estimer le TCP sur les images du domaine source, son estimation de la confiance sur les images du domaine cible peut souffrir considérablement de ce gap de domaine. Comme cela se fait classiquement dans l'UDA, nous proposons un apprentissage adversarial de notre modèle auxiliaire afin de résoudre ce problème. Plus précisément, nous voulons que les cartes de confiance produites par C dans le domaine source ressemblent à celles obtenues dans le domaine cible.

Un discriminateur $D : [0, 1]^{H \times W} \to \{0, 1\}$, avec les paramètres $\boldsymbol{\psi}$, est entraîné simultanément avec C dans le but de reconnaître le domaine (1 pour la source, 0 pour la cible) d'une image étant donné sa carte de confiance. La fonction de perte suivante est minimisée par rapport à $\boldsymbol{\psi}$:

$$\mathcal{L}_D(\boldsymbol{\psi}; \mathcal{D}_{\text{s}} \cup \mathcal{D}_{\text{t}}) = \frac{1}{N_{\text{s}}} \sum_{n=1}^{N_{\text{s}}} \mathcal{L}_{\text{adv}}(\boldsymbol{x}_{\text{s},n}, 1) + \frac{1}{N_{\text{t}}} \sum_{n=1}^{N_{\text{t}}} \mathcal{L}_{\text{adv}}(\boldsymbol{x}_{\text{t},n}, 0), \tag{6.10}$$

où $\mathcal{L}_{\text{adv}}$ désigne la perte d'entropie croisée du discriminateur basé sur les cartes de confiance :

$$\mathcal{L}_{\text{adv}}(\boldsymbol{x}, \lambda) = -\lambda \log \left(D(\mathsf{C}^{\boldsymbol{\omega}}_{\boldsymbol{x}}; \boldsymbol{\psi}) \right) - (1 - \lambda) \log(1 - D(\mathsf{C}^{\boldsymbol{\omega}}_{\boldsymbol{x}}; \boldsymbol{\psi})), \tag{6.11}$$

pour $\lambda \in \{0, 1\}$, qui est fonction à la fois de $\boldsymbol{\psi}$ et de $\boldsymbol{\omega}$. En alternance avec l'apprentissage du discriminateur à l'aide de Eq. (6.10), l'apprentissage adversarial du réseau de confiance est effectué en minimisant, par rapport à $\boldsymbol{\omega}$, la fonction de perte suivante :

$$\mathcal{L}_C(\boldsymbol{\omega}; \mathcal{D}_{\text{s}} \cup \mathcal{D}_{\text{t}}) = \mathcal{L}_{\text{conf}}(\boldsymbol{\omega}; \mathcal{D}_{\text{s}}) + \frac{\lambda_{\text{adv}}}{N_{\text{t}}} \sum_{n=1}^{N_{\text{t}}} \mathcal{L}_{\text{adv}}(\boldsymbol{x}_{\text{t}}, 1), \tag{6.12}$$

où le deuxième terme, pondéré par $\lambda_{\text{adv}} > 0$, encourage C à produire des cartes dans le domaine cible qui confondront le discriminateur.

Le schéma d'apprentissage adversarial de confiance proposé agit également comme un régulateur pendant la formation, améliorant la robustesse de la confiance de la cible TCP inconnue. Comme l'apprentissage du modèle de confiance peut en fait être instable, l'apprentissage adversarial fournit un signal d'information supplémentaire, imposant en particulier que l'estimation de la confiance soit invariante aux changements de domaine. Nous observons empiriquement que cet apprentissage adversarial de la confiance fournit de meilleures estimations de la confiance et améliore la convergence et la stabilité du schéma d'apprentissage.

Architecture multi-échelle. Dans de nombreux jeux de données de segmentation, l'existence d'objets à des échelles différentes peut compliquer l'estimation de la confiance. Comme dans les travaux récents traitant d'échelles variables des objets [7], nous améliorons encore notre réseau de confiance C en ajoutant une architecture multi-échelle basée sur le regroupement spatial pyramidal. Cette architecture consiste en un schéma efficace en termes de calcul pour ré-échantillonner une carte de caractéristiques à différentes échelles, puis pour agréger les cartes de confiance. Nous illustrons l'architecture multi-échelle pour un réseau de confiance dans Fig. 4.4. À partir d'une carte de features, nous appliquons en parallèle des couches convolutives à trous avec des noyaux de taille 3x3 et des taux d'échantillonnage différents, chacune d'entre elles étant suivie d'une série de quatre couches convolutives standardes avec des noyaux de taille 3x3. Contrairement aux couches convolutionnelles avec de grands noyaux, les couches de convolution à trous élargissent le champ de vision des filtres et permettent d'incorporer un contexte plus large sans augmenter le nombre de paramètres et le temps de calcul. Les caractéristiques résultantes sont ensuite additionnées avant d'être sur-échantillonnées à la taille de l'image originale de $H \times W$. Nous appliquons une activation sigmoïde finale pour obtenir une carte de confiance avec des valeurs comprises entre 0 et 1.

Expériences. Nous considérons la tâche spécifique d'adaptation de données synthétiques à des données réelles dans des scènes urbaines. Nous expérimentons avec deux jeux de données sources synthétiques – SYNTHIA [165] et GTA5 [152]. – et deux ensembles de données cibles réelles : Cityscapes [77] et Mapillary Vistas [14]. Nous évaluons la méthode d'auto-apprentissage proposée sur trois architectures d'adaptation de domaine état de l'art (au moment du projet): *AdaptSegNet* [154], *AdvEnt* [155], *DADA* [159]. Elles sont toutes basées sur DeepLabv2 [7], un réseau de segmentation sémantique standard. ConDA apporte une amélioration systématique des performances par rapport à l'auto-formation basée sur la probabilité de classe maximale (MCP) standard. Fig. 4.7 présente les résultats qualitatifs de ces méthodes de pseudo-étiquetage. En particulier, ConDA obtient des résultats état de l'art (au moment du projet) sur trois benchmarks de segmentation UDA (GTA5 $\rightarrow$ Cityscapes, SYNTHIA $\triangleright$Cityscapes et SYNTHIA $\triangleright$Mapillary Vistas) en utilisant le réseau de segmentation standard DeepLabv2 [7] comme backbone.

Détection conjointe d'erreurs et d'anomalies avec les modèles évidentiels

Les modèles d'apprentissage automatique reposent généralement sur l'hypothèse que les données source et cible sont indépendantes et identiquement distribuées (*i.i.d.*). Pourtant, dans la pratique, les changements de distribution apparaissent naturellement dans de nombreux scénarios du monde réel. Par exemple, les voitures autonomes ont du mal à être performantes dans des conditions différentes de celles de l'entrainement, comme les variations de météo [167], de lumière [168] et de pose d'objets [169]. Pire encore, les modèles peuvent être exposés à des entrées provenant de classes non vues qu'ils tenteront de prédire malgré tout. Ces échecs peuvent passer inaperçus car ils n'entraînent pas d'erreurs explicites dans le modèle.

Alors que les travaux précédents de la littérature traitent séparément de la détection des erreurs de classification et de la détection des entrées hors distributions (OOD), nous soutenons qu'il est nécessaire pour un système de reconnaissance d'être capable d'identifier à la fois les erreurs de classification et les entrées inconnues/invisibles au moment du test pour un déploiement sûr dans des environnements ouverts [110]. Nous illustrons cette tâche dans Fig. 5.1. En particulier, nous constatons dans Section 5.5 que toutes les approches précédentes ne sont pas aussi performantes sur les deux tâches de détection, ce qui atténue leur capacité sur la tâche de détection conjointe.

Pour répondre à la tâche de détection simultanée des mauvaises classifications et des échantillons OOD, une bonne mesure d'incertitude devrait discriminer les prédictions correctes et les prédictions erronées pour les échantillons issus de la même distribution que celle d'entrainement tout en augmentant les valeurs d'incertitudes pour les entrées loin de la distribution. Par conséquent, elle devrait capturer à la fois l'incertitude aléatoire et épistémique. Les approches bayésiennes [40, 52] et les ensembles [53, 54] sont des méthodes qui induisent une estimation plus précise de l'incertitude épistémique. Ces techniques produisent une densité de probabilité sur la distribution catégorielle prédictive $p(y|\boldsymbol{x}, \mathcal{D})$ obtenue par échantillonnage comme le montre la ligne supérieure de Fig. 2.7. Mais cela se fait au prix d'un coût de calcul accru.

Une classe récente de modèles, appelée réseaux de neurones évidentiel (ENN) [56, 135, 170], propose plutôt d'apprendre explicitement les paramètres de concentration d'une distribution de Dirichlet $q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x}) = \text{Dir.}(\boldsymbol{\pi}|\boldsymbol{\alpha})$ sur les probabilités de sortie. Il a été démontré qu'elles améliorent la généralisation [170] et la détection des OOD [171]. L'apprentissage des ENN est formulé comme une approximation variationnelle visant à minimiser la divergence de Kullback-Leibler (KL) entre la distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\pi}|\boldsymbol{x})$ et la vraie distribution postérieure $p(\boldsymbol{\pi}|\boldsymbol{x}, y)$. En suivant [170], nous utilisons un prior uniforme $p(\boldsymbol{\pi}|\boldsymbol{x}) = \text{Dir}(\boldsymbol{\pi}|\mathbf{1})$. La fonction de perte d'entraînement d'un ENN est :

$$\mathcal{L}_{\text{var}}(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{N} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \Big(\psi(\alpha_y) - \psi(\alpha_0) \ + \lambda \text{KL}(\text{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \ \| \text{Dir}(\boldsymbol{\pi}|\mathbf{1})) \Big), \tag{6.13}$$

avec l'hyperparamètre $\lambda > 0$. En particulier, la minimisation de cette fonction de perte impose que la précision de l'échantillon d'apprentissage α_0 reste proche de $C + 1/\lambda$.

Basés sur le cadre de la logique subjective [96], les modèles évidentiels capturent différentes sources

d'incertitude. L'incertitude de premier ordre concerne l'espérance de la distribution de Dirichlet et est causée par des preuves contradictoires, par exemple la confusion entre classes. L'incertitude de second ordre exprime le manque d'évidence dans une prédiction [101], qui est caractérisée par la dispersion de la distribution de Dirichlet. Par exemple, les huskies partagent de nombreuses caractéristiques avec les loups bien qu'ils soient une race de chien, ce qui entraîne une grande incertitude du premier ordre due à la confusion de classe. En présence d'un dessin d'un husky, on s'attend à une confusion de classe similaire, mais à une quantité moindre de preuves en raison du changement de distribution.

De manière surprenante, les précédents travaux de la littérature n'exploitent pas la distribution sur les probabilités sur le simplex pour dériver une telle mesure jointe des deux sources d'incertitude. Certaines méthodes se concentrent sur la détection des OOD en caractérisant uniquement la dispersion de la distribution, *e.g.*, en utilisant l'information mutuelle [135]. Les approches ciblant l'incertitude totale réduisent en fait les distributions de probabilité sur le simplexe à leur valeur en espérance et calculent des mesures d'incertitude du premier ordre, *e.g.*, l'entropie prédictive [56]. Cependant, ces mesures sont invariantes à la dispersion de la distribution, alors que l'incertitude causée par la confusion de classe et le manque de preuves devrait être cumulative, une propriété naturellement remplie par la variance prédictive dans la régression bayésienne [98]. En outre, certaines méthodes pour les modèles évidentiels utilisent des données auxiliaires pendant la formation afin d'imposer un étalement de distribution plus élevé sur les entrées OOD. Mais lorsque l'accès aux données d'entraînement OOD n'est pas envisageable, le comportement de grande dispersion n'est pas garanti pour tous les exemples OOD [116, 117] et les mesures d'incertitude d'ordre deux peinent à les discriminer des exemples issus de la distributio d'entrainement.

KLoS. Nous introduisons une nouvelle mesure, appelée *KLoS*, qui calcule la divergence KL entre la sortie du modèle et une distribution de Dirichlet "peaké" avec des concentrations $\boldsymbol{\gamma}_{\hat{y}}$ concentrées sur la classe *prédite* $\hat{y}$:

$$\mathrm{KLoS}(\boldsymbol{x}) \triangleq \mathrm{KL}\Big(\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \parallel \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\gamma}_{\hat{y}})\Big), \tag{6.14}$$

où $\boldsymbol{\alpha} = \exp f(\boldsymbol{x}, \boldsymbol{\theta})$ sont la sortie du modèle et $\boldsymbol{\gamma}_{\hat{y}} = (1, \ldots, 1, \tau, 1, \ldots, 1)$ sont les paramètres de concentration uniforme sauf pour la classe prédite avec $\tau = 1/\lambda + 1$.

Plus KLoS est petit, plus la prédiction est certaine. Les prédictions correctes auront des distributions de Dirichlet similaires à la distribution de Dirichlet du prototype $\boldsymbol{\gamma}_{\hat{y}}$ et seront donc associées à un score d'incertitude faible (Fig. 5.4a). Les échantillons présentant une confusion de classe élevée présenteront une distribution de probabilité en espérance plus proche du centre du simplex que le prototype de classe en espérance $p^*_{\hat{y}} = (\frac{1}{K-1+\tau}, \cdots, \frac{\tau}{K-1+\tau}, \ldots, \frac{1}{K-1+\tau})$, ce qui entraîne un score KLoS plus élevé (Fig. 5.4b). De même, KLoS pénalise également les échantillons dont la précision α_0 est différente de la précision $\alpha^*_0 = \tau + K - 1$ du prototype $\boldsymbol{\gamma}_{\hat{y}}$. Les échantillons dont la quantité d'évidence est inférieure (Fig. 5.4c) et supérieure (Fig. 5.4d) à α^*_0 reçoivent un score KLoS plus élevé.

Puisque les échantillons de la distribution doivent avoir une précision proche de α^*_0 pendant l'entrainement, les prototypes par classe sont des estimations fines des paramètres de concentration des données d'entrainement pour chaque classe. Par conséquent, KLoS est une métrique basée sur

la divergence, qui n'a besoin que des données de la distribution pendant l'entrainement pour calculer ses prototypes. Ce comportement est illustré dans Section 5.5.1. La mesure proposée sera efficace pour détecter différents types d'échantillons OOD dont la précision est loin de α_0^*. En revanche, les mesures d'incertitude de second ordre, telles que l'information mutuelle, supposent que les échantillons OOD ont des α_0 plus petits, une propriété difficile à respecter pour les modèles entrainés uniquement avec des échantillons issus de la distribution d'entrainement (voir Fig. 5.3). Dans Section 5.5.5, nous explorons plus en profondeur l'impact du choix des données d'entraînement OOD sur les valeurs réelles de α_0 pour les échantillons OOD.

KLoSNet. Lorsque le modèle classe mal un exemple, c'est-à-dire que la classe prédite $\hat{y}$ diffère de la vérité terrain y, KLoS mesure la distance entre la sortie du ENN et le postérieur $p(\boldsymbol{\pi}|\boldsymbol{x}, \hat{y})$ estimé sur la mauvaise classe. Mesurer plutôt la distance à la distribution postérieure à la vraie classe $p(\pi|\boldsymbol{x}, y)$ donnerait plus probablement une plus grande valeur, reflétant le fait que le classifieur a fait une erreur. Ainsi, une meilleure mesure pour la détection des erreurs de classification serait :

$$\mathrm{KLoS}^*(\boldsymbol{x}, y) \triangleq \mathrm{KL}\Big(\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \parallel \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\gamma}_y)\Big), \tag{6.15}$$

où $\boldsymbol{\gamma}_y$ correspond aux concentrations uniformes sauf pour la *vrai* classe y avec $\tau = 1/\lambda + 1$.

Évidemment, la vraie classe d'une prédiction n'est pas disponible lors de l'estimation de la confiance sur des échantillons de test. Nous proposons d'apprendre KLoS* en introduisant un réseau de neurone auxiliaire de confiance, appelé KLoSNet, avec des paramètres $\boldsymbol{\omega}$, qui produit une prédiction de confiance $C(\boldsymbol{x}, \boldsymbol{\omega})$. KLoSNet consiste en un petit décodeur, composé de plusieurs couches denses attachées à l'avant-dernière couche du réseau de classification original. Pendant l'apprentissage, nous cherchons $\boldsymbol{\omega}$ tel que $C(\boldsymbol{x}, \boldsymbol{\omega})$ soit proche de KLoS$^*(\boldsymbol{x}, y)$, en minimisant

$$\mathcal{L}_{\mathrm{KLoSNet}}(\boldsymbol{\omega}; \mathcal{D}) = \frac{1}{N} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \big\| C(\boldsymbol{x}, \boldsymbol{\omega}) - \mathrm{KLoS}^*(\boldsymbol{x}, y) \big\|^2. \tag{6.16}$$

Expériences. Nous avons évalué notre approche sur la tâche de détection simultanée des mauvaises classifications et des échantillons OOD par rapport à diverses méthodes de référence, y compris les métriques d'incertitude de premier et de second ordre, les méthodes de post-entrainement pour la détection OOD [112, 113]) et notre précédent travail ConfidNet. Les prédictions correctes sont considérées comme des échantillons positifs tandis que les entrées mal classées et les exemples OOD constituent des échantillons négatifs. Des expériences sont menées avec les architectures VGG-16 [74] et ResNet-18 [76] sur les jeux de données CIFAR-10 et CIFAR-100 [35]. Les résultats montrent que KLoSNet agit comme un estimateur de densité par classe et surpasse les mesures d'incertitude actuelles.

La littérature sur les modèles évidentiels ne traite que d'un ensemble d'entraînement OOD en lien avec le jeux de données d'entrainement, *e.g.*, CIFAR-100 pour les modèles entraînés sur CIFAR-10. Dans Fig. 5.14, nous faisons varier l'ensemble d'entraînement OOD utilisé pour entraîner un modèle évidentiel avec la fonction de perte de divergence KL inverse [135] et évaluons les performances en

utilisant TinyImageNet comme ensemble de test OOD. Comme prévu, l'utilisation de CIFAR-100 comme données d'entraînement OOD améliore les performances pour chaque mesure (MCP, Mut. Inf. et KLoS). Cependant, l'amélioration apportée par l'entraînement avec des échantillons OOD dépend fortement de l'ensemble de données choisi. La performance de Mut. Inf. diminue de 92,6% AUC avec CIFAR-100 à 82,9% en passant à LSUN, et devient même pire avec SVHN (78,5 %) par rapport à l'utilisation de données OOD (80,6%). Nous constatons également que KLoS surpasse ou est à égalité avec MCP et Mut. Inf. dans tous les cas. Plus important encore, l'utilisation de KLoS sur des modèles sans données d'entraînement OOD donne de meilleures performances de détection que d'autres mesures prises à partir de modèles entraînés avec des échantillons OOD inappropriés, c'est-à-dire tous les jeux de données OOD autres que CIFAR-100.

Conclusion et perspectives

La principale contribution de cette thèse est d'utiliser un modèle de confiance auxiliaire pour apprendre la confiance d'une prédiction d'un réseaux de neurone profond en classification. Étant donné un modèle de classification entraîné, le modèle de confiance apprend à estimer à partir des données un critère adéquat dérivé du classifieur, tel que la probabilité de la vrai classe pour les réseaux de neurones standard et KLoS pour les modèles évidentiels. Au moment du test, nous utilisons directement la sortie du modèle de confiance comme estimation de l'incertitude. L'un des principaux avantages de cette méthode est d'être agnostique en termes d'architecture : dans nos expériences, nous avons réussi à améliorer l'estimation de l'incertitude pour les modèles de classification avec différentes architectures d'apprentissage profond (MLP, LeNet, VGGs, ResNets). Nous avons appliqué notre approche à trois tâches : prédiction d'échec (Chapter 3), adaptation non supervisée de domaine pour la segmentation sémantique (Chapter 4), et détection simultanée des erreurs de distribution et des échantillons hors distribution (Chapter 5). Pour chaque tâche, il y a deux défis principaux à relever : (1) quel critère utiliser, et (2) comment entraîner efficacement le modèle de confiance.

Discutons maintenant des directions intéressantes qui pourraient être abordées dans des travaux futurs en relation avec nos contributions.

Génération de données d'erreurs pour faciliter l'apprentissage de confiance . L'apprentissage de confiance a montré des améliorations significatives par rapport aux méthodes de références pour l'estimation de l'incertitude dans chacun des travaux considérés. Néanmoins, l'apprentissage du modèle auxiliaire dépend de la qualité du jeu de données, c'est-à-dire du nombre d'erreurs disponibles. Les réseaux de neurones modernes sont sur-paramétrés et ont tendance à sur-apprendre les données d'entraînement, atteignant ainsi une grande précision sur les jeux d'entraînement et ne laissant qu'une petite fraction d'échantillons mal classés. Nous pensons que ce problème de déséquilibre des données atténue les performances de l'apprentissage de confiance. Pour résoudre le problème du déséquilibre, une solution serait de générer artificiellement des erreurs. Les perturbations adversariales sont de petites perturbations de l'entrée qui sont presque imperceptibles pour les humains, mais qui trompent un réseau de neurone, transformant ainsi une prédiction correcte en une prédiction erronée. Un

ensemble combiné d'entrées authentiques et adverses permettrait de rééquilibrer la formation. Il a été démontré que Mix-up [177], et plus généralement les techniques agressives d'augmentation des données telles qu'AugMix [178] et CutMix [179] améliorent la robustesse et pourraient également être appliquées ici pour générer des échantillons avec des probabilités mixtes, fournissant ainsi une plus grande gamme de valeurs TCP pour l'apprentissage de la confiance.

Apprentissage de la confiance d'un ensemble En tant qu'alternative simple aux méthodes entièrement bayésiennes, les ensembles a été un sujet de recherche populaire au sein des méthodes probabilistes [180, 181, 53]. Pourtant, lorsqu'il s'agit de prédire les défaillances, les méthodes précédentes reposent sur la réduction des prédictions en un seul vecteur de probabilité moyen et sur la dérivation des mesures habituelles telles que le MCP et l'entropie. Une direction intéressante serait de combiner l'idée de l'apprentissage de confiance au contexte des ensembles. La solution la plus simple serait d'entraîner un modèle auxiliaire pour régresser la valeur TCP du vecteur de probabilité moyen. Les expériences préliminaires ont montré des difficultés à converger vers la régression de telles valeurs moyennes, car chaque modèle individuel se comporte différemment et nous ne pouvons pas nous fier aux poids initialisés d'un modèle arbitraire. Nous pourrions essayer d'attacher un modèle auxiliaire à chaque membre de l'ensemble et de les entraîner séparément avant de faire la moyenne des sorties de tous les modèles de confiance. Une approche intelligente impliquerait de tirer parti de la diversité des prédictions pour affiner le critère à estimer pendant l'apprentissage de la confiance.

Modèles génératifs pour la détection d'échantillons hors distribution Dans le Chapter 5, nous avons mis en évidence le comportement d'estimateur de densité par classe de KLoS, qui est une propriété cruciale en l'absence de données d'entraînement OOD pour améliorer la détection simultanée des erreurs de classification et des échantillons OOD. Parallèlement à cette contribution, nos expériences ont également révélé que si les performances des mesures d'incertitude existantes sont considérablement améliorées par l'utilisation d'échantillons OOD, elles dépendent aussi de manière critique du type de ces échantillons (Section 5.5.5). Parmi les méthodes utilisant des échantillons OOD lors de l'apprentissage de classifieur profonds, Hendrycks *et al.* [115] proposent d'apprendre à classer les échantillons de la distribution d'entrainement tout en produisant une entropie prédictive élevée pour les échantillons OOD. En conséquence, ils utilisent l'entropie prédictive pour distinguer les échantillons de la distribution d'entraiement des échantillons hors distribution. Cette méthode repose sur la disponibilité d'un grand ensemble de données OOD, par exemple 80 millions TinyImages avec CIFAR-10 ou CIFAR-100 comme jeu de données de distribution. Mais si trouver des échantillons OOD appropriés peut être facile pour certains jeux de données académiques, cela peut s'avérer plus problématique dans les applications du monde réel [116, 117], avec le risque de dégrader les performances avec un choix inapproprié. La construction d'un ensemble d'entraînement OOD adapté aux tâches réelles est une perspective de recherche qui pourrait alléger le besoin d'échantillons OOD réels mais difficiles à trouver. Pour garantir une bonne généralisation à d'autres types d'anomalies, le principal défi consiste à produire des échantillons proches de la distribution, même à la limite, comme pour l'ensemble de données jouet présenté dans Fig. 6.1. Les modèles génératifs tels que proposés dans [182, 117, 183] pourraient constitués une solution intéressante pour répondre à ce problème.

Applications supplémentaires de l'apprentissage de confiance De manière analogue au Chapter 4, l'approche de l'apprentissage de confiance pourrait être appliquée à de nouveaux contextes où la qualité des estimations d'incertitude est cruciale. L'apprentissage semi-supervisé (SSL) [184] est un paradigme d'apprentissage qui vise à améliorer les performances d'apprentissage à partir de données étiquetées et des instances non étiquetées supplémentaires. Une famille d'approches pour SSL [37, 163] propose de prédire des pseudo-étiquettes sur des données non étiquetées et de réentraîner un réseau en utilisant ces pseudo-étiquettes. La clé est de sélectionner les étiquettes les plus fiables. De toute évidence, et en raison du lien étroit entre l'adaptation de domaine et les méthodes d'apprentissage semi-supervisé, une extension naturelle de ConDA consiste à l'appliquer dans ce dernier contexte. Une alternative à l'apprentissage semi-supervisé est l'apprentissage actif dans lequel les données sont échantillonnées pour être étiquetées par des oracles humains dans le but de maximiser la performance du modèle tout en minimisant les coûts d'étiquetage. Diverses stratégies d'échantillonnage ont été proposées pour l'apprentissage actif au fil des ans, selon des perspectives différentes, par exemple l'incertitude [39] et la représentativité [185]. L'apprentissage par la confiance pourrait améliorer la sélection d'échantillons utiles, comme cela est fait dans un travail connexe où les auteurs visent à apprendre la valeur de perte [186].

Appendix A

Additional Analysis for Failure Prediction Experiments

A.1 Effect on confidence loss

The influence of the loss (MSE, BCE, Focal Loss or Ranking based on TCP) is analysed for SVHN, CIFAR10 and CamVid in Table A.1. We also tested the normalized variant of the TCP confidence criterion, nTCP. We can observe that its performance is lower than the one of TCP on small datasets such as CIFAR-10 where few errors are present, but higher on larger datasets such as CamVid where each pixel is a sample. This emphasizes once again the complexity of incorrect/correct classification training.

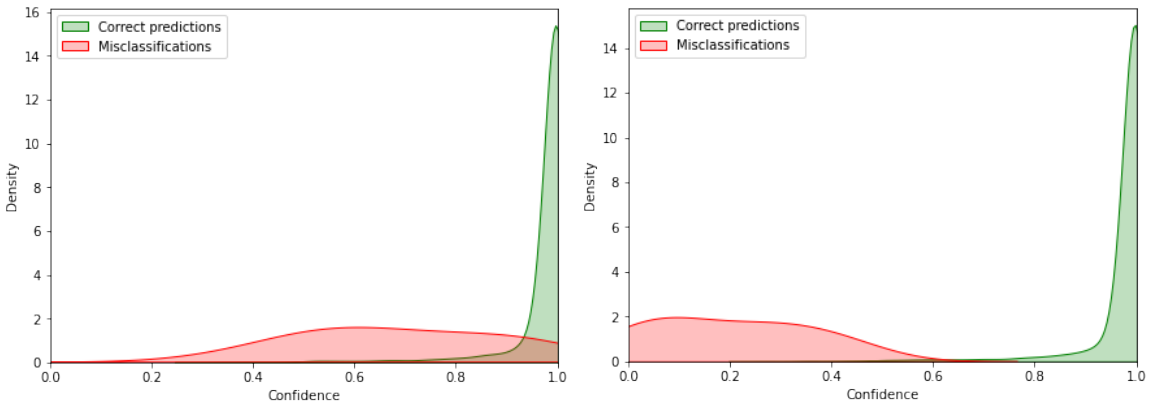
A.2 Empirical error and success distributions

In this section, we provide the plots, analogous to Figure 1 in the main paper, that show the distribution of the confidence measures over correct and incorrect predictions respectively, for each dataset and each model in our failure prediction experiments. We also include absolute numbers of incorrect and correct predictions grouped into 3 bins ('$> 1/K$', '$[\frac{1}{K}, \frac{1}{2}]$' and '$> 1/2$') to validate our assumptions about TCP's properties. The plots are available for MNIST with MLP in Fig. A.1, for MNIST with a small convnet in Fig. A.2, for SVHN with a small convnet in Fig. A.3, for CIFAR-100 with VGG-16 in Fig. A.4 and for CamVid with SegNet in Fig. A.5.

Table A.1: **Effect of the loss and of the confidence criterion on the error-detection performance of ConfidNet**. Comparison in between proposed MSE and three other alternatives, all based on TCP as confidence criterion, except last one which is MSE with normalized TCP (nTCP). This table extends Table 3.2.

Dataset	Loss	FPR@95%TPR↓	AUPR↑	AUROC ↑
SVHN SmallConvNet	BCE	29.34%	50.00%	92.76%
	Focal	28.67%	49.96%	93.01%
	Ranking	31.04%	48.11%	92.90%
	nTCP	30.19%	47.04%	93.12%
	TCP	**28.58%**	**50.72%**	**93.44%**
CIFAR-10 VGG-16	BCE	45.20%	47.95%	91.94%
	Focal	45.20%	47.76%	91.93%
	Ranking	46.99%	44.04%	91.49%
	nTCP	45.02%	48.78%	92.06%
	TCP	**44.94%**	**49.94%**	92.12%
CamVid SegNet	BCE	61.68%	48.96%	83.41%
	Focal	61.64%	49.05%	84.09%
	nTCP	**60.41%**	**51.35%**	**85.18%**
	TCP	61.52%	50.51%	85.02%

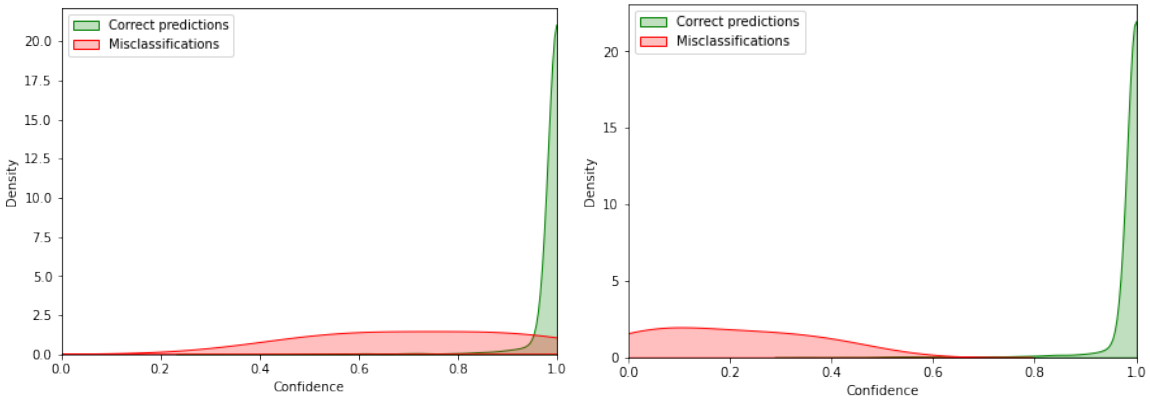
Figure A.1: Distributions of MCP and TCP confidence estimates computed over correct and erroneous predictions by a trained **MLP on MNIST.**



(a) MCP (b) TCP

Model	Nb. of Errors			Nb. of Successes			AUPR↑	AUROC ↑
	$> 1/K$	$[\frac{1}{K}, \frac{1}{2}]$	$> 1/2$	$< 1/K$	$[\frac{1}{K}, \frac{1}{2}]$	$> 1/2$		
MCP	0	25	170	0	28	9777	37.70%	97.13%
TCP	81	114	0	0	28	9777	98.77%	99.98%

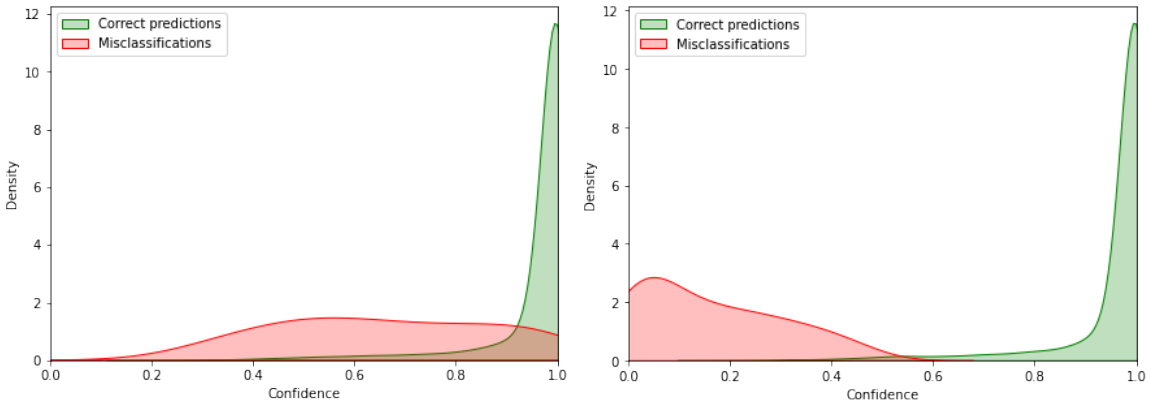
Figure A.2: Distributions of MCP and TCP confidence estimates computed over correct and erroneous predictions by a trained **small ConvNet model on MNIST.**



(a) MCP (b) TCP

Model	Nb. of Errors			Nb. of Successes			AUPR ↑	AUROC ↑
	$> 1/K$	$[\frac{1}{K}, \frac{1}{2}]$	$> 1/2$	$< 1/K$	$[\frac{1}{K}, \frac{1}{2}]$	$> 1/2$		
MCP	0	8	82	0	11	9899	35.05%	98.63%
TCP	32	58	0	0	11	9899	99.41%	99.41%

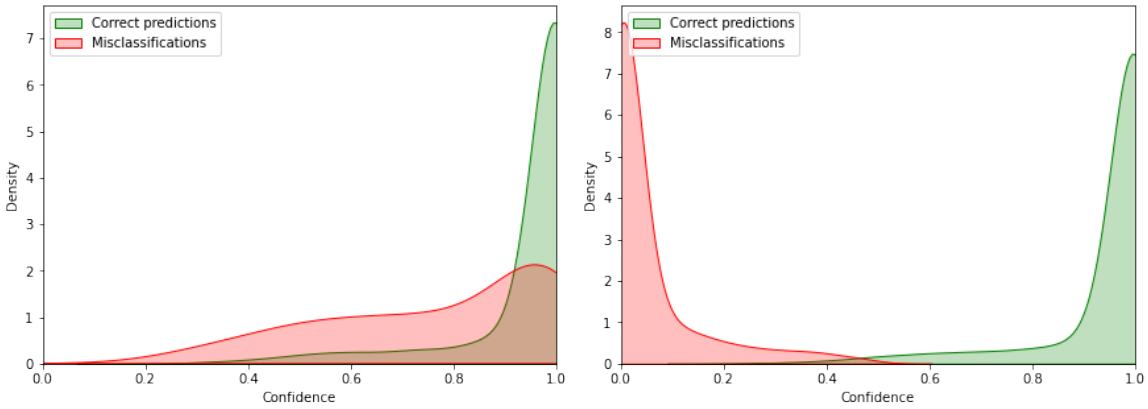
Figure A.3: Distributions of MCP and TCP confidence estimates computed over correct and erroneous predictions by a trained **small ConvNet architecture on SVHN.**



(a) MCP (b) TCP

Model	Nb. of Errors			Nb. of Successes			AUPR↑	AUROC ↑
	$> 1/K$	$[\frac{1}{K}, \frac{1}{2}]$	$> 1/2$	$< 1/K$	$[\frac{1}{K}, \frac{1}{2}]$	$> 1/2$		
MCP	0	329	857	0	206	24640	48.18%	93.20%
TCP	500	686	0	0	206	24640	98.93%	99.95%

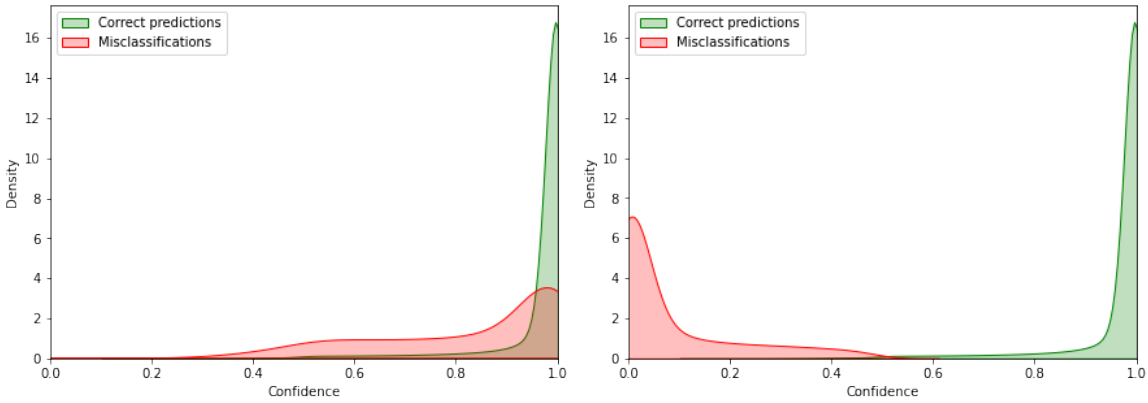
Figure A.4: Distributions of MCP and TCP confidence estimates computed over correct and erroneous predictions by a trained **VGG-16 model on CIFAR-100.**



(a) MCP (b) TCP

Model	Nb. of Errors			Nb. of Successes			AUPR ↑	AUROC ↑
	$> 1/K$	$[\frac{1}{K}, \frac{1}{2}]$	$> 1/2$	$< 1/K$	$[\frac{1}{K}, \frac{1}{2}]$	$> 1/2$		
MCP	0	603	2801	0	118	6478	71.99%	85.67%
TCP	2724	680	0	0	118	6478	99.91%	99.91%

Figure A.5: Distributions of MCP and TCP confidence estimates computed over correct and erroneous predictions by a trained **SegNet model on CamVid.**



(a) MCP (b) TCP

Model	Nb. of Errors			Nb. of Successes			AUPR↑	AUROC↑
	$> 1/K$	$[\frac{1}{K}, \frac{1}{2}]$	$> 1/2$	$< 1/K$	$[\frac{1}{K}, \frac{1}{2}]$	$> 1/2$		
MCP	0	401,573	55,506,172	0	188,128	34,166,526	48.53%	84.42%
TCP	41,84,875	1,722,871	0	0	188,128	34,166,526	99.92%	99.99%

Appendix B

Details and Further Experiments for KLoS

B.1 Experimental Setup

In this section, we provide comprehensive details about the datasets, the implementation and the hyperparameters of the experiments shown in Chapter 5.

B.1.1 Image Classification Datasets.

In Section 5.5 to Section 5.5.5, the experiments are conducted using CIFAR-10 and CIFAR-100 datasets [35]. They consist in 32×32 natural images featuring 10 object classes for CIFAR-10 and 100 classes for CIFAR-100. Both datasets are composed with 50,000 training samples and 10,000 test samples. We further randomly split the training set to create a validation set of 10,000 images.

OOD datasets are TinyImageNet[1] – a subset of ImageNet (10,000 test images with 200 classes) –, LSUN [174] – a scene classification dataset (10,000 test images of 10 scenes) –, STL-10 – a dataset similar to CIFAR-10 but with different classes, and SVHN [36] – an RGB dataset of 28×28 house-number images (73,257 training and 26,032 test images with 10 digits) –. We downsample each image of TinyImageNet, LSUN and STL-10 to size 32×32.

Training Details. We implemented in PyTorch [193] a VGG-16 architecture [74] in line with the previous works of [116, 135, 171], with fully-connected layers reduced to 512 units. Models are trained for 200 epochs with a batch size of 128 images, using a stochastic gradient descent with Nesterov momentum of 0.9 and weight decay 5e-4. The learning rate is initialized at 0.1 and reduced by a factor of 10 at 50% and 75% of the training progress. Images are randomly horizontally flipped and shifted by ± 4 pixels as a form of data augmentation.

Balancing Misclassification and OOD Detection. Most neural networks used in our experiments tend to overfit, which leaves very few training errors available.

[1] https://tiny-imagenet.herokuapp.com/

We provide accuracies on training, validation and test sets in Table B.1. With such high predictive performances, the number of misclassifications is usually lower than the number of OOD samples (∼10,000). Hence, the oversampling approach proposed in the paper helps to better balance misclassification detection performances and OOD detection performances in the reported metrics.

	CIFAR-10	CIFAR-100
Train	99.0 ±0.1	91.2 ±0.2
Val	93.6 ±0.1	70.6 ±0.3
Test	93.0 ±0.3	70.1 ±0.4

Table B.1: Mean accuracies (%) and std. over five runs.

KLoSNet. We start from the pre-trained evidential model described above. As detailed in Section 3.2 of the main paper, KLoSNet consists of a small decoder attached to the penultimate layer of the main network. In CIFAR experiments, this corresponds to VGG-16's fc1 layer of size 512. This auxiliary neural network is composed of five fully-connected layers of size 400, except for the last layer obviously. KLoSNet decoder's weights ω are trained for 100 epochs with ℓ_2 loss (Eq. 8 in the main paper) and with Adam optimizer with learning rate 1e-4. As KLoS* ranges from zero to large positive values (>1000), one may encounter some issues when training KLoSNet. Consequently, we apply a sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$, after computing the KL-divergence between the NN's output and γ_y. To prevent over-fitting, training is stopped when validation AUC metric for misclassification detection starts decreasing. Then, a second training step is performed by initializing new encoder E' such that $\theta_{E'} = \theta_E$ initially and by optimizing weights $(\theta_{E'}, \omega)$ for 30 epochs with Adam optimizer with learning rate 1e-6. We stop training once again based on the validation AUC metric.

B.2 Additional Results

B.2.1 Detailed Results for Synthetic Experiments

We detail in Table B.2 the quantitative results for the task of simultaneous detection of misclassifications and of OOD samples for the synthetic experiment presented in Section 4.1 of the paper. First-order uncertainty measures such as MCP and Entropy perform obviously well on the first task with 80.2% AUC for MCP. However, their OOD performance drops to ∼15% AUC on this dataset. On the other hand, Mahalanobis is adapted to detect OOD samples but not as good for misclassifications. KLoS achieves comparable performances to best methods in misclassification detection and in OOD detection (79.4% for Mis. and 98.8% for OOD). As a result, when detecting both inputs simultaneously, KLoS improves all baselines, reaching 89.2% AUC.

B.2.2 Results with SVHN as OOD test dataset

We report in Fig. B.1 all the results when evaluating with SVHN [36] as OOD dataset. Along with simultaneous detection results, we also provide separate results for misclassifications detection and OOD detection respectively. Similarly to the comparative results in the main paper, KLoSNet outperforms all the baselines in every simultaneous detection benchmark, with Mahalanobis being second.

Method	Mis. (↑)	OOD (↑)	Mis+OOD (↑)
MCP	**80.2** ±1.1	15.9 ±0.7	48.6 ±1.9
Entropy	**78.4** ±1.5	11.0 ±0.3	45.7 ±1.0
Mut. Inf.	75.0 ±2.3	2.2 ±0.2	38.8 ±1.2
Diff. Ent.	74.2 ±2.7	1.9 ±1.0	38.0 ±1.3
Mahalanobis	51.5 ±2.8	**98.5** ±0.3	75.0 ±1.4
KLoS	**79.4** ±1.2	**98.8** ±0.3	**89.2** ±0.5

Table B.2: Synthetic experiment: misclassification (Mis.), out-of-distribution detection (OOD) and simultaneous detection (Mis+OOD) (mean % AUC and std. over 5 runs). Bold type indicates significant top performance ($p < 0.05$) according to paired t-test.

B.2.3 Detail results of selective classification

In addition to aggregated results shown in Section 5.5.3, we provide detailed results of selective classification in presence of domain shifts by corruption for CIFAR-10-C (Section B.2.3) and CIFAR-100-C (Section B.2.3). For almost every corruption, KLoSNet outperforms other methods. When averaged on all corruptions, KLoSNet scores 48.6% AURC while the second best, ConfidNet, reaches 49.0% AURC.

		Noise			**Blur**				**Weather**				**Digital**				
Method	Clean	Gaussian	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG	Mean
MCP	48.3%	46.7%	48.0%	43.7%	48.6%	45.7%	45.4%	43.4%	44.4%	42.5%	40.4%	46.5%	43.3%	43.3%	43.0%	1.9%	42.2%
Entropy	48.8%	47.2%	48.5%	43.9%	49.1%	45.9%	45.6%	43.6%	44.7%	42.7%	40.6%	46.9%	43.5%	43.7%	43.2%	2.0%	42.5%
ConfidNet	47.4%	45.8%	47.1%	42.9%	48.1%	45.2%	44.9%	42.5%	43.5%	41.6%	39.1%	45.9%	42.6%	42.0%	42.1%	1.3%	41.4%
Mut. Inf.	53.6%	52.3%	53.2%	48.3%	53.0%	49.3%	49.1%	48.8%	49.8%	47.8%	46.9%	51.5%	47.9%	49.8%	48.2%	4.8%	47.1%
ODIN	49.3%	47.7%	48.9%	44.3%	49.5%	46.3%	46.0%	44.1%	45.2%	43.2%	41.2%	47.3%	43.9%	44.3%	43.7%	2.2%	42.9%
Mahalanobis	48.9%	46.9%	48.6%	42.5%	49.7%	45.3%	44.8%	43.1%	44.1%	41.4%	38.6%	45.8%	42.6%	42.9%	42.4%	1.0%	41.8%
KLoSNet	**47.0%**	**45.3%**	**46.9%**	**42.3%**	**48.0%**	**45.0%**	**44.6%**	**42.2%**	**43.1%**	**41.1%**	**38.1%**	**45.2%**	**42.4%**	**41.8%**	**42.0%**	**0.9%**	**41.0%**

Table B.3: **Detailed results for selective classification on CIFAR-10-C**. Comparative performance in AURC (%) of classification with the option to reject misclassified test samples and samples from shifted distributions. Results are average on 5 runs (mean ± std.).

		Noise			**Blur**				**Weather**				**Digital**				
Method	Clean	Gaussian	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG	Mean
MCP	**53.3%**	**52.7%**	55.2%	50.8%	54.0%	52.8%	52.5%	51.3%	52.0%	50.5%	47.9%	52.1%	50.7%	50.4%	51.3%	13.1%	49.4%
Entropy	53.5%	53.0%	55.8%	51.3%	54.8%	53.3%	53.1%	51.8%	52.6%	51.0%	48.2%	52.7%	51.1%	50.9%	51.7%	13.5%	49.9%
ConfidNet	53.5%	52.8%	**55.0%**	50.1%	53.7%	52.3%	51.9%	50.8%	51.6%	50.0%	47.2%	51.7%	50.3%	49.8%	50.9%	11.7%	49.0%
Mut. Inf.	54.5%	54.0%	57.1%	53.1%	56.2%	54.9%	54.7%	53.5%	54.3%	52.6%	50.0%	54.4%	52.4%	53.0%	53.2%	16.5%	51.5%
ODIN	53.4%	52.9%	55.5%	51.2%	54.5%	53.2%	52.9%	51.7%	52.4%	50.9%	48.2%	52.5%	51.0%	50.8%	51.7%	13.7%	49.8%
Mahalanobis	54.5%	53.9%	56.7%	50.9%	55.5%	52.8%	52.9%	52.2%	52.5%	50.7%	47.8%	52.4%	51.0%	51.4%	51.9%	11.0%	49.9%
KLoSNet	54.0%	53.1%	55.5%	**49.5%**	**53.6%**	**51.6%**	**51.4%**	**50.7%**	**51.1%**	**49.2%**	**46.5%**	**51.3%**	**49.9%**	**49.7%**	**50.7%**	**9.7%**	**48.6%**

Table B.4: **Detailed results for selective classification on CIFAR-100-C**. Comparative performance in AURC (%) of classification with the option to reject misclassified test samples and samples from shifted distributions. Results are average on 5 runs (mean ± std.).

(a) CIFAR-10 with VGG-16

Method	CIFAR-10 Mis. (↑)	SVHN OOD (↑)	SVHN Mis+OOD (↑)
MCP	87.6 ±1.6	87.3 ±2.2	88.9 ±0.5
Entropy	83.5 ±2.4	85.5 ±2.3	86.9 ±1.9
ConfidNet	90.2 ±0.8	**89.0** ±3.1	91.0 ±1.1
Mut. Inf.	84.1 ±1.5	80.0 ±3.9	83.2 ±1.7
Diff. Ent.	86.8 ±1.0	86.0 ±2.0	87.6 ±0.9
EPKL	83.9 ±1.5	79.4 ±4.2	82.8 ±1.9
ODIN	86.0 ±2.0	86.8 ±2.2	87.7 ±1.0
Mahalanobis	91.2 ±0.3	**89.1** ±2.8	91.5 ±1.1
KLoSNet (Ours)	**92.5** ±0.6	**89.8** ±3.0	**92.7** ±1.2

(b) CIFAR-10 with ResNet18

Method	CIFAR-100 Mis. (↑)	SVHN OOD (↑)	SVHN Mis+OOD (↑)
MCP	84.9 ±0.8	79.6 ±1.0	83.0 ±0.9
Entropy	84.6 ±0.8	79.6 ±1.1	82.8 ±0.9
ConfidNet	90.7 ±0.4	84.6 ±1.1	88.6 ±0.6
Mut. Inf	80.6 ±0.6	77.0 ±1.2	79.4 ±0.9
Diff. Ent	82.7 ±0.6	78.3 ±1.2	81.1 ±0.9
EPKL	80.2 ±0.6	76.8 ±1.3	79.0 ±0.9
ODIN	83.7 ±0.7	78.9 ±1.0	81.9 ±0.9
Mahalanobis	91.2 ±0.4	90.7 ±0.4	91.8 ±0.3
KLoSNet (Ours)	**93.9** ±0.4	**93.1** ±1.1	**94.4** ±0.3

(c) CIFAR-100 with VGG-16

Method	CIFAR-10 Mis. (↑)	SVHN OOD (↑)	SVHN Mis+OOD (↑)
MCP	82.9 ±0.8	70.8 ±3.9	**81.3** ±2.0
Entropy	82.2 ±0.8	**72.9** ±3.9	**81.5** ±2.0
ConfidNet	84.4 ±0.6	68.0 ±3.4	80.8 ±2.0
Mut. Inf.	78.9 ±0.8	**72.7** ±4.9	79.5 ±2.5
Diff. Ent.	80.2 ±0.8	**72.4** ±4.9	80.2 ±2.5
EPKL	78.8 ±0.8	**72.7** ±4.8	79.4 ±2.4
ODIN	82.1 ±0.8	**72.0** ±3.8	**81.3** ±1.9
Mahalanobis	84.0 ±0.2	**73.4** ±5.6	**83.2** ±2.5
KLoSNet (Ours)	**86.7** ±0.4	70.4 ±5.7	**83.5** ±2.8

(d) CIFAR-100 with ResNet18

Method	CIFAR-100 Mis. (↑)	SVHN OOD (↑)	SVHN Mis+OOD (↑)
MCP	84.9 ±0.8	79.6 ±1.0	83.0 ±0.9
Entropy	84.6 ±0.8	79.6 ±1.1	82.8 ±0.9
ConfidNet	90.7 ±0.4	84.6 ±1.1	88.6 ±0.6
Mut. Inf	80.6 ±0.6	77.0 ±1.2	79.4 ±0.9
Diff. Ent	82.7 ±0.6	78.3 ±1.2	81.1 ±0.9
EPKL	80.2 ±0.6	76.8 ±1.3	79.0 ±0.9
ODIN	83.7 ±0.7	78.9 ±1.0	81.9 ±0.9
Mahalanobis	91.2 ±0.4	90.7 ±0.4	91.8 ±0.3
KLoSNet (Ours)	**93.9** ±0.4	**93.1** ±1.1	**94.4** ±0.3

Figure B.1: **Results with SVHN as OOD dataset** (% mean AUROC and std. over 5 runs).