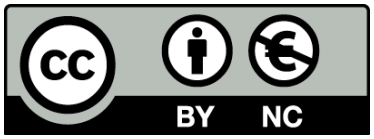
UNIVERSITAT DE BARCELONA

Modelling Uncertainty in Black-box Classification Systems

José Mena Roldán



Modelling Uncertainty in Black-box Classification Systems

José Mena Roldán

UNIVERSITAT DE
BARCELONA

Doctor of Philosophy

Facultat de Matemàtiques i Informàtica

Universitat de Barcelona

2020

Directors	Dr. Oriol Pujol Dept. de Matemàtiques i Informàtica Universitat de Barcelona
	Dr. Jordi Vitrià Dept. de Matemàtiques i Informàtica Universitat de Barcelona
Tutor	Dr. Oriol Pujol Dept. de Matemàtiques i Informàtica Universitat de Barcelona
Responsible from Eurecat	Dr. Marc Torrent Unitat Big Data and Data Science Eurecat, Centre Tecnològic de Catalunya

Abstract

Currently, thanks to the Big Data boom, the excellent results obtained by deep learning models and the strong digital transformation experienced over the last years, many companies have decided to incorporate machine learning models into their systems. Some companies have detected this opportunity and are making a portfolio of artificial intelligence services available to third parties in the form of application programming interfaces (APIs). Subsequently, developers include calls to these APIs to incorporate AI functionalities in their products. Although it is an option that saves time and resources, it is true that, in most cases, these APIs are displayed in the form of black-boxes, the details of which are unknown to their clients. The complexity of such products typically leads to a lack of control and knowledge of the internal components, which, in turn, can drive to potential uncontrolled risks. Therefore, it is necessary to develop methods capable of evaluating the performance of these black-boxes when applied to a specific application.

In this work, we present a robust uncertainty-based method for evaluating the performance of both probabilistic and categorical classification black-box models, in particular APIs, that enriches the predictions obtained with an uncertainty score. This uncertainty score enables the detection of inputs with very confident but erroneous predictions while protecting against out of distribution data points when deploying the model in a productive setting.

In the first part of the thesis, we develop a thorough revision of the concept of uncertainty, focusing on the uncertainty of classification systems. We review the existing related literature, describing the different approaches for modelling this uncertainty, its application to different use cases and some of its desirable properties. Next, we introduce the proposed method for modelling uncertainty in black-box settings. Moreover, in the last chapters of the thesis, we showcase the method applied to different domains, including NLP and computer vision problems. Finally, we include two real-life applications of the method: classification of overqualification in job descriptions and readability assessment of texts.

A mis padres.

A la Imma i la Vera, per les estones perdudes.

Acknowledgements

This work has been partially funded by the Spanish projects TIN2016-74946-P, TIN2015-66951-C2 (MINECO/FEDER, UE), RTI2018-095232-B-C21 and 2017 SGR 1742, and by AGAUR of the Generalitat de Catalunya through the Industrial PhD grant.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*José Mena Roldán*)

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

BB	Black-box
CV	Computer Vision
DL	Deep Learning
ML	Machine Learning
MLE	Maximum Likelihood Estimate
NLP	Natural Language Processing
OOD	Out of Distribution sample
RV	Random Variable

Greek Symbols

α	Diriclet concentration parameters
β	The uncertainty correction factor estimated
λ	Regularization parameter
μ	location parameter of a Gaussian RV
σ	scale parameter of a Gaussian RV

Other Symbols

$\mathbb{R}$	Real numbers
$\mathcal{N}$	The Gaussian Distribution

Dir	The Dirichlet Distribution

Roman Symbols

$\mathcal{D}$	Dataset
D	Output dimensionality
H	Hidden dimesionality
L	Number of network layers
x^*	Prediction input data point for model
x_n	Training input data point for model
y^*	Prediction output data point for model
y_n	Training label for model
$\mathbf{A}$	matrix
$\mathbf{a}$	vector
$\mathbf{W}$	Weight matrix
$\mathbf{X}$	Dataset inputs(matrix with N rows, one for each data point)
$\mathbf{Y}$	Dataset labels(matrix with N rows, one for each data point)
a	scalar

Chapter 1

Introduction

Contents

1.1 Context

Machine learning as a service (MLaaS) defines a set of services delivered by Cloud providers that aim to deliver Artificial Intelligence solutions through a kit of tools and APIs. These AI solutions include a wide variety of applications, from computer vision to natural language processing, based on last generation technologies like Big Data platforms or Deep Learning. The main advantage of these services is that the provider takes care of all the details in their data centres. These details include storing the data, training the models or the infrastructure required to emit the predictions. MLaaS thus allows the customer to get started quickly with machine learning without having to provision their servers, to install software or having advanced data science skills, applying the same paradigm as in other cloud products. This relaxation in the requirements is especially relevant in Deep Learning, where the needs for data, computational resources and talented engineers can be very demanding.

Moreover, the increasing demand for this type of services is fostering the growth of the MLaaS market. All the big players in the cloud industry are currently offering ML platforms and APIs, including Microsoft, Amazon, Google or IBM. According to

different market analysis, the MLaaS market is still growing with a foreseen CAGR of around 40% for the next years. This type of products is especially relevant for medium and small companies that may do not have enough capacity to put resources into the development of ML models but can take advantage of these solutions for optimizing their supply chain. Still, big companies can also benefit from these solutions when looking for experimentation environments before committing for final developments or for reducing direct costs. Moreover, public administrations, research institutes and academia are a potential recipient for these technologies, and applications can be found in business so unlike as defence, healthcare, retail, finance or agriculture.

The variety of services delivered by MLaaS platforms is great: risk management, fraud detection, automatic translation, recommender systems, user segmentation, and almost any ML model. These Machine Learning models include different types depending on the training they follow: from supervised, semi- or unsupervised learning, including reinforcement learning; and different types of applications including regression, classification, anomaly detection or clustering.

Figure 1.1: Machine learning pipeline

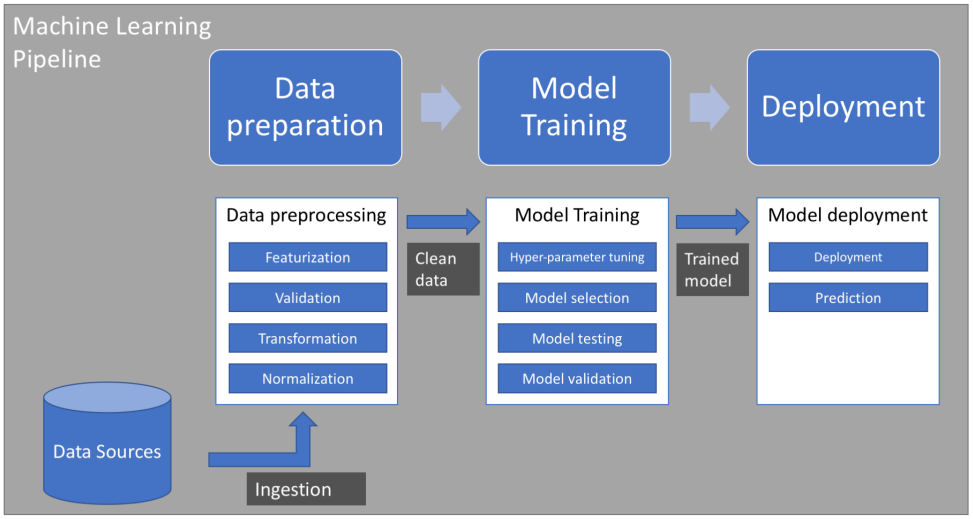


Figure 1.1 displays a typical MLaaS architecture. The diagram includes all the phases:

- Data Preparation. Usually, MLaaS provide connectors for a variety of data sources, including files with different formats, databases, or distributed storage

systems. This phase also includes the ETLs required to transform the data, join different sources or clean the data, including handling missing values, normalizing, validating formats, and all the preparation required for each case. All these processes run in the cloud, and it is the provider who manages the data storage, access or back-ups. The platform also takes care of handling both applications that operate using batch processing or that handle real-time streams.

- Build and Training the model. Once the data is ready, it is time for the learning properly said. This phase will include model selection, choosing the attributes to be included, optimization of hyper-parameters for each model, and validation of the results.

- Deploy and Predict. Finally, the trained and validated models have to start emitting predictions. The process of controlling different versions of the model, orchestrating training processes, and all the engineering work of delivering ways to consume the model is carried out by the provider in this later phase. It may also include visualization and monitoring tools to allow the end-user to gain control over the operation of the predictive models.

As one can see, there is a lot to do when developing an ML model beyond training it. It is a very resource-demanding task that not all organizations can afford. The first resource that it requires is time. Time for acquiring the data, label it for supervised problems, designing and optimizing the models. Plus developing the serving modules required to start using and monitoring them. The critical second resource is data. Particularly relevant in DL settings. Having access to enough data from the target application is vital for correctly training the models. In supervised settings, the process of labelling the data might imply a significant effort in terms of cost, time and access to domain experts. Besides, another aspect to bear in mind is the computational resources needed to handle significant volumes of data or to train the models. Finally, the specific skillsets required for developing ML models are not very common among IT professionals these days, so companies are struggling for incorporating profiles such as Data Engineers or Data Scientists to their developing teams.

Therefore, for companies that might have difficulties affording the requirements mentioned above, taking advantage of MLaaS might be a sensible choice. Depending on the customer needs and available resources, vendors offer different approaches for using the MLaaS pipeline. MLaaS might appear as a platform where the user designs, develops and test their models, benefiting from the provided infrastructure

while abstracting from the engineering details and the operation of the system. Alternatively, customers might see the services delivered as APIs that act as completely opaque **black-boxes**, where the user only has access to the outcomes of the provided methods. Very often, MLaaS vendors offer a trade-off between the two: pre-trained models with fixed data formats and outputs that still allow some personalization by supplying custom data examples to fine-tune the existing models.

As practitioners, this new offering of available pre-trained models in the shape of APIs opens a new scenario where the issue might be on selecting the appropriate tool for our given problem, more than developing a custom one. Take, for example, the case where we have a corpus of product reviews that we want to classify according to their sentiment. Performing a quick search, one can find solutions from all the prominent vendors: Amazon Comprehend [1], Microsoft's Text Analytics [2], Google's Natural Language API [3] or IBM's Watson [4], or many other specialized companies like Twinwords, Aylien or MeaningCloud, among many others. Each vendor has their pricing policy, including free samples and different plans, and delivers different outcomes that might include additional metrics like polarity of the sentiments, for example.

In this use case, and many others, the number of choices is significant. In such a scenario, some questions arise: which is the option that works better in our problem? Which are the criteria to follow when choosing among the different implementations? Will it work for all the instances of the target data set? How will it adapt to potential changes in the distribution of the data once integrated? For answering these and other questions, a new set of tools that allow practitioners to assess the performance of these APIs in particular target applications is required.

1.2 Goals

The main goal of the present work is to furnish developers and practitioners considering the use of black-box classification systems in their applications with tools that allow them to assess the performance of these pre-trained models and to intervene according to this performance. In this thesis, this score will evaluate the confidence of the BB for predicting a given input. The principal hypothesis, in this case, is that con-

[1] https://aws.amazon.com/es/getting-started/tutorials/analyze-sentiment-comprehend/

[2] https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/

[3] https://cloud.google.com/natural-language/docs/analyzing-sentiment

[4] https://www.ibm.com/watson/services/tone-analyzer/

fident predictions will belong to right predictions while uncertain ones will correspond to erroneous results. Thus, the goals of this thesis are three-fold:

- Assess the confidence of a BB applied to a particular problem and domain. The provided confidence score will allow determining to what extent the predictions emitted are to be trusted or not for the target use case. By analyzing the confidence of the predictions, one can detect potential issues at some examples of the target domain, or a class level. This analysis is especially relevant when there is a need to include some mapping due to differences in the outcomes of the black-box model and the target problem. Take, for example, the case where our problem is to classify a review on whether it is positive or negative, and the BB model predicts a rating score of 1 to 5 stars. In this case, predictions belonging to 3 stars might result very uncertain for the binary case.

- Provide actionable mechanisms based on the confidence score. If the practitioner integrates a BB into the target application, it is important to have a way to estimate the uncertainty of the prediction and actuate accordingly. When detecting an uncertain result, the system might opt for warning the user and ask for human supervision or rejecting the prediction when it is below a given confidence threshold. This mechanism is particularly pertinent is some fields of application where they might impact human lives, such as on healthcare DigitalReasoning [2019], education Turnitin [2019], or employment AvrioAI [2019]; Ideal [2019]. The increment on reports on failures and unintended effects Amodei et al. [2016]; Bostrom [2009]; D. Sculley and Young. [2014]. Having a way to detect and prevent these risks is, therefore, essential when incorporating third-party technologies to a system.

- Early detection of system degradation. After integrating the black-box and start serving predictions, changes may occur that may lead to a decrease in the performance of the system. Variations in the equipment that captures the images in a medical diagnose system, or the addition of new types of products to the catalogue in a review classifier, might introduce domain shifts that can impact on the confidence of the predictions. To have a way to monitor and detect these variations and be able to intervene in situations of degradation of the system may prevent and eventually mitigate when they end up in potential harm or risk.

1.3 Contributions

In this thesis, we present some contributions to the characterization of uncertainty in Deep Learning Classification systems, especially when served as black-boxes. The main contribution is an empirical method for estimating this uncertainty, complemented by some applications of this uncertainty and extended with the detection of out of distribution samples. Thus, the main contributions can be grouped into three:

- A method for estimating the uncertainty of black-box classifiers predictions. We introduce the concept of uncertainty in DL classification systems and highlight the constraints imposed in a black-box scenario. We propose a method for estimating the uncertainty of the predictions in such scenarios. The main idea is to turn each original output of the BB into a random variable. Here we propose to apply the concept of a wrapper that works around the original black-box, overlaying it and operating with its input and the generated output or prediction. This wrapper, based on a DL architecture, enables the obtention of a numerical uncertainty score of the prediction from this RV, which allows assessing the confidence of the BB for every input sample. In the context of this work, we studied different methods for implementing this wrapper, working with both Gaussian and Dirichlet distributions for modelling this RV, being the Dirichlet version the one that obtained better results, as we will show in the following chapters.

- Out-of-distribution detection in Deep Learning classification systems. The wrapper proposed for estimating the uncertainty in the predictions of the BB can learn the uncertainties from a training set of samples from the target domain. Even though this method ensures that the wrapper will be able to generalize for other samples of the distribution of the target dataset, it might be interesting to complement this with the ability to detect points that might not be in this target distribution. Here, we complement the proposed wrapper with a new training method and loss functions that allow identifying OOD samples.

- Applications of uncertainty in BB Classification systems. After modelling the uncertainty of the predictions of a given classifier, we discuss different applications of this uncertainty.

  - Analysis of the uncertainty. First, we show how by analyzing this uncertainty, we can determine to what extent the API applied is appropriate for

the target application. We can analyze the results obtained at the class level and identify categories with potentially higher risks of miss-classification. We can also compare different APIs to determine which fits better in our problem.

- Rejection. Another application of this uncertainty, once the API is in use, might be to reject uncertain predictions. We propose a method of finding the optimal rejection point based on the uncertainty found. This method will allow to reject the prediction directly or to ask for the intervention of an expert or an alternative decision system.
- Attribution. Another aspect that can be of interest is to find an explanation for the uncertainty score obtained. Based on explainability methods, we propose a method for attributing the elements of the input that contribute more to the uncertainty. This attribution can be useful, in the case of NLP, to identify which words are making the prediction more uncertain, for example.

1.4 Thesis Layout

The first part of this thesis, Chapter 2, will be devoted to introducing the concept of uncertainty in classification systems, and its peculiarities when applied to black-box systems. Chapter 3 includes a description of the methods developed during the course of the present work, showing the evolution of the work carried out and the incremental complexity of the developed system.

Chapter 4 is devoted to showcasing the applications of uncertainty and to highlight potential implications of its usage on real scenarios. In Chapter 5, we present an implementation of a variation of the proposed method in a real project where we applied rejection based on the uncertainty of predictions in the analysis of overqualification in the description of job offers. Chapter 6 describes another application of the proposed method, focused on this case to assess the readability of texts. A relevant aspect of this application is that, in this case, the black-boxes do not correspond to ML models but readability indexes, demonstrating how extensive are the potential applications of the method. Finally, Chapter 7 concludes the work, pointing to possible research lines that can derive from this thesis.

Some of the results presented in this thesis, including some of the experiments run,

are available at https://github.com/menajosep/dirichlet_uncertainty.

Chapter 2

Uncertainty in Classification Systems

Contents

As mentioned in the introduction, the goal of this thesis is to asses the performance of pre-trained classification models delivered as black-boxes, as part of APIs or MLaaS solutions. In many situations, these systems will deliver a probability for an input to belong to a given class. Sometimes the system returns this probability as a measure of the confidence of the predictions - e.g. a high probability like 0.91 is considered as a right and confident prediction in a binary classification setting. In contrast, in the

same setting, a probability of 0.51 might mean that the system is not very sure about the predicted class.

In any case, the output is a single number: a categorical value or a probability, but just a point estimate. In contrast, as we will see, the method proposed here transforms this output into a random variable following probability distribution. To some extent, we are going from a frequentist approach of parameter estimation to a Bayesian approach.

Statistical inference is one of the main branches of Statistics. Its goal is to make inferences of populations based on random samples. These inferences are possible by modelling the population as probability distributions. One of the main tasks of statistical inference is the estimation of the parameters of these distributions. In statistics, there are two main philosophies for tackling this problem, see Cox [1946]: the frequentists and the Bayesian. The root of the difference between the two is in the concept of probability. While frequentists consider them as long-term frequencies of repeatable random events, the Bayesian concept of probabilities is broader in the sense that they consider them as degrees of belief, representing the uncertainty, in any event, no matter if this event is repeatable or not. In fact, one can consider frequencies as a particular case of probabilities, according to Jaynes [1986].

When tackling the problem of finding the parameters of the distribution, frequentist will base their approach on the data available. For example, if we model an attribute like the height of the population in a country as a Gaussian random variable, we must find values for the mean, μ, and the variance, σ^2. Frequentists will look at the sample of data available and will approximate these two parameters according to that data. Thus, they will find a point-estimate for the values of the parameters, what is called the maximum likelihood estimate, MLE.

On the other hand, Bayesians will consider the uncertainty of finding the unknown values of the two parameters. Instead of focusing on finding the value that best describes the available data, they will consider each parameter as a random variable. They will approximate the parameters for these new distributions. To some extent is like adding a new layer of complexity to the problem: we model the parameters of distribution with RV variables, and we estimate the parameters of the secondary distributions.

The Bayesian process follows the Bayes Rule. Parting from a prior distribution that holds our previous beliefs on the data, we use newly collected data to narrow the distribution around the (unknown) parameter value. It is using this concept of

probability as a degree of belief: we update those degrees as new data is coming. Following this update process, we will obtain an RV for each parameter. This process will enable not only the obtention of the expected value of the parameter but also a way to estimate the uncertainty of this estimate.

Therefore, the Bayesian process delivers a natural way of estimating the uncertainty of the predictions. However, frequentists also have a way to express this uncertainty: the confidence intervals. In confidence intervals, one defines a priori a confidence level, usually 90% or 95%, and then compute the margin of error for the chosen level. The trick here is that it is not correct to say that it covers the real value with a probability of 95%. It only says that in the long-run, 95% of the confidence intervals generated by following the same procedure will cover the correct value.

Hence, there is a subtle difference to what its Bayesian counterpart, credible intervals, promises: the interval has a 95% probability of holding the real value. Again, this is a matter of interpretation of the concepts involved in the definition of these intervals. On the frequentists side, the parameter to estimate is considered as fixed, while the intervals and the data used are RV. On the contrary, Bayesians consider that it is the parameter what is an RV and leave the rest as fixed.

Moreover, there is also a difference in the assumptions previous to the definition of the intervals. In the frequentist case, it is the confidence level which is defined before to have access to any data. In the Bayesian approach, one must define a prior over the distribution of the data. Usually, the confidence levels are chosen arbitrarily, while selecting a prior allows bringing previous knowledge of the problem that can help on having better results.

In this work, we do not want to take sides in the conflict between the two philosophies (we are not so strict as in Briggs [2012]), but we still consider that for our goal, the Bayesian approach is more natural. We want to have a way to estimate the uncertainty of the predictions of a model given samples from a target data distribution. Thus, by considering these predictions as RVs, we can later analyse the corresponding distributions to measure this uncertainty.

2.1 Problem formulation

Before going more rooted in the definition of uncertainty, first, we would like to describe the problem we try to solve, i.e. assess the quality of classification systems. In such systems, we have a training data set, $\mathcal{D} = \{(x_i, y_i)\}, i = 1 \dots N$, where $y \in 1 \dots C$

with C the number of categories, and $x_i \in R^d$. Given a new sample of data points $\mathbf{x}^*$, we want to predict their labels y.

In this work, we focus on discriminative models, where the goal is to capture the distribution that generated the outputs based on the data. We assume that from the observed data in Y, X and $\mathbf{x}^*$, we want to infer the distribution parameters that maximizes the following conditional:

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = p(y^*|\mathbf{x}^*, X, Y) \tag{2.1}$$

To estimate this output distribution, we employ a model with parameters $\mathbf{W}$. Applying the Law of Total Probability we have:

$$p(y^*|\mathbf{x}^*, X, Y) = \int_{\mathbf{W}} p(y^*|\mathbf{W}, \mathbf{x}^*, X, Y) p(\mathbf{W}|\mathbf{x}^*, X, Y) d\mathbf{W} \tag{2.2}$$

At this point, usually, we can make two assumptions for making this problem solvable:

- Assumption 1: when learning the parameter $\mathbf{W}$, usually we do not have access to the test data $\mathbf{x}^*$, so we approximate:

$$p(\mathbf{W}|\mathbf{x}^*, X, Y) = p(\mathbf{W}|X, Y) \tag{2.3}$$

- Assumption 2: we assume that the model $\mathbf{W}$ has been able to capture the distributions in X, Y, so when predicting the output y^* we approximate by:

$$p(y^*|\mathbf{W}, \mathbf{x}^*, X, Y) = p(y^*|\mathbf{W}, \mathbf{x}^*) \tag{2.4}$$

Bearing these assumptions in mind, we end up with:

$$p(y^*|\mathbf{x}^*, X, Y) = \int_{\mathbf{W}} p(y^*|\mathbf{W}, \mathbf{x}^*) p(\mathbf{W}|X, Y) d\mathbf{W} \tag{2.5}$$

Once reached this point, there are two main approaches to approximate the resulting integral. One approach is to consider a discrete set of models $m \in W$ such as:

$$p(y^*|\mathbf{x}^*, X, Y) = \sum_{m \in \mathbf{W}} p(y^*|\mathbf{W}, \mathbf{x}^*) \alpha_m \tag{2.6}$$

This approach is followed by ensemble methods like Random Forests, Bagging, or similar others.

On the other hand, there is the traditional approach of estimating a model W^* such as it maximizes the likelihood of the predictions. Assuming that model W^* is trained until finding its optimal set of parameters, we have that:

$$\begin{aligned} \mathbf{W}^* &= \underset{\mathbf{W}}{\operatorname{argmax}}\, p(\mathbf{W}|X,Y) \\ p(\mathbf{W}|X,Y) &= \begin{cases} 1 & \mathbf{W} = \mathbf{W}^* \\ 0 & otherwise \end{cases} \end{aligned} \tag{2.7}$$

Thus, in prediction time, the target probability is approximated as follows:

$$p(y^*|\mathbf{x}^*,X,Y) = p(y^*|\mathbf{x}^*,\mathbf{W}^*) \tag{2.8}$$

The last expression describes the prediction obtained using the MLE of the optimized model. This resembles the previous discussion where we confronted the frequentist and Bayesian approaches. Again, in this case there is no direct way to measure the uncertainty of the predictions thus obtained.

Moreover, let us remark that the expression in 2.8 corresponds to the type of predictions obtained when using a black-box predictor. As stated in MacKay [1992b], in the classification task, we should have two outputs: one for the prediction of the most probable model, MLP, and another one that takes into account the uncertainty of the prediction. In this work, we consider the black-box as the MLP and proposed ways to estimate the associated uncertainty. To be able to measure the uncertainty of those predictions, we should go back to the expression in 2.5. In the next section, we will see how the different parts of this integral link with different aspects of the uncertainty of the predictions.

2.2 Definition of Uncertainty

In this section, we review the concept of uncertainty. The Collins Concise English Dictionary defines the concept of uncertainty as "the state or condition of being uncertain", relating it with words like doubt, hesitancy, unpredictability or indeterminacy. When related to Machine Learning models, and classification systems in particular, uncertainty is associated with a lack of confidence in the predictions emitted by a model.

In section 2.1, we described the process followed to estimate the probability of the output of a classifier. In equation 2.5, we can observe how two terms contribute to this integral:

- $p(y^*|\mathbf{W}, \mathbf{x}^*)$: given a model with parameters $\mathbf{W}$, this term defines the probability of the output for the inputs $\mathbf{x}^*$ received during test time. Remember that, in this case, we assumed that model $\mathbf{W}$ has been able to capture the distributions in X, y. Here, therefore, by considering y^* as an RV, we want to estimate the variance in the prediction caused by the noise inherent in $\mathbf{x}^*$.

- $p(\mathbf{W}|X, Y)$: the second term estimates the probability of the model with parameters $\mathbf{W}$ conditioned by the training dataset $\mathcal{D}$. Also, as commented in section 2.1, we assume that we train the model using only X, Y, as we do not have access to $\mathbf{x}^*$ or y^* during the training. At this point, there is the option of taking the model which parameters $\mathbf{W}$ maximize this probability, the frequentist approach, or we can consider these parameters as RV to have a way to express this probability, the Bayesian approach.

According to this definition, we can identify different sources of uncertainty. First, let us review the assumption that the model $\mathbf{W}$ captures the distribution of the training data. What if the expressivity of the model is not enough to capture the original distributions? What if $\mathcal{D}$ is not a representative sample of the whole data distribution? This links with the second assumption made, where we consider enough to train the model with $\mathcal{D}$ and then apply that model for predicting $\mathbf{x}^*$. What if the distributions in $\mathbf{x}^*$ or y^* differ from the training ones? How to deal with the inherent noise induced by the acquisition or labelling of the data?

The answers to these questions and more lead to different sources of uncertainty. Depending on the question that they try to answer, in the literature, we find different names for each type of uncertainty. One possible classification, as discussed in Faber [2005]; Senge et al. [2014]; Kendall and Gal [2017], is considering the trained model as an expert that takes decisions and define the following types of uncertainty:

- **Epistemic uncertainty** corresponds to the uncertainty originated by a lack of knowledge ("episteme" is the Ancient Greek word for knowledge). This lack of knowledge can come from two different sources. One is the lack of evidence to learn. In this case, the model, our expert here, has not seen enough cases to have a proper knowledge to emit confident predictions. Another source of lack of knowledge might come from the ignorance of what kind of model suits better for the given problem.

- **Aleatoric uncertainty**. Aleatoric, or aleatory uncertainty, refers to statistical variability and effects that are inherently random in the data generation process,

i.e. due to measurement noise or inherent ambiguity of the data. There are two types of aleatoric uncertainty according to the following assumptions:

- **Homocedastic uncertainty** measures the level of noise derived from the measurement process. This uncertainty is constant for all the data.
- **Heteroscedastic uncertainty** measures the level of uncertainty caused by the data. For example, in the case of NLP, this can be explained by the ambiguity of some words or sentences.

Another criterion used to categorize uncertainty is to which extent we can reduce it:

- **Reducible**. This type of uncertainty can decrease as we go incorporating more examples to our training set, for example. Hence, gaining more knowledge about the problem, we reduce the derived uncertainty. The same happens with the type of model applied. By increasing the complexity and tuning architectures and hyperparameters, the uncertainty associated with the capacity of the model shrinks.

- **Irreducible**. The uncertainty is inherent to the data. Even after adding more examples or increasing the capacity of the models, still, this type of uncertainty will not decrease. In this case, uncertainty is caused by the statistical variability and effects that are inherently random in the process of acquiring the data or in the data itself.

In many works, epistemic uncertainty is associated with reducible and aleatoric with irreducible uncertainties. By increasing the knowledge, we reduce the epistemic uncertainty, while even when we master the problem, the aleatoric uncertainty will never get smaller. The problem is that this distinction between epistemic and aleatoric is not always a matter of blacks and whites. As stated in Faber [2005]; Der Kiureghian and Ditlevsen [2009], this categorization is not something static. There might be situations where aleatoric uncertainty turn into epistemic uncertainty as the model evolves. There are situations where changes in the measurement of the data, e.g. adding more precision, will reduce the aleatoric uncertainty.

Moreover, aleatoric and epistemic uncertainties are not disjunct categories. As reflected in Hora [1996], both types of uncertainty are related, as aleatoric uncertainty is conditioned many times by the underlying epistemic one. Take for example the equation 2.5. One can consider that the term, $p(y^*|\mathbf{W}, \mathbf{x}^*)$, models the aleatoric uncertainty,

also known as the data uncertainty, as it estimates the variability in the output. On the other hand, the second term, $p(\mathbf{W}|X,Y)$, can be seen as modelling the epistemic uncertainty, sometimes called the model uncertainty. A closer look at the aleatoric expression, though, will reveal how the model parameters condition the output. Thus, we can appreciate how there exists a relation between the two types of uncertainties that makes it very hard to estimate them separately.

This link between the two is especially relevant in the case of using black-box models. In this case, the parameters of the model are given and fixed, and there is no way to reduce the epistemic uncertainty. Therefore, we have to focus on the term $p(y^*|\mathbf{W},\mathbf{x}^*)$ and try to measure the aleatoric uncertainty associated.

In the next section, we will review the role of uncertainty in Deep Learning and different approaches that tried to measure the different types of uncertainties described so far.

2.3 Related Work

The present section surveys the role of uncertainty in classification systems mainly focused on Deep Learning Systems. We include works that propose different approaches for incorporating an estimate of the uncertainty together with the prediction emitted. These works include ways to model the different types of uncertainty listed in the previous section, focusing in some cases in both epistemic and aleatoric, or tackling each of them separately.

2.3.1 Epistemic Uncertainty

One of the main types of uncertainty, the epistemic one, is also known as model uncertainty because it measures the confidence of the model according to the amount of data seen while training it or its expressivity and capability of capturing the complexity of the data.

As stated in the introduction, a natural approach for including the study of uncertainty in predictive models is to apply the Bayesian inference method. As described at the beginning of the chapter, this Bayesian approach will consider all parameters as unknown constants and will model them as RV. A neural network implemented by considering a prior over its parameters and that approximates the posterior using Bayesian inference is called a Bayesian Neural Network. In this section, we review some of the

approaches utilised for inferring this posterior.

2.3.1.1 Laplace approximations

One of the first works to implement a Bayesian NN was MacKay [1992c] where David MacKay ported his previous work MacKay [1992a] to the backpropagation learning method commonly used in DL. The goal, in this case, is to compare among models and to empirically obtain values for the regularisation terms.

Beyond the commonly used approach of hold-out data or cross-validation, they propose to include an additional step in the inference process called model selection. This model selection includes also the selection of hyper-parameters like regularization terms, terms that are included in the loss function to, for example, penalize large weights and achieve a smoother or simpler mapping, Rumelhart et al. [1986]. This additional step is part of a Bayesian inference schema where he sets a Gaussian prior over the parameters and uses Bayes to estimate the posterior of the *evidence*, i.e. $p(D|\mathbf{W}, \mathcal{R})$, where $\mathcal{R}$ corresponds to the set of regularisation parameters. Then, he uses the value of this evidence for comparing models.

More relevant to this work is the extension he did for classification systems in MacKay [1992b]. The schema followed is the same as in the previous work. However, in this new one, he proposes that every classifier should have two kinds of outputs. In essence, the most probable, used for training the models, and a *moderated* one used for making decisions based on the predictions, compare models and use it even for active learning schemas.

In both works, he evaluates the evidence employing the Laplace's method with a Gaussian approximation from the properties of the *most probable* fit $\mathbf{W}_{MAP}$, and the error bars A^{-1}, which is the inverse of the Hessian matrix of the parameters of the loss function. For this Gaussian approximation, he made some assumptions, like considering the surface around the minima as quadratic, something that rarely occurs in real problems.

Moreover, approximating the Hessian, especially when using complex architectures with millions of parameters like those used in DL nowadays, is not trivial and, as reported by Bishop [1995], "the accurate evaluation of the evidence can prove to be very difficult. One of the reasons for this is that the determinant of the Hessian is given by the product of the eigenvalues and so is very sensitive to errors in the small eigenvalues". Besides, to obtain the moderated output, resulting from the combination of the Gaussian and the sigmoid non-linearity, they introduce a new approximation based

on a function of the variance parameter applied to the output activations.

2.3.1.2 Markov chain Monte Carlo approximations

MacKay's approach tackles the problem of solving the integral in equation 2.5 by employing Gaussian approximations for the prior and posterior distributions, which allows solving these integrals analytically. Because of the limitations posed in the previous section, sometimes this Gaussian approximation might not be recommended, especially in nowadays architectures. An alternative for solving the integral in equation 2.5 is to approximate the integral with the sum of L models:

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \frac{1}{L}\sum_{i=1}^{L} p(y^*|\mathbf{W}_i, \mathbf{x}^*) \tag{2.9}$$

Similarly to what they do in ensemble learning. In this case, the vector of parameters, $\mathbf{W}_i$, is sampled from the distribution $p(\mathbf{W}|\mathcal{D})$. In Neal [2012] reviews this approach and different ways to do this sampling. The proposed method is a combination of different sampling policies, called hybrid Monte Carlo. Briefly, the method combines a Gibbs sampling method for selecting the hyperparameters together with a Metropolis-like sampling method for selecting the parameters or the network. By using the hybrid Monte Carlo algorithm, one can generate suitable sampling vectors of parameters $\mathbf{W}_i$ in reasonable computational times. Once the method reaches an equilibrium, in prediction time, it is possible to sample predictions from the posterior. By analysing this sampling, it is possible to assess the uncertainty of the predictions given an input $\mathbf{x}_i$, what we called the aleatoric uncertainty of the model.

The problem with this kind of solutions is that they can take longer to converge because of the stochasticity inherent in the process, even when using smart approaches like hybrid Monte Carlo. Besides, it is difficult to obtain insights about the evidence of the model, as it was the case for Gaussian approaches burdening the capacity of these algorithms for model selection.

Following, in Chen et al. [2014], they work on the problem of convergence by using second-order Langevin dynamics with a friction term that counteracts the effects of the noisy gradient. They study the effect of applying this method on the classification of MNIST images, and they observe how sampling-based methods are better than optimisation-based ones, showing an advantage to Bayesian inference in this setting.

2.3.1.3 Variational Inference

A different approach for estimating the posterior is that followed in Hinton and Van Camp [1993], called *ensemble learning*. The premise is that the amount of information in the weights should be much less than in the output. It provides a way to derivate the error together with the amount of information in the weights. This assumption is especially relevant in problems with a small dataset. It seems to be limited to models that have a linear output, however. They apply the Minimum Description Length Principle that asserts that the best model of some data is the one that minimises the combined cost of describing the model and describing the misfit between the model and the data. In supervised NN the model cost is the number of bits it takes to describe the weights, and the data-misfit cost is the number of bits it takes to describe the discrepancy between the correct output and the output of the neural network on each training case.

For encoding the misfits, the output, it discretises the distribution in t-wide bins and each output unit is modelled by a Gaussian with 0 mean and a standard deviation of σ. Assigning a value for this σ to the root square deviation of the misfits from zero, it ends up with a misfit cost function of a constant. For coding the weights, again, they are modelled as a Gaussian with 0 mean and σ_w. Then the cost function is equal to an RMSE with quadratic weight decay. They complain though that this approach uses the same σ for all the weights, without taking into account the different levels of contribution that they might have, translated into a waste of bits for encoding this information. Comparing to MacKay [1992a], where they also looked at variations at the output once the model is trained, here they look at this variance during the training.

They use the information theory to encode the weights adding Gaussian noise to each weight. So, noisy weights are more natural to communicate, but they may lead to high variance on the output that would imply more cost at communicating the misfits. To measure the cost, they propose to use the $\mathcal{KL}$ divergence as a measure of the difference between the likelihood $\mathcal{P}$ and the posterior $\mathcal{Q}$. To determine the bits that it cost to communicate the weights, $\mathcal{C}(w)$, it takes each weight's posterior and binarises it. For each bin, it computes the information using the prior. Then it encodes the noisy weights by using the weights and obtain the exact posterior $\mathcal{Q}$. Next, it uses the $\mathcal{Q}$ and binarises the misfits to compute the transmission cost of the weights, $\mathcal{R}(w)$. The difference between one and another is, again, the $\mathcal{KL}$, which by using Gaussian can be solved analytically.

To learn the communication cost of the misfits, it needs the expected value of the

RMSE. Considering the noisy weights, obtained the root squared error might be hard to get. They do the calculations considering linear outputs and a single hidden layer. By having one single layer, they can create a matrix with the mean and variance for the outputs, by sampling with Monte Carlo. Then, for computing the RMSE, for each output unit, they use the sum of the mean for subtracting the ground truth, adding a variance term obtained from the table. The number of computations required for this approach when using DL it is unfeasible. A nice feature of the paper is that they do not require a unimodal Gaussian approach for the prior. For adapting to cases like a pike around 0 or 1, they propose to use a mixture of Gaussians instead. This approach for computing the posterior, therefore, is an alternative to restrictive Gaussian approximations, by using more adaptative functions.

Moreover, by employing the $\mathcal{KL}$ divergence, they can approximate the integral of the posterior analytically, avoiding the high computational requirements associated with the use of MCMC. We would also like to highlight the fact that it is addressing the estimate the epistemic uncertainty when modelling the distribution for the weights, and the aleatoric one when modelling the distribution of the outputs. We would also like to remark that the architectures used in the paper to illustrate the method are very simple, compared to the current DL models. Besides, a significant limitation of this work when applied to the classification problem is that they assume a linear output, which is not the case for probabilistic classifiers that use a softmax no-linearity.

To tackle the issues exposed in Hinton and Van Camp [1993], in Barber and Bishop [1998] they extend it to allow a Gaussian approximating distribution with a general covariance matrix, so it can capture the posterior correlations between parameters, while still leading to a tractable algorithm. Besides, for illustrating the method, they overcome the stated limitation of using linear output functions only and apply a variation of a sigmoid function(cumulative Gaussian).

Like in Hinton and Van Camp [1993], in Barber and Bishop [1998] they make use of the $\mathcal{KL}$ divergence to approximate the integral defined in equation 2.5. In order to capture the variance in weights and output, this method still applies a Gaussian approximation for both concepts. In the case of the output, they model the variance with a parameter β and the weights with a covariance matrix $\mathbf{A}$. Hence, the marginal likelihood is given by:

$$p(D|\beta,\mathbf{A}) = \int_{\mathbf{W}} p(D|\mathbf{W},\beta)p(\mathbf{W}|\mathbf{A})dW \tag{2.10}$$

Being this integration analytically intractable. Instead of applying a Laplace approxi-

mation as in MacKay [1992c], here the authors consider the logarithm of the marginal likelihood defined in equation 2.10. By introducing a variational posterior $Q(\mathbf{W})$, and making use of Jensen's inequality together with the convexity of the logarithmic function, they end up with the following decomposition of the marginal likelihood:

$$\log p(D|\beta, \mathbf{A}) = \mathcal{KL}(Q||P) + \mathcal{F}(Q) \tag{2.11}$$

The main idea is that this posterior $Q(\mathbf{W})$ represents a lower bound on the log marginal likelihood, as $\mathcal{KL}(Q||P) \geq 0$. By maximising the lower bound $\mathcal{F}(Q)$ for the parameters of Q, is equivalent to minimising the Kullback-Leibler divergence. When taken to the limit, if $Q(\mathbf{W})$ is equal to the true posterior, the $\mathcal{KL}(Q||P)$ will be 0. The trick here is to choose a form for the $Q(\mathbf{W})$ distribution such that they can evaluate the lower bound $\mathcal{F}(Q)$ efficiently.

Again, they choose a Gaussian model for the posterior $Q(\mathbf{W}) \sim \mathcal{N}(\bar{\mathbf{W}}, \mathbf{C})$, where $\mathbf{C}$ is a covariance matrix. By playing with decompositions of the covariance matrix $\mathbf{C}$ and making use of analytical techniques to reduce the different non-tractable integrals to one-dimensional ones, they end up with a set of derivates that can be applied to a gradient optimiser.

Still, so far, the variance parameters, $\beta, \mathbf{A}$, have been considered as constants. To include those parameters in the Bayesian framework, they consider a standard isotropic prior covariance matrix of the form $\mathbf{A} = \alpha\mathbf{I}$ and introduce hyperpriors given by Gamma distributions. Then by modelling the joint posterior $p(W, \alpha, \beta|D)$ by a factorised approximating distribution of the form $Q(w)\mathcal{R}(\alpha)\mathcal{S}(\beta)$, they can use an alternate optimization process to approximate these posterior distributions.

We can observe how the proposed method yet has a strong dependency on Gaussian distributions to take advantage of analytical operations that simplify the calculation of the integrals. The covariance matrix used for modelling the variance on the weights may represent a problem with modern architectures as it correlates with the number of parameters. While it is true that they provide an approximation for using a diagonal matrix, still it represents a significant amount of extra-parameters to optimise. Also, the method and problems used to illustrate it, focus on regression problems, leaving the classification problem unsolved.

One of the main problems of the methods described so far is that they are hard to implement in modern DL architectures, where the number of parameters is high. In Graves [2011], they tackle this problem by introducing a variational stochastic method. They propose to work with expectations of variational distributions that approximate

the true posterior, instead of trying to apply analytical solutions to find it.

For implementing the variational stochastic method that estimates these posteriors, they get back to the Minimum Description Length Principle introduced by Hinton and Van Camp [1993], applying it to the loss function for finding a trade-off between the prediction accuracy and the complexity of the model. This variational distribution takes the shape of a diagonal Gaussian $Q(\mathbf{W}|\beta)$. By applying the MDL Principle, the loss is:

$$\mathcal{L} = \mathcal{L}^N(\mathbf{W}, \mathcal{D})_{\mathbf{W}\sim Q(\beta)} + \mathcal{KL}(Q(\beta) \parallel P(\alpha)) \tag{2.12}$$

Where the left term corresponds to the accuracy objective, i.e. the negative log-likelihood(NLL) of the network, and the right term controls the complexity by trying to minimise the distance between the posterior and the prior. The NLL is computed by obtaining samples from the posterior $Q(\beta)$. In the paper, they run experiments with different combinations of posterior distributions, e.g. Gaussian and Delta, and priors, e.g. Uniform, LaPlace and Gaussian, obtaining the best results when both are Gaussian. To optimise the loss function, they apply an optimiser that can tolerate noisy gradient estimates, so they experiment with Steepest descent with momentum and RPROP. The method has been later criticised because it presents limitations due to a bias effect on the gradients caused by the MC samplings.

In the same line, in Hoffman et al. [2013] they also develop the concept of Stochastic Variational Inference for approximating the posterior of predictive models, emphasising the fact that the stochastic version of Variational Inference is better suited for massive data sets. By making use of the mean-field assumption where they consider all the parameters as independent variables, they achieve noisy but unbiased gradients and apply and stochastic optimisation process to estimate the posteriors similarly to what they did in Graves [2011]. In this case, though, the priors and posteriors used are modelled as Dirichlet distributions applied to topic modelling problems.

Built upon the work of Graves [2011] on Variational Inference, we find the Bayes by Backprop method described in Blundell et al. [2015] for learning a probability distribution on the weights of a neural network, focusing in this case in the regularisation effect when attributing a compression costs over the weights while obtaining the uncertainty associated with the predictions. In this case, the novelty is the reparametrisation trick, which consists of decomposing the parameters of Gaussian variational posteriors :

$$w = \mu + \log(1 + \exp(\rho)) \cdot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, 1) \tag{2.13}$$

Hence, by sampling using the mean after drawing MC samples from this distribution, they derivate the gradients of the mean, μ, and standard deviation, ρ, and use them to update the weights, again obtaining noisy but unbiased gradients. Note how, in this case, by doubling the number of parameters, two per weight, they handle an infinite number of models to an ensemble. As for the prior distribution they propose to use a mixture of Gaussians with 0 mean but different variances. An interesting thing to highlight in this work is that they illustrate the method with a classification problem, MNIST.

Beyond this seminal works, Variational Inference, VI, is a very active research topic. Different research lines are tackling problems like creating black-box versions of the VI or using other distributions rather than Gaussians, and how to apply the reparametrisation trick to them, Knowles [2015]; Ruiz et al. [2016]; Figurnov et al. [2018]; Jankowiak and Obermeyer [2018].

Another approach recently used in modern architectures is Flipout, Wen et al. [2018], which performs a Monte Carlo approximation of the distribution integrating over the weights. Flipout uses roughly twice as many floating-point operations as the reparameterisation estimator but has the advantage of significantly lower variance. By substituting the layers of a classifier by the equivalent Flipout ones, it is easy to apply VI to a classification system, including RNN or CNN architectures. Then, they use the Flipout gradient estimator to minimise the KL divergence up to a constant, also known as the negative Evidence Lower Boundor ELBO. It consists of the sum of two terms: the expected negative log-likelihood, which they approximate via Monte Carlo; and the KL divergence, which is added via regulariser terms which are arguments to the layer.

Another approach for applying VI in the case of the classification problem is that proposed in Rossi et al. [2018]. In that work, they extend the VI regression setting to k-class classification problems. They assume a one-hot encoding of the labels so that Y is a matrix of zeros and ones. Then, they model the posterior over classification functions by applying regression on a transformation of the labels, as proposed in Milios et al. [2018].

2.3.1.4 Expectation propagation

In this section, we review an alternative to VI introduced in Hernandez-Lobato and Adams [2015] called probabilistic backpropagation (PBP). Again, the goal of the method is to tackle the scalability problems of BNN, similarly to stochastic VI, while preserv-

ing their benefits, i.e. hyperparameter tuning, less overfitting, a measure for uncertainty and smaller volumes of training data. The main idea is to compute a forward propagation of probabilities through the network to obtain the marginal likelihood and then doing a backward computation of gradients of the marginal likelihood for the parameters of the posterior approximation.

They assume Gaussian distribution for the output conditioned to the input, the weights and the γ with a mean equal to the output of the NN, and deviation equal to γ^{-1}. They also assume priors over the weights modelled as Gaussians with mean equal to 0 and deviation λ^{-1}. Hyper priors of both γ and λ are Gamma distributions. The posterior distribution for the parameters W, γ and λ can then be obtained by applying Baye's rule. However, the exact computation of this posterior is not tractable.

For approximating this posterior, they propose the PBP. Similar to traditional BP, in the forward phase, the apply the weights to the input to produce the activations. As the weights are now RV distributions, the resulting activations are also random with untractable distributions. They approximate these activations with one-dimensional Gaussians that match their marginal means and variances. Even though the example they use for illustrating is again a regression, at the output, they do not evaluate the error but the logarithm of the marginal probability of the target variable. Based on this, they update the gradients of this quantity for the means and variances of the approximate Gaussian posterior.

The update of the weights, in this case, is complex. Applying the Baye's rule to update the beliefs based on the data requires an approximation with a more straightforward distribution function $Q(\mathbf{W})$. This new $Q(\mathbf{W})$ function is one that minimises the $\mathcal{KL}$ with the original update function. The parameters of this new function come from an update based on the original mean and variances. All in all, they approximate the posterior with a factored distribution of these $Q(\mathbf{W})$ and the Gamma hyperpriors, assuring that both the true posterior and the approximation share the same moments.

The Expectation propagation method, EP, improves this ADF implementation by iteratively incorporating each factor multiple times. On each pass, EP removes one factor from the posterior, re-estimates it, reincorporating it afterwards. One drawback of EP is that it requires to keep all the factors in memory for each sample, but as they operate in mini-batches, it is not a big issue.

In Li et al. [2015], they propose a variation of the method called Stochastic expectation propagation. In that article, the authors illustrate how the method can be applied to different classification problems.

One of the main advantages of using PBP is that it does not require hyperparameters as in traditional BP approaches, like learning rates.

2.3.1.5 MC Dropout

The methods presented so far all belong to the category of Bayesian Neural Networks. Mainly they present different approaches for solving the integral defined in Equation 2.5, solving it analytically assuming some approximations, using Monte Carlo by sampling different models or approximating the posterior with a simpler distribution. The weights or parameters of the network are assumed to follow a given distribution and different ways of learning the parameters of those distributions with variants of the backpropagation optimisation process of the NN.

In this section, we present an alternative that does not require to modify the architecture of the NN nor assume distributions over the network parameters. It builds upon the concept of dropout introduced in Srivastava et al. [2014]. Dropout was conceived as a method for reducing overfitting on the training of NN. The main idea is that, during training, some neurons are deactivated, forcing the active ones to assume the training effort of the deactivated. Imagine a company organised by departments with a person assigned to each of them: sales, HR, accounting,... picture now that the HR responsible breaks a leg. Instead of looking for a substitute, the manager splits the HR workload among the rest of the team members. Now the salesperson gets married, and the manager repeats the process. Hence, after some time, all the members of the team will know all the aspects of the company, and they will adapt better to unknown situations instead of getting overspecialised in their specific duties. The same applies to neurons in a NN using dropout. Each unit will have a probability of getting offline. On every training pass, based on this probability, some units will get off, and the rest will keep the training process for the examples of a given batch. Thus, by avoiding specialisation on the neurons, the NN will be more robust against overfitting problems.

Another point of view of the dropout method is the fact that on every training batch, the architecture and the parameters are different. The probabilities associated with each unit allow sampling of different models similar to the MC method proposed in Neal [2012]. In Gal and Ghahramani [2015], they take advantage of this sampling ability and propose to use dropout as an approximation to the probabilistic deep Gaussian process. With regular Dropout layers, as present in many architectures, dropout minimises the Kullback–Leibler divergence between an approximate distribution and the posterior of a deep Gaussian process.

As in VI methods, the idea is to reduce the KL divergence between the true posterior and a variational posterior $Q(\mathbf{W})$. In this case though, $Q(\mathbf{W})$ is a distribution over matrices $\mathbf{W}_i = \mathbf{M}_i \dot{diag}([z_{i,j} j = 1])$, and $z_{i,j} \sim Bernoulli(p_i)$. The binary variable $z_{i,j} = 0$ corresponds then to unit j in layer $i-1$ being dropped out as an input to layer i.

On prediction time, they sample T sets of vectors from the Bernoulli distribution $z_{i,j} \sim Bernoulli(p_i)$, thus obtaining different T values for y^*, and $E(y^*|\mathbf{x}^*)_{(Q)} \approx \frac{1}{T}\sum_{t=1}^{T} f^{\hat{\mathbf{W}}_j}(\mathbf{x}^*)$. This Monte Carlo estimate is what they call MC Dropout. The main advantage of the MC Dropout is the fact that it operates directly with dropout layers present in the definition of many NN architectures, so there is no need of extra parameters, modifications on the backpropagation process or computing complex Hessian matrices.

2.3.1.6 Ensemble and Bootstrapping

In the line of MC Dropout, where the approach is to have a way of having multiple models at the same time and then be able to sample from them and thus obtain an approximation for the posterior of the data, we find Ensemble Lakshminarayanan et al. [2017] and Bootstrapping Osband et al. [2016]. The idea is to approximate the integral of the posterior with a set of models, similarly to equation 2.6.

In Ensemble methods, the idea is to train M models using different weights initialisation for each of them. Then, they train the models altogether sharing a joint loss function. On prediction time, the outcome of the ensemble model is the average of the individuals. By observing the variance of the outputs, one can also obtain a measure of uncertainty of the models.

Bootstrapping uses a similar strategy, but instead of replicating the same model M times, it uses a shared network with different heads. The training data, $\mathcal{D}$ is split into different subsamples $\tilde{\mathcal{D}}$ sampled uniformly with replacement from $\mathcal{D}$. It can be seen as a data dropout, in the sense that each data point is included in the training of a particular head depending on a probability $\sim Bernoulli(p)$. Thus, each head is trained only on its bootstrapped $\tilde{\mathcal{D}}$, letting the shared network to learn a joint feature representation across all the data.

Both Ensemble and Bootstrapping represent methods that do not require significant changes in the architecture of the network while providing means for obtaining the uncertainty of the predictions. In particular, despite its simplicity, uncertainties obtained with the Ensemble method manifest higher robustness to posterior domain

shifts according to Snoek et al. [2019].

2.3.2 Heteroscedastic Aleatoric Uncertainty

Approaches surveyed in the previous section had a common goal which was to estimate the uncertainty of the models. By imposing probability distributions over the weights or proposing approximations like using dropout or ensembles of different networks, they measured to which extent the model was sure about its predictions based on its capabilities and limitations.

This model uncertainty is what we called epistemic uncertainty in section 2.2, as associated with the term $p(\mathbf{W}|X,y)$ in equation 2.5. This term expresses a probability of the weights conditioned by the training data. As we said, this epistemic can be reduced by training with more data or by increasing the expressiveness of the models.

However, let us remark here that this work tackles uncertainty associated with black-box models. Thus, we can not reduce the epistemic uncertainty by including additional data, nor increasing the capabilities of the model, as the model is already trained, and we have no option of modifying it.

In this black-box scenario, we have to focus on to which extent the given model adapts to our use case, specifically to our data. Hence, we focus on another type of uncertainty: heteroscedastic aleatoric uncertainty. From equation 2.2, we analyse now the term $p(y^*|\mathbf{W},\mathbf{x}^*)$, where we evaluate the probability of the output of a given model with parameters $\mathbf{W}$ predicting testing input data. The so-called data uncertainty.

Despite methods in section 2.3.1 mostly studied epistemic uncertainty, and mainly on regression problems, some already considered the uncertainty of the data. In the introduction of Variational Inference, when describing the work in Hinton and Van Camp [1993], we saw how they split the contribution of the noisy weights and the data-misfit. In this case, they centred on a regression task, and they were approximating the output distribution of the square error with a Gaussian posterior distribution. One could analyse the uncertainty on the output activations by observing their posterior distribution. However, it might be difficult to isolate only the aleatoric part, as they are trained together with the posteriors over the weights.

Moreover, still, there would the problem of adapting the method to classification were the output is not a single unit and includes non-linearities like the standard softmax function. In recent VI approaches, Zhang et al. [2018], they tackled this problem by using categorical distribution at the output, instead of Gaussian ones. Nevertheless,

the training process includes both the weights and outputs as one.

2.3.2.1 Gaussian output models

With regards to this separation of concerns, between model and data uncertainties, we find a relevant contribution in Gal [2016]; Kendall and Gal [2017]. In those works, they extend the application of dropout as a method for estimating the model uncertainty in NN introduced in Gal and Ghahramani [2015], bearing the classification problem and aleatoric uncertainty explicitly.

In their proposal for uncertainty in classification tasks, instead of modelling the output distribution, $y^* \sim Categorical(p)$, they model the noise at the logits u activation, avoiding thus to deal with the softmax non-linearity.

$$\mathbf{u} \sim \mathcal{N}(f^{\mathbf{W}}(\mathbf{x}^*), diag(\sigma^2(\mathbf{x}^*))) \tag{2.14}$$

Here, each logit is modelled with a Gaussian distribution, with mean equal to the output for the corresponding class as obtained from the model $f^{\mathbf{W}}(\mathbf{x}^*)$ with parameters W, and the variance modelled as a diagonal matrix with a parameter σ that is learned from the input data. The resulting output of the model is:

$$p = softmax(\mathbf{u}) \tag{2.15}$$

And the associated likelihood is obtained from the expectation of this probability by applying MC to this distributions, drawing M samples from the Gaussian modelling the logits:

$$\mathbb{E}[p] = \frac{1}{M}\sum_{m=1}^{M} softmax(\mathbf{u}_m) \tag{2.16}$$

Next, the model is trained with a cross-entropy loss function that includes the resulting log-likelihood. To able to apply backpropagation to the training including these new Gaussian RV, they apply the reparametrization-trick introduced in Blundell et al. [2015]:

$$\mathbf{u} = f^{\mathbf{W}}(\mathbf{x}^*) + \sqrt{diag(\sigma^2(\mathbf{x}^*))} \cdot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0,1) \tag{2.17}$$

Finally, to obtain the model parameters, W, they combine the MC dropout method with this aleatoric cross-entropy loss, allowing to model aleatoric and epistemic uncertainty jointly.

In the original paper, Kendall and Gal [2017], they focused on Computer Vision, analysing the role of uncertainty in image classification and segmentation. In Xiao and Wang [2019], they extend this work by studying its impact in the NLP field. It is interesting how they study the different contributions of epistemic, aleatoric and a combination of both uncertainties. They run different NLP experiments: Language modelling, Sentiment analysis and Named Entity Recognition, NER. In those experiments, they observe how the combination of including a combination of both epistemic and aleatoric uncertainties in training contributes to achieving better results at the classification tasks included. Especially relevant to our work are the results obtained with NER, where the best results are obtained when incorporating only the aleatoric uncertainty.

2.3.2.2 Dirichlet models

In the method described in the previous section, we can observe two limitations. First of all, contrary to what happens in regression, where they directly approximate the output distribution, in classification, they model the aleatoric uncertainty by modelling RVs at the logits level, not the output. They model each logit using a separate Gaussian RV. By modelling each logit variable individually, they are not considering possible interactions between classes, and they tie these RVs together by applying a softmax at the output.

Moreover, using Gaussian modelling as a generic recipe might impose unnecessary constraints in some problems. In the regressions setting, it is imposing a uni-modality that might not reflect the actual condition of the data. In this type of problems, some alternatives consider more elaborated approximations as detailed in Brando et al. [2019].

In a classification problem, ideally, we would like to model the Categorical output distribution, as composed by the probabilities associated with each output class. A natural interpretation of the Categorical output can bet to consider the natural conjugates of those distributions. Thinking of this output as elements in a simplex, one can contemplate a Beta for the binary case or a Dirichlet for the Multinomial.

The use of these distributions is the approach suggested in Sadowski and Baldi [2018]. In this work, they consider different applications of Beta and Dirichlet for different solutions. They use the Beta for problems with a real output with upper and lower bounds. More relevant to us is the solution they provide for multinomial classification systems. In those scenarios, they propose two alternatives. One is to model the output as a multi-headed Dirichlet, where each output unit corresponds to an α control

parameter of the overall distribution. This approach resembles the one in Kendall and Gal [2017], but instead of modelling at the logits level, they model each output unit. Alternatively, they propose also to model the softmax output directly with a Dirichlet and learn the parameters at once. For each option, they derive the corresponding gradients to enable training using standard optimisers. They detected stability problems due to the digamma terms in the gradient of the Dirichlet, that may become large for small α_i, causing large gradients. To fix those problems, they introduce different activation functions, i.e. Inverse-Linear (IL), Exponential-Linear (EL) and Exponential-Tanh (ET). However, even after including these activation units, they needed to add some regularizers to prevent very small α that generated overflow errors.

Another work that states using a Dirichlet as a probabilistic output layer is Gast and Roth [2018]. In this work, they include uncertainty in CNN by two means. First, they propose to change the output layer for a probabilistic one, in the line of Sadowski and Baldi [2018]. Second, by modelling the input as a Gaussian RV, they apply expectation propagation, similarly to Hernandez-Lobato and Adams [2015]. For implementing ADF in a CNN, they provide a variational approximation for all the layers involved: pool and max-pool layers, and Relu and leaky Relus. Thus, they achieve to model each output activation as a Gaussian RV.

For classification, they use a Dirichlet as the output probabilistic layer, building on top of the ADF outputs. To convert these Gaussian activations to the α_j control parameters that reign the Dirichlet, they propose a decomposition of these parameters based on the Gaussian output activations.

$$p(y|\mathbf{x}^*, \mathbf{W}) = Dir(\alpha(\mu, v), \quad \alpha(\mu, v) = \frac{m}{s}, \quad s = \sum_j \alpha_j^{-1} \tag{2.18}$$

and

$$m = softmax(\mu), \quad s = c_1 + c_2 \sqrt{\sum_j m_j v_j} \tag{2.19}$$

Where c_1 controls how peaky the distribution is and c_2 is an amplification factor for converting uncertainties into the Dirichlet scale. By including the m term in the variance, the corresponding uncertainty mainly comes from the class contributing most to the softmax activation. Hence, as the output is already a distribution, they can apply the maximum conditional likelihood learning that finds the parameters by maximising the conditional likelihood of the data under a predictive model.

Similarly to Kendall and Gal [2017], in Gast and Roth [2018] we find a way to model the aleatoric uncertainty, by using a Dirichlet at the output, and the epistemic by including Gaussian noise at the input and propagating the expectation with ADF. By combining both types of uncertainty, including minimal changes in the architecture of the network, they achieve the goal of converting a CNN that outputs point estimates to a probabilistic NN that outputs the predictions together with a measure of uncertainty.

An extension of Gast and Roth [2018] is proposed in Loquercio et al. [2020]. In there, they incorporate prior information coming from the sensors used in robotics to the Gaussian input noise. Their work focuses on regression problems and tackles both data and model uncertainties. For model uncertainties, they rely on the existence of dropout layers that enable MC dropout sampling. They claim to deliver a general framework applicable also to pre-trained networks, but, again, it relies on the dropout layers to find the most suitable dropout rates for the new problem.

In Thiagarajan et al. [2019], we find another extension for Gast and Roth [2018], adding in this case an uncertainty attribution method. This attribution means for identifying which features in the input data have a higher impact on the aleatoric uncertainty of the model. They focus on regression problems with tabular input data, so it is easy to illustrate the influence at an attribute level. For measuring this attribution, they suggest decomposing the function value f(x) as a sum of relevance scores, obtained using a first-order Taylor expansion of the function. Another benefit of this decomposition is that it can act as a regulariser when added to the loss, increasing its generalisation.

In Malinin and Gales [2018]; Chen et al. [2018], we find new approaches that build on top of Dirichlet distributions, focused in this case on the detection of out-of-distribution data. Both methods model *distributional uncertainty*, i.e. a distributional mismatch between the test and training data distributions. Although this type of uncertainty is different from the aleatoric one analysed in this section, we include them here because of their usage of Dirichlet distributions for modelling the data. In Malinin and Gales [2018], they explicitly separate the three types of uncertainty:

$$p(y^*|\mathbf{x}^*,X,Y) = \int_W \int_\mu \underbrace{p(y^*|\mu)}_{data} \underbrace{p(\mu|\mathbf{W},\mathbf{x}^*)}_{distributional} \underbrace{p(\mathbf{W}|X,Y)}_{model} d\mu\mathbf{W} \tag{2.20}$$

The point estimate for the data uncertainty comes determined by the distributional uncertainty. To model the new term $p(\mu|\mathbf{W},\mathbf{x}^*)$ they use a Dirichlet distribution $Dir(\mu|\alpha)$, where $\alpha = f(\mathbf{x}^*;\mathbf{W}^*)$. For training the posterior, they explicitly train in a multi-task fashion to minimize the KL divergence between the model and a sharp Dirichlet distribution focused on the appropriate class for in-distribution data, and between the model

and a flat Dirichlet distribution for OOD:

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}_{\mathrm{p_{in}}(\mathbf{x})}[KL[\mathrm{Dir}(\mu|\hat{\alpha})\|\mathrm{p}(\mu|\mathbf{x};\mathbf{W})]] + \mathbb{E}_{\mathrm{p_{out}}(\mathbf{x})}[KL[\mathrm{Dir}(\mu|\tilde{\alpha})\|\mathrm{p}(\mu|\mathbf{x};\mathbf{W})]] \quad (2.21)$$

Where we can find a decomposition of the loss composed by a term for the in distribution and OOD data samples. The multi task part comes from the need of having in and out of distribution points, that they simulate by generating points on the boundary of the in-domain region using a generative model or using a different, real dataset as a set of samples from the out-of-domain distribution.

Following a similar approach, in Chen et al. [2018] they propose to parameterize a posterior model $p_{\mathbf{W}}(Z|X)$ and optimize its parameters to approach such true posterior $p_{\mathbf{W}}(Z|X)$ given a pairwise input (X,Y) by minimizing their KL-divergence:

$$D_{\mathrm{KL}}\left(P_{\mathbf{W}}(Z|X)\|P(Z|Y)\right) \quad (2.22)$$

For minimising the divergence, they use VI using a Dirichlet as the variational posterior. In this case, though, they do not explicitly separate the OOD part. To demonstrate the OOD detection, they train the network using a dataset and then analyse the uncertainties found when applying the model to a different one.

To wrap up, table 2.1 holds a summary of the different methods analysed including the type of uncertainty studied on each of them.

Table 2.1: Summary of uncertainty methods reviewed

Method	Reference	Type of uncertainty	Section
Bayesian Neural Networks using Laplace approximation	MacKay [1992c,b]	Epistemic	2.3.1.1
Markov chain Monte Carlo approximations	Neal [2012]	Epistemic Aleatoric	2.3.1.2
Stochastic gradient Hamiltonian Monte Carlo	Chen et al. [2014]	Epistemic Aleatoric	2.3.1.2
Minimum Description Length Principle	Hinton and Van Camp [1993]	Epistemic Aleatoric	2.3.1.3
Ensemble Learning	Barber and Bishop [1998]	Epistemic Aleatoric	2.3.1.3
Variational Inference	Graves [2011]	Epistemic	2.3.1.3
Stochastic Variational Inference	Hoffman et al. [2013]	Epistemic	2.3.1.3
Bayes by Backprop	Blundell et al. [2015]	Epistemic	2.3.1.3
Stochastic gradient variational Bayes for gamma approximating distributions	Knowles [2015]	Epistemic	2.3.1.3
The generalized reparameterization gradient	Ruiz et al. [2016]	Epistemic	2.3.1.3
Implicit reparameterization gradients	Figurnov et al. [2018]	Epistemic	2.3.1.3
Pathwise derivatives beyond the reparameterization trick	Jankowiak and Obermeyer [2018]	Epistemic	2.3.1.3
Flipout	Wen et al. [2018]	Epistemic Aleatoric	2.3.1.3
Expectation propagation	Hernandez-Lobato and Adams [2015]; Li et al. [2015]	Epistemic	2.3.1.4
MC Dropout	Gal and Ghahramani [2015]	Epistemic	2.3.1.5
Network Ensemble	Lakshminarayanan et al. [2017]	Epistemic	2.3.1.6
Bootstrapped DQN	Osband et al. [2016]	Epistemic	2.3.1.6
Aleatoric MC Dropout	Gal [2016]; Kendall and Gal [2017]	Epistemic Aleatoric	2.3.2.1
Neural Network Regression with Beta, Dirichlet, and Dirichlet-Multinomial Outputs	Sadowski and Baldi [2018]	Aleatoric	2.3.2.2
Lightweight probabilistic deep networks (LPN)	Gast and Roth [2018]	Epistemic Aleatoric	2.3.2.2
A general framework for uncertainty estimation in deep learning	Loquercio et al. [2020]	Epistemic Aleatoric	2.3.2.2
Understanding DNN through input uncertainties	Thiagarajan et al. [2019]	Aleatoric	2.3.2.2
Predictive Uncertainty Estimation via Prior Networks	Malinin and Gales [2018]	Epistemic Aleatoric Distributional	2.3.2.2
Variational Dirichlet Framework	Chen et al. [2018]	Aleatoric Distributional	2.3.2.2

2.3.3 Measuring Uncertainty

In the previous section, we have reviewed previous works that mean to model uncertainty in DL models. In the present section, we review how to measure this uncertainty, i.e. how to obtain a numerical score out of the probability distributions that describe the output of the predictive system.

In classification systems, the subject of this thesis, specifically in probabilistic classifiers, the output of the model already has the shape of a probability distribution over the predicted classes, as resulting from applying a softmax operation over the output logits. Therefore, a first approach for obtaining a measure of the uncertainty is by observing the output distribution itself. At this point, we want to make two remarks. On the one hand, recall that even when the probability associated with the predictions is high, it can correspond to a confident but erroneous prediction. On the other hand, the output of a DL might suffer from miscalibration, as detailed in Guo et al. [2017].

However, assuming a calibrated and right prediction, these are the metrics derived directly from observing the predictions of the original model, namely:

- Softmax response, as the maximal neuronal response of the softmax layer.
- Predictive entropy: the difference between all predictions, as defined by information theory, Dagan and Engelson [1995], understood as the entropy of the softmax probability distribution.

$$\frac{-\sum_y P_{\mathbf{W}}(\mathbf{y}_1^*|\mathbf{x}^*)\log_2 P_{\mathbf{W}}(\mathbf{y}_1^*|\mathbf{x}^*))}{\log_2(n)} \tag{2.23}$$

- Least confidence: the difference between the most confident prediction and 100% confidence, as defined in Culotta and McCallum [2005].

$$\frac{n\,(1-P_{\mathbf{W}}\,(\mathbf{y}_1^*|\mathbf{x}^*))}{n-1} \tag{2.24}$$

- Margin of confidence: difference between the top two most confident predictions.

$$1-(P_{\mathbf{W}}\,(\mathbf{y}_1^*|\mathbf{x}^*)-P_{\mathbf{W}}\,(\mathbf{y}_2^*|\mathbf{x}^*)) \tag{2.25}$$

- Ratio of confidence: ratio between the top two most confident predictions , as defined in Scheffer et al. [2001], understood as the margin and ratio respectively between the first and second more confident output units.

$$\frac{P_{\theta}\,(\mathbf{y}_2^*|\mathbf{x})}{P_{\theta}\,(\mathbf{y}_1^*|\mathbf{x})} \tag{2.26}$$

All these metrics obtain an uncertainty score out of point estimates. Nevertheless, as explained in the previous section, in order to accurately model uncertainty, instead of point estimates, a proper output probability is mandatory. In literature, Gal [2016], we find different ways of obtaining a numerical uncertainty score out of a probabilistic output layer:

Variation ratios measures the variability of the predictions obtained from sampling Freeman [1965] by computing the fraction of samples with the correct output. This heuristic is a measure of the dispersion of the predictions around its mode. To that respect, for a given data point x we can sample M output vectors from the output distribution $y_x^t, t = 1 \dots M$, and compute the variation ratios as follows,

$$VR = 1 - \frac{f_{c^*}}{M} \tag{2.27}$$

where $f_{c^*} = \sum_t 1[\mathbf{y}_x^t = c^*]$, and c^* corresponds to the sampled majority class,

$$c^* = \arg \max_{c=1,\dots,C} \sum_{t=1}^{M} 1[\mathbf{y}_x^t = c]. \tag{2.28}$$

Predictive entropy considers the average amount of information contained in the predictive distribution, Shannon [1948]. Results with low entropy values correspond to confident predictions, whereas high entropy leads to significant uncertainty. This score takes into account the variability of the predicted value by sampling from the wrapper inferred distribution, the sampled predictive entropy, defined as

$$\mathbb{H} = -\sum_c \mathbb{E}[\mathbf{y}^*]_c \log \mathbb{E}[\mathbf{y}^*]_c. \tag{2.29}$$

Mutual information considers the relationship between two RV, X, Y and measures the mutual dependence between the two. Again, it relates to the Information theory as it can be viewed as the amount of information obtained about one RV by observing the other RV. Mutual information, MI, determines how different the joint distribution of the pair (X,Y) is to the product of the marginal distributions of$X$ and Y:

$$I(X;Y) = D_{\mathrm{KL}}\left(P_{(X,Y)} \| P_X \otimes P_Y\right) \tag{2.30}$$

Where we use KL divergence to measure this difference, which when considering continuous variables translates to:

$$\mathrm{I}(X;Y) = \int_{\mathcal{Y}} \int_{\mathcal{X}} p_{(X,Y)}(x,y) \log \left(\frac{p_{(X,Y)}(x,y)}{p_X(x) p_Y(y)} \right) dxdy \tag{2.31}$$

Or seeing it from the Information theory perspective:

$$\mathrm{I}(X;Y) = \mathrm{H}(Y) - \mathrm{H}(Y|X) \tag{2.32}$$

When measuring the uncertainty of NN models, the two variables to consider are the prediction y^* and the posterior over the model parameters $\mathbf{W}$. Usually, this metric relates more to epistemic uncertainty as it compares the posterior of $\mathbf{W}$ with Y, as approximated in Gal [2016]:

$$\mathbb{I}[Y, \mathbf{W}|X, \mathcal{D}_{\text{train}}] \approx \underset{T\to\infty}{\longrightarrow} \mathbb{H}[Y|X, \mathcal{D}_{\text{train}}] - \mathbb{E}_{q^*_\theta(\mathbf{W})}[\mathbb{H}[y|X, \mathbf{W}]] \tag{2.33}$$

by sampling T models for the posterior $q^*_\theta(\mathbf{W})$. In Houlsby et al. [2011], they state that test points $\mathbf{x}^*$ that maximise the mutual information are points on which the model is uncertain on average, as the model parameters $\mathbf{W}$ erroneously produce predictions with high confidence.

Additional methods for measuring uncertainty are present also in the works reviewed in the previous section, Malinin and Gales [2018]; Chen et al. [2018], that make use of Dirichlet distributions.

In Chen et al. [2018], they propose to use the entropy of the resulting Dirichlet output distribution:

$$E(\alpha) = -C(\alpha) = -\int_z z \text{Dir}(z|\alpha)dz = \log B(\alpha) + (\alpha_0 - K)\,\psi(\alpha_0) - \sum_i^k (\alpha_i - 1)\,\psi(\alpha_i) \tag{2.34}$$

The final uncertainty score corresponds to the negative of this entropy. In this work, they also analyze the resulting α parameters. They observe how these parameters tend to adopt big values $\alpha >> 1.0$, compromising the performance of the method. To prevent this problems, they suggest a smoothing factor $\hat{\alpha} = \log(\alpha + 1)$, while other smoothing functions can be applied.

Finally, in Malinin and Gales [2018], they propose different metrics of uncertainty. Recall that in that article, they talk about three different types of uncertainty: model, data and distributional. Depending on the type of marginalisation applied to equation 2.20, they can calculate the different types of uncertainty. For obtaining the total uncertainty, they propose the use of *max probability*, $\mathcal{P} = \max_c \text{P}(\omega_c|x^*; \mathcal{D})$, and the predictive entropy. For the rest of uncertainties, they propose to use the mutual information between the total uncertainty and each type of uncertainty.

2.4 Desirable properties of uncertainty

In previous sections, we have seen the importance of uncertainty and its role in classification systems. We have reviewed different methods for modelling the different

types of uncertainty associated with classifiers, and different ways to measure it. In the present section, we outline some properties that are desirable in any measure of uncertainty.

2.4.1 Calibration

As mentioned in section 2.3.3, the more straightforward way to measure the uncertainty of classification systems is to observe the probability associated with the predicted class. Take, for instance, a binary classifier—a prediction of 0.9 would mean that the classifier is confident on assigning a positive prediction. In contrast, a prediction of 0.51 would mean that the uncertainty of the prediction is high.

Apart from the fact that these predictions might be confident but erroneous, another aspect that is essential for considering these probabilities is the fact that they need to be calibrated. Usually, the output of a classifier comes from the application of a sigmoid or a softmax function. These functions transform the output into values from 0 to 1. In the case of softmax, by normalizing the outputs, results describe sets of C values, one for each class, that adds to 1, which resembles a probability distribution.

To consider a classification system as calibrated its output must match the expected distribution of probabilities for each class. The problem is that most of the time this is not the case, especially with modern DL architectures. As reported in Gast and Roth [2018], the more complex the architecture, the less calibrated the output is. The miscalibration problem also extends to other types of predictive systems, as analyzed in Niculescu-Mizil and Caruana [2005], like Support Vector Machines, Random Forests or boosted trees.

2.4.1.1 How to measure calibration

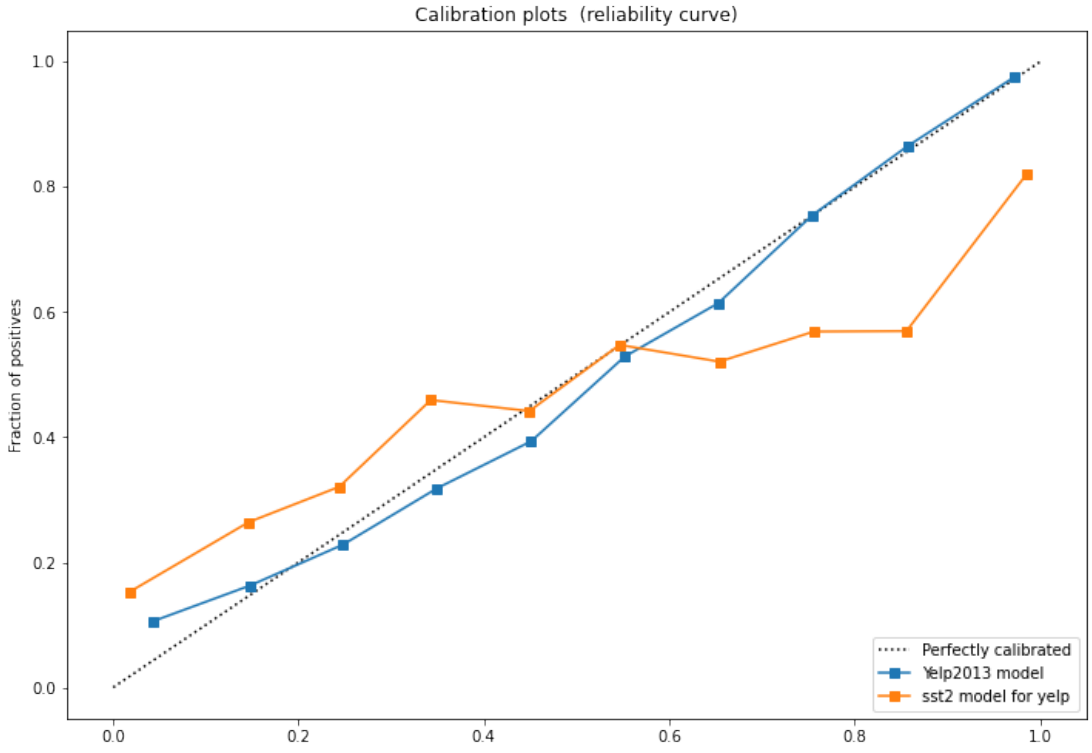
The workhorse, when analyzing calibration in a classification system, is reliability diagrams, also known as calibration curves. For a further analysis of reliability diagrams, see Bröcker and Smith [2007], but the main idea is that a proper output probability distribution should adequately reflect the actual likelihood of the predicted data. For example, given 100 predictions, each with a probability of correctness of 0.8, we expect that 80 should be correctly classified. So a definition for the perfect calibration should be:

$$P(y^* = y|\hat{P} = p) = p, \quad \forall p \in [0,1] \tag{2.35}$$

where $\hat{P}$ is the confidence of the prediction, and p is the actual likelihood of the predicted data.

To build a reliability diagram, we group predictions into M interval bins (each of size 1/M) and calculate the average accuracy and confidence of each bin. Next, we plot the expected sample accuracy as a function of confidence. Figure 2.1 shows an example of reliability diagrams. In this case, we observe how the ideal calibration would correspond to the diagonal, where the accuracy is equal to the confidence. In contrast, we observe how the blue line corresponds to a better-calibrated model than the orange one.

Figure 2.1: Example of reliability diagrams



Even though the reliability diagram is a handy tool for diagnosing calibration issues, sometimes we require a numerical score. One of the most common metrics is Expected Calibration Error (ECE). It is very related to reliability diagrams, as it measures the difference between the average accuracy and confidence, as calculated in the diagrams:

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\operatorname{acc}(B_m) - \operatorname{conf}(B_m)| \tag{2.36}$$

Where M is the number of bins m, n is the number of samples, and acc and confi-

dence correspond to the accuracy and confidence of the predictions respectively. A variation of this measure is Maximum Calibration Error (MCE), which computes the maximum instead of the average. Furthermore, sometimes, measuring the Negative log-likelihood, NLL, of the model can be used.

2.4.1.2 Calibration methods

The methods described in Gast and Roth [2018] help on tackling this calibration issues. These methods required a hold-out data set drawn from the same distribution to train the calibrators. These are the main ones:

- Isotonic regression. The most common non-parametric calibration method. It learns a constant function f to transform uncalibrated outputs. The function f is trained to reduce the RMSE between the label and the prediction:

$$\sum_{i=1}^{n} \left(f\left(y_i^*\right) - y_i\right)^2 \tag{2.37}$$

- Platt Scaling. A parametric approach to calibration, where the original predictions, y^*, of a classifier are used as features for a logistic regression model, $\hat{q}$. This model is trained on the validation to learn scalar parameters $a, b \in \mathbb{R}$ such as $\hat{q}_i = \sigma\left(az_i + b\right)$ emits a calibrated probability.

- Temperature Scaling. It is an extension of Platt scaling that uses a single scalar parameter $T > 0$ to modify the temperature of the softmax function applied to the output logits. T is optimized for NLL on the validation set, modifying the output probabilities but preserving the predicted class.

To sum up, calibration is an essential property of classification systems, especially when evaluating the output probabilities for measuring the confidence of the predictions. As reported in Gast and Roth [2018]; Niculescu-Mizil and Caruana [2005], methods that rely on measuring probabilities, like logistic regression, seem to be more calibrated than others. Thus, when developing methods that transform the outputs of classifiers as RV, we should expect that the outputs of such systems will be better calibrated than the original black boxes.

2.4.2 Robustness

Another aspect that any uncertainty modelling technique must bear in mind is the robustness of its metric. By robustness of an uncertainty metric, we understand its capa-

bility of adapting to new data distribution on prediction time. These data distributions might differ from the original ones used for training the models, but the uncertainty must keep efficiently detecting uncertainty of the new data points.

To this extent, we find the work presented in Snoek et al. [2019] very relevant to the robustness of uncertainty methods. In this work, they analyse different uncertainty metrics, and they subject them to different types of distortion of the original training inputs, including a study of their behaviour on OOD data.

The methods analysed in the study include a vanilla implementation based on the Maximum softmax probability, Hendrycks and Gimpel [2016], the same metric but after applying the Temperature scaling calibration method introduced in the previous section, MC Dropout, Gal and Ghahramani [2015], Stochastic Variational Inference, Blundell et al. [2015], ensemble methods, Lakshminarayanan et al. [2017] and a last-layer-only variation of both dropout ad SVI, Riquelme et al. [2018].

For measuring the quality of the uncertainty modelled, they use the metrics detailed in section 2.4.1, namely ECE and NLL, plus Brier Score, a score that measures the accuracy of predicted probabilities, as the squared error of a predicted probability vector, $\mathbf{y}^*$, and the one-hot encoded true response $\mathbf{y}$.

They include experiments using datasets from different domains, including image and text classification, tabular data or even genomics. For each dataset, they include the original data plus two additional datasets: one with gradual levels of distortion, and another one with a completely different distribution to test the OOD case. In the case of OOD, as labels do not apply, they plot a distribution of the predictive entropy to observe the detection capability of the model for OOD points.

They provide the reader with some insights that are relevant when talking about the robustness of uncertainty measures. First is that along with accuracy, the quality of uncertainty consistently degrades with increasing dataset shift regardless of method. Methods seem to work well with test data, like Temperature scaling, do not adapt to shifts in the data distributions. For the epistemic models, they outline Ensemble as the best performing, while indicating that SVI, even though they seem to work well on small datasets, are hard to apply to datasets that require complex architectures.

All in all, from the analysis we can derive that modelling uncertainty for a given problem or dataset does not mean that this model adapts well to changes on the data once we deploy the model in a real environment. At this point, we want to remark a couple of aspects relative to the present work.

The first thing to consider is that, except for the Vanilla and Temperature scaling,

the included methods were all tackling epistemic uncertainty. They measure how the model adapts to the data, not the uncertainty of the data itself. Training the original models with data points from the additional distorted datasets would reduce this uncertainty. In the present work, we do not measure the epistemic uncertainty as the model is a fixed black box, but focused on measuring the uncertainty inherent in this data. After rotating an image that represents a horse, the image will still represent a horse. If the picture is appropriately representing a horse, the associated aleatoric uncertainty will be low. The problem is when the picture of this horse is labelled as a dog. Then the model will be very confident predicting it like a horse, so epistemic uncertainty will be low, whereas the aleatoric uncertainty will be high.

Besides, in their analysis, they explicitly discard methods that train specifically for OOD detection. As mentioned above, they analyse the uncertainty of OOD points by observing the resulting predictive entropy. Looking at the results reported for the text models, the only ones reported for OOD, we observe how, except for the last-layer-only versions, all the methods used, including Vanilla, already report OOD points with high uncertainty. We reckon that as outlined in Malinin and Gales [2018], epistemic and distributional uncertainties are separate concepts, and consequently, it must be explicitly tackled.

Nonetheless, the work presented in Snoek et al. [2019] delivers an appealing framework for evaluating uncertainty in scenarios that might mimic what one can expect when deploying classification models to the wild, where data distributions may shift or new types of examples might appear.

2.4.3 Interpretability

Last but not least, another aspect that is crucial when estimating the uncertainty of a classification system is its interpretability. Habitually, uncertainty is considered as a source of interpretability of the predictions of a classifier: uncertain predictions can warn the practitioner about problematic inputs that might require the intervention of a human expert or complementary sources of information, like sensors in a self-driving car.

In this section, though, we would like to focus on the interpretability of the uncertainty itself. In section 2.3.3, we analysed different ways of obtaining a numerical score for measuring uncertainty. Taking the simpler of those methods, the confidence or softmax response of the prediction, we observe that it can be treated as a proba-

bility(omitting calibration issues). In this case, the interpretation of the uncertainty is quite straightforward, as humans understand well probabilities, Cosmides and Tooby [1996].

The problem comes with more complex metrics like predictive entropy. Such measures might require further explanations for making them understandable and useful for practitioners. For example, predictive entropy scores will depend on the number of classes predicted, being 0.69 the maximum entropy for a binary classifier and 2.30 for ten classes one.

Many of these complex metrics derive from the output probability layers modelled, as detailed in section 2.3.2. What they want to capture is the variance of the distributions that describe the output, based on the given inputs. Depending on the distribution used, variance might be the σ parameter of a Gaussian RV. In other cases, it can be a β parameter resulting from the decomposition of the α concentration parameters of a Dirichlet distribution. In any case, the model trained for estimating the uncertainty, u, takes input and outputs a number for that input, so $u(\mathbf{x}_i) = \sigma$, where sigma corresponds to the uncertainty parameter. In any case, we can consider it as a regression problem. As such, we can apply explainability tools to search for explanations on which parts of the input contribute more to this uncertainty. Tools like LIME, Ribeiro et al. [2016b], os Shap values, Lundberg and Lee [2017], among others, can hep on attributing uncertainty to words in a text that increase uncertainty in a sentiment analysis tool. Tools like the What-if-tool, Wexler et al. [2019], can help on identifying attributes in a tabular dataset that can be changed to diminish this uncertainty.

To the best of our knowledge, there is no previous work on the interpretability of uncertainty, and we firmly believe that it could benefit the application of uncertainty in real problems. Having ways to identify which elements of our data contribute more to uncertain predictions can enable actionable mechanisms that allow increasing the quality of the predictions by intervening in the data. An example of this could be an image of a horse behind a fence. This kind of patterns might hinder the results of a pre-trained image classifier, increasing the associated uncertainty. What if we had a way to detect this issue? In this case, we could apply a pre-processing to our target dataset images, removing fences(see Jonna et al. [2015]) and thus increasing the quality of the resulting predictions.

2.5 Applications of uncertainty

In the present section, we analyse potential applications of uncertainty. Beyond the interpretability of the predictions introduced in the previous section, in literature, we find many examples of how to use the different types of uncertainty to help with improving the quality of ML models. Using epistemic uncertainty, we will see how one can estimate to which extent a model adapts for a given problem or how to select new examples that foster the training speed. We will also observe how by using the aleatoric uncertainty, we can discard those examples that are more uncertain, increasing the quality of the preserved predictions or permitting the intervention of experts to decide on those cases.

2.5.1 Model Selection

The first application of uncertainty is model comparison or selection. When designing DL models, the choices of architectures, parameters and regularizers can be cumbersome. Working with held-out data or cross-validation to optimise these hyperparameters imply sacrificing some of the training data available for that purpose. This sacrifice has brought researchers to try to include the optimisation of these hyperparameters in the model training. To this aim, the use of epistemic uncertainty allows to compare models effectively and include the implicit optimisation of hyperparameters and regularizers.

One of the first to introduce this use of model uncertainty was David Mackay in MacKay [1992a,b]. By adding the evidence step to the modelling, he modelled the probability of the data given the model parameters and regularizers, $p(D|\mathbf{W}, \mathcal{R})$. By estimating this posterior to find its MAP, he was able to find the most optimal combination of parameters and regularizers.

In Mackay's work, the estimation of this model uncertainty was carried out in the additional step in the inference process called model selection. Later works like VI, EM or ensembles, propose to include the uncertainty in the training process. By considering the weights as RV, they obtain not only one valid model but a collection of infinite(or finite in the case on ensembles) models that maximise the posterior of those model given the training data.

In Gal [2016]; Kendall and Gal [2017], they propose to use dropout for sampling models $\mathbf{W}$, also tackling the optimisation of the regularizers by using the aleatoric uncertainty in what they call loss attenuation.

After modelling the model uncertainty by choosing one of the approaches presented in section 2.3.1, it would be possible to sample parameters $\mathbf{W}^*$ from the obtained distributions. This sampling will allow estimating the expectation of the output, $\mathbb{E}_{p(\omega|\mathcal{D}_{\text{train}})}[Y|X,\omega]$. With this expectation, it is possible then to estimate the mutual information between the prediction Y and the posterior over the model parameters $\mathbf{W}$. As illustrated in Gal [2016], mutual information captures the model's confidence in its output, and it is an excellent way for comparing the performance of different models for a given problem.

In any case, model selection based on epistemic uncertainty lies out of the goal of the present work, as when working with black-boxes, we can not alter the model to include the required variability for measuring it. In this work, we propose to analyse the aleatoric uncertainty of pre-trained models given the target dataset. By comparing the resulting uncertainty, we can measure to which extent those model fit our problem and decide which of them we apply to solve our requirements.

2.5.2 Active Learning

Another recurrent application of uncertainty present in the literature of ML is Active Learning (AL), Cohn et al. [1996]. The goal of AL is to boost the training process by selecting meaningful and representative examples that let the model learn faster. This task is especially relevant in constrained scenarios where obtaining training data requires the intervention of human experts to label the samples, implying extra-efforts in time and costs. In such scenarios, having a way to drive the selection of the new samples to label that fosters the training speed, is a very desirable feature.

There exist different ways to tackle AL, as surveyed in Settles [2009]. Among them, the simplest one is uncertainty sampling, Lewis and Gale [1994]. By observing the output confidence or predictive entropy of the so-far-trained model when applied to unlabelled points, one can choose those that are more uncertain, expecting that those will be the more challenging ones for the model, thus fostering the training process.

The problem with this approach is that sampling from the more uncertainty areas of the unlabelled data distributions, most of the times are equivalent to sample from the decision boundary of the classifier. As David Mackay suggested in MacKay [1992b]: "The strategy of sampling on decision boundaries is motivated by the argument that we are unlikely to gain information by sampling in a region where we are already confident of the correct classification. But similarly, if we have already sampled a great deal on

one particular boundary then we do not gain useful information by repeatedly sampling there either, because the location of the boundary has been established!".

In this work, Mackay proposes another way to use uncertainty for AL. The main idea is to divide the unlabelled dataset in what he called regions of interest and obtain labels for points that maximize the mean marginal entropy. The intuition is that those points with a higher variance according to the entropy of the predictions obtained drawing samples from the posterior over the models, will translate to more representative points and better AL.

Recently, this concept of regions of interests has been extended to the context of Thompson sampling (TS) and multi-armed bandits. In Russo et al. [2018], they propose to use ensembles as the arms and include TS in the selection of new training inputs. In Bouneffouf et al. [2014], they propose to create clusters of the unlabelled points and consider each of these clusters as the arms. Then applying TS, they chose examples based on a reward function that looks at the cosine distance of a vector composed by the points before and after adding the selected input.

Other approaches to AL are those based on variance reduction and Fisher information Ratio, like Zhang and Oles [2000]. In this case, they are pool-based method that look at the ratio $X^*_{FIR} = \underset{X}{\operatorname{argmin}} \operatorname{tr}\left(I_X(\theta)^{-1} I_{\mathcal{U}}(\theta)\right)$, between the selected point and the rest of the unlabelled examples. Fisher information ratio is a measure of epistemic uncertainty that in addition to the uncertainty of the model also measures which model parameters are most responsible for this uncertainty.

In general, based on the uncertainty of the model, different strategies select new samples to label based on different criteria: reducing the expected error or maximizing the variance of the model or how much it changes after adding the sample to the training.

Regarding AL, the use of aleatoric uncertainty is not indicated, as the same conclusions extracted from the selection of points in the decision boundaries would apply here. Aleatoric uncertainty measures the noise inherent in the data, so selecting points that are noisy by their own would not help on improving the variance of the model or the expected error, as this type of uncertainty will not reduce with more training, unlike it occurs with epistemic.

2.5.3 Classification with rejection

Another application where the role of uncertainty is key is classification systems with rejection. The process of abstaining on producing an answer or discarding a prediction when the system is not confident enough can be found under different names in the machine learning literature: Rejection Methods, Selective Classification, Abstention, or Three-Way Classification. The most common approach sets a threshold based on a given metric and discards predictions accordingly. Following the ideas proposed in the seminal work of Chow [1970], where the authors set a threshold for rejection with which to minimise the classification risk, other works like Cordella et al. [1995]; De Stefano et al. [2000b]; El-Yaniv and Wiener [2010]; Geifman and El-Yaniv [2017b] consider different cost-based rejection models based on the output responses of deep learning models.

Alternatively, other works Bartlett and Wegkamp [2008]; Cortes et al. [2016]; De Stefano et al. [2000a] include a term for the rejector in the loss that is learned together with the classifier. Following a similar approach, in Thulasidasan et al. [2019], the authors propose to omit noisy labels in order to improve the training process. These works have in common the approach of considering a fixed-cost for the abstention, in which the classifier incurs a fixed cost every time the abstain option is invoked. On the contrary, in Shekhar et al. [2019], authors set a fixed fraction of input samples, δ, in which the learner is allowed to abstain without incurring any costs. They propose a plug-in classifier that employs unlabelled samples to decide the region of abstention and derive an upper-bound on the excess risk. Another strategy, followed by authors of Geifman and El-Yaniv [2019], is to train the model together with a particular loss function for rejection. In this case, though, they fit a model specifically for each degree of coverture chosen a priori.

The main limitation of the methods described so far is the fact that they jointly learn the classification model together with the rejection function. Unfortunately, in a black-box scenario, re-training the model is not an option.

Finally, the evaluation of the performance of classification with rejection models usually use standard metrics such as accuracy or F1-score for obtaining accuracy-rejection curves (ARC) Nadeem et al. [2009] or 3D receiver operating characteristic Landgrebe et al. [2006]. Other works focus on the rejection of multi-label classification systems Pillai et al. [2013]. These approaches present limitations as they are not able to determine the optimal rejection rate by comparing the performance of the clas-

sifiers. Beyond the ROC space, some authors Hanczar [2019] have analysed different representation spaces to build a rejection system, mainly the cost-reject (CR) and the error-reject (ER) space.

Aligned with this error-rejection approach, we find Condessa et al. [2015], where authors propose three different performance measures for evaluating the best rejection point that overcomes the previous restrictions. In order to evaluate the rejection metric, they split the dataset using two criteria: whether the method **R**ejects the data point or **N**ot; and whether the point is **A**ccurately classified, or **M**issclassified named as R, N, A or M respectively. By using these values, three quality metrics can be derived (illustrated in Fig.2.2):

- **Non-rejected Accuracy** measures the ability of the classifier to classify non-rejected samples accurately. It is computed as follows,

$$NRA = \frac{|A \bigcap N|}{|N|}$$

- **Classification Quality** measures the ability of the classifier with rejection to classify non-rejected samples accurately and to reject misclassified samples. It is computed as follows,

$$CQ = \frac{|A \bigcap N| + |M \bigcap R|}{|N| + |R|}$$

- **Rejection Quality** measures the ability to concentrate all misclassified samples onto the set of rejected samples. It is computed as follows,

$$RQ = \frac{|M \bigcap R|\,|A|}{|A \bigcap R|\,|M|}$$

An optimal rejection point will show a trade-off between the three metrics, being able to divide misclassified predictions from the right ones and preserve only those points that provide useful information. The higher the value displayed, the better that metric performs for rejection.

In a rejection setting, the model is enhanced with a new class, the *rejection* class. Thus, the final prediction becomes

$$y^* = \begin{cases} f(\mathbf{x}), \text{ if } \phi(\mathbf{x}) < \tau \\ reject, \text{ if } \phi(\mathbf{x}) \geq \tau \end{cases}$$

Figure 2.2: Rejection performance metrics as proposed in Condessa et al. [2015]

Here, $f(\mathbf{x})$ is the classifier without rejection, the function $\phi(\mathbf{x})$ is a function on the input that evaluates the confidence of the prediction model, and τ is a threshold value that indicates the rejection point. Those predictions with lower confidence than the given threshold get rejected, whereas those more confident are predicted using the original classifier.

In the context of the present (observing the black-box constrain), the $\phi(\mathbf{x})$ function can not be learned together with the classifier when it comes to pre-trained black-boxes. In this case, we must obtain the confidence score from the input and the corresponding prediction. In probabilistic classifiers, risk can derive from the observation of the output probabilities employing different metrics, like Least Confidence Culotta and McCallum [2005], Margin of Confidence Scheffer et al. [2001] and Variation Ratios and Predictive Entropy, as proposed in Gal [2016].

In the present work, we propose to use the aleatoric uncertainty as the $\phi(\mathbf{x})$ function to model the uncertainty of black-box classification systems. To validate that the proposed method better captures the uncertainty in such models, we apply the performance measures proposed in Condessa et al. [2015].

An option to automatize the selection of the rejection point could be to use a method like Geifman and El-Yaniv [2017a], where they use a binary search to optimise it.

Chapter 3

Uncertainty Estimation for Black-box Classifiers

Contents

In the previous sections, we have reviewed how different approaches modelled the uncertainty in classification systems. The present chapter aims to address the primary goal of this thesis which is how to model this uncertainty for black-box classifiers. In previous works, we saw how to model uncertainty they apply different changes in the training process. It included different ways to model the weights as RVs, or changes in the output activation units to include some variability, turning them into probability distributions.

In the case of black-box pre-trained classification systems, like those offered by third-party APIs, there is no chance to modify the model or to retrain it using the original data. These constraints prevent the application of previously described methods and demands for the development of new methods that work directly with the outcomes of those black-boxes.

In this work, we focus on discriminative models, where the goal is to capture the distribution that generated the outputs based on the data. We assume that from the data in X, Y and $\mathbf{x}^*$, we want to know the distribution that generated y^*. However, in the case of using a black-box classifier, the model is given, and the parameters W^* set:

$$p(y^*|\mathbf{x}^*, X, Y) = \int_{\mathbf{W}^*} p(y^*|\mathbf{W}^*, \mathbf{x}^*) p(\mathbf{W}^*|X, Y)\mathbf{W} \tag{3.1}$$

, where the probability $p(\mathbf{W}^*|X, Y)$ is equal to 1.

Thus, the fact that black-box fixes the model prevents us from estimating the epistemic uncertainty, as there is no chance to add variability to the parameters $\mathbf{W}^*$, forcing us to focus on the aleatoric one: $p(y^*|\mathbf{W}^*, \mathbf{x}^*)$. In the rest of the chapter, we describe different approaches that we analysed to estimate the aleatoric uncertainty, following the same timeline that took us to the final method proposed. These approaches consist of different versions of a deep learning wrapper, based on a probability distribution, that given a black-box classification system allows the measurement of aleatoric uncertainty. Moreover, at the end of the chapter, we illustrate the different methods analysed through the thesis employing a toy problem to compare them and to show the intuition behind the proposed method.

3.1 Gaussian approach

The first version of the wrapper for modelling aleatoric uncertainty builds on top of the idea of considering $p(y^*|\mathbf{W}^*, \mathbf{x}^*)$ as a Gaussian RV. Because we are assuming a black-box model, we are constrained to the estimation of heteroscedastic uncertainty. Thus, we consider that we have a fixed deterministic black-box model $f^{\mathbf{W}}(\mathbf{x})$ with undisclosed non-trainable parameters $\mathbf{W}$. Our goal is to compute the variability of the term $p(y^*|\mathbf{W}, \mathbf{x}^*)$ in Equation 3.1. For the sake of simplicity, let us assume that this conditional distribution follows a normal distribution, i.e. $y^*|\mathbf{W}, \mathbf{x}^* \sim \mathcal{N}(f^{\mathbf{W}}(\mathbf{x}^*), \sigma^2(\mathbf{x}^*))$, where $f^{\mathbf{W}}(\mathbf{x}^*)$ is the black-box model evaluated at the data point $\mathbf{x}^*$, and $\sigma^2(\mathbf{x}^*)$ is a function of the input data that models the variance for that data point. In regression tasks, applying this approximation to the log-likelihood adds a term to the loss

that depends on $\sigma(\mathbf{x})$ Kendall and Gal [2017]. However, in classification tasks, this approximation is not as straightforward as in regression.

We consider the scenario where $f^{\mathbf{W}}(\mathbf{x}^*)$ is implemented by a deep neural network. In general, the computation of aleatoric uncertainty for fully trainable networks considers the introduction of a stochastic layer to represent the output logits space. Here we consider that the output logits, u, of the classification model follow a Normal distribution, with a diagonal variance term, described as follows,

$$\mathbf{u} \sim \mathcal{N}(f^{\mathbf{W}}(\mathbf{x}^*), diag(\sigma^2(\mathbf{x}^*))) \tag{3.2}$$

$$\mathbf{p} = softmax(\mathbf{u}) \tag{3.3}$$

$$y \sim Categorical(\mathbf{p}) \tag{3.4}$$

by reparameterizing the logits, u, we obtain,

$$\mathbf{u} = f^{\mathbf{W}}(\mathbf{x}^*) + \sqrt{diag(\sigma^2(\mathbf{x}^*))} \cdot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0,1) \tag{3.5}$$

In its simple approach, working with this stochastic layer requires of its sampling. In general, we would need to compute the expected value by applying Monte Carlo sampling, obtaining,

$$\mathbb{E}[\mathbf{p}] = \frac{1}{M} \sum_{m=1}^{M} softmax(\mathbf{u_m}) \tag{3.6}$$

When applied to a cross-entropy loss allows us to obtain the loss we will use for the wrapper, i.e.

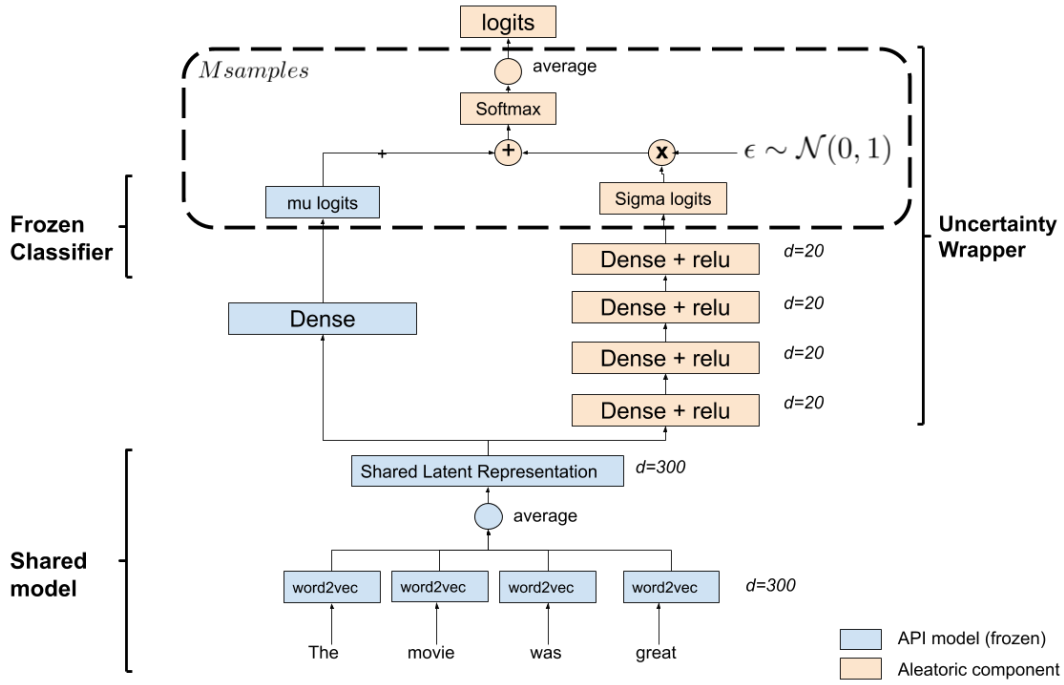
$$\mathcal{L}(\mathbf{W}) = \frac{1}{N} \sum_{i=1}^{N} -\frac{1}{C} \sum_{c=1}^{C} \mathbf{y}_{i,c} \log(p_{i,c}) = \tag{3.7}$$

$$\frac{1}{N} \sum_{i=1}^{N} -\frac{1}{C} \sum_{c=1}^{C} \mathbf{y}_{i,c} \log \frac{1}{M} \sum_{m=1}^{M} softmax(\mathbf{u}_m) \tag{3.8}$$

Where N is the number of examples, C is the number of classes and M is the number of Monte Carlo samples.

Hence, the goal of the wrapper is to endow this model with an aleatoric uncertainty layer given a black-box model. For all purposes, we can not change the black box as it is frozen, and we do not have access to the internals of the model. However, we require an entry point that allows connecting the black box to the wrapper. In this work, we consider that the black box gives access to the pre-normalized logits (before

Figure 3.1: Architecture of the full aleatoric model. The blue components correspond to the original classifier as exposed by the API. In orange, there is the aleatoric trainable part of the model.



the softmax) of the last layer[1]. This is a very mild requirement. It is agnostic to the particular architecture of the classifier since it does not interfere with the model internals and all models display this same structure making the wrapper generic for any architecture.

Figure 3.1 shows an illustration of the proposed wrapper architecture. The wrapper architecture aims at computing aleatoric uncertainty as expressed in Equation 3.5. In that equation we distinguish two components: the original black-box model, shown in blue/gray in figure 3.1, corresponding to $f^{\mathbf{W}}(\mathbf{x})$ and the trainable wrapper architecture, shown in orange/light gray in figure 3.1, that will gives us $\sigma(\mathbf{x})$. The first component corresponds to the logits of the original classifier, what we call μ-logits, and is the result of the last layer when applying the original classifier to the inputs. The second component, $\sigma(\mathbf{x})$, is the aleatoric part of the equation and will capture the variance of the predictions. We train this component using as input the same latent representation resulting from applying the frozen model to the training examples. The result of this component is what we call the σ-logits. It is worth mentioning that the composition

[1]This is consistent with the former use of the notation of the model, $f^{\mathbf{W}}(\mathbf{x})$.

of the μ-logits and σ-logits define a normal random variable layer. The evaluation of the network using random variables requires of its sampling. Note that the same input might generate different instantaneous predictions. Through this sampling process, we use the reparametrization trick, as described in Equation 3.5 to be able to propagate gradients through the wrapper layers and infer the output distributions and statistics. In particular, by analysing the variance of these predictions, we may infer the aleatoric uncertainty and use it as a heuristic for rejecting uncertain predictions, as shown in section 4.

3.1.1 Measuring uncertainty

Using the former wrapper, we have access to the variance of the logits. In this section, we discuss how to compute uncertainty scores based on that measure. In regression systems, it is usual to approximate the output with a Gaussian random variable. In this setting, the uncertainty score can be identified with the associated standard deviation. However, in classification systems, obtaining a single estimate is harder as the random variable is applied to the set of the last layer logits.

For the case of classification, we apply two of the metrics described in section 2.3.3: variation ratios and predictive entropy.

The first heuristic, variation ratios, evaluates the variability of the predictions made when sampling different predictions using the aleatoric model, sort of a measure of the dispersion of the predictions around their mode. Once we trained the wrapper, to obtain the variation ratios we sample M logits as defined in equation 3.2, obtaining M predictions after applying $softmax(\mathbf{u}_y)$. Then we count the number of votes in M to the majority class:

$$c^* = \arg\max_{c=1,\dots,C} \sum_{t=1}^{M} 1[\mathbf{y}_x^t = c]. \tag{3.9}$$

And from that we obtain the variation ratio:

$$VR = 1 - \frac{f_{c^*}}{M} \tag{3.10}$$

The second heuristic, predictive entropy, is based on the information theory and evaluates the average amount of information contained in the predictive distribution. In this case, again, we sample M predictions, drawing samples from the Gaussian distribution $\mathcal{N}(f^w(\mathbf{x}^*), diag(\sigma^2(\mathbf{x}^*)))$. After obtaining the predicted output probability distribution, $\hat{y}$, we compute the predictive entropy by:

$$\mathbb{H} = -\sum_{c} \mathbb{E}[y^*]_c \log \mathbb{E}[y^*]_c. \tag{3.11}$$

Those results with lower entropy values will correspond with confident predictions, whereas a high entropy will correspond with high uncertainty. In our case, we evaluate variation ratios and predictive entropy as both can be obtained directly analysing the output layer of the black-box model. We discard mutual information because it describes the relationship between the distribution of the output and the posterior distribution over the parameters of the model, $\mathbf{W}$, and it is more related to epistemic uncertainty.

In this first approach, by considering the output logits as Gaussian RV, we achieved the goal of being able to measure the aleatoric uncertainty of the black-box predictions. Even though in some circumstances practitioners will have access to the output logits of the system, this requirement of having access to these internal components breaks the constraint of working with black-box systems. It is, therefore, necessary to extend the method with a solution that can deal directly with the output of the classifier.

3.2 Dirichlet approach

Very often, third-party classification APIs will deliver their predictions as probabilities over the predicted classes or, in some cases, as categorical values, preventing the application of methods that require access to the internals of those APIs, like those presented in the previous section or section 2.3.2. In this section, we propose a wrapper algorithm that takes a black-box model and operates on top of it. As such, there are several constraints to observe. First, we need to exclusively operate on the inputs and outputs of the black-box classifier, as we will not have access to the internals of the model, such as the logits. Second, the input of the wrapper has to be compatible with the original distribution over the output classes. Finally, the wrapper must be able to operate on both probabilistic and categorical classification systems.

In the previous section, we showed how by using independent Gaussian random variables to model the pre-activation value of the logits, we could then use Monte Carlo sampling to obtain the variance of the final output. In our opinion, independent Gaussian distributions impose unnecessary assumptions and need for additional normalization steps. Here we consider a more natural approach for the output distribution and suppose it can be modelled by a Dirichlet probability distribution, following Ma-

linin and Gales [2018]; Chen et al. [2018]; Gast and Roth [2018]. Again, the problem with these approaches is that they do not conform to the black-box constraints here imposed, as they need to train the model and to have access to internal parameters. We propose a new method that combines a Dirichlet output distribution with a Monte Carlo sampling method that allows for modelling the uncertainty while observing the black-box restrictions. This approach results in a model that is compatible with both hard and soft predictions.

3.2.1 Dirichlet concentration reparameterization

We denote with $\mathcal{D}_{target}$ the data set of our target application. This is the data corresponding to the domain we want to apply the black-box classifier, f^{bb}. $\mathcal{D}_{target}$ is composed of pairs $\{(\mathbf{x}_i, y_i)\}_{i=1...N}$, where $y_i \in \mathbb{R}^{C_{target}}$ and C_{target} is the number of different classes. This data set is usually different from that used to train the black-box model. When applying the black-box model to our data, we obtain a new prediction $y^*_{target} = f^{bb}(\mathbf{x}_{target})$.

Sometimes, the original black-box model may have been trained with a slightly different set of target labels, C_{bb}, than our domain. Thus we also consider a mapping function m that maps the classes from one set to the other, $m(y) : C_{bb} \rightarrow C_{target}$.

A wrapper is defined as a model, f^w, that considers a black-box and operates on its inputs and outputs while potentially adding new features:

$$y^w = f^w(f^{bb}(\mathbf{x}), \mathbf{x}). \tag{3.12}$$

Initially, we will consider the output of the black-box classifier, $f^{bb}(\mathbf{x})$, to be a vector representing the multinomial distribution associated to the probabilities of belonging to each of the output classes. Later we will relax this assumption and consider pure hard categorical outputs. The proposed wrapper changes the black-box output vector into a random variable that follows a Dirichlet distribution. This can be regarded as modelling the probability of the multinomial black-box output:

$$p(y^w|\mathbf{x}, m(y^{bb}), w) \sim Dir(\alpha), \tag{3.13}$$

where w are the parameters of the wrapper.

We will also impose the wrapper to preserve the translated multinomial output $m(y^{bb})$ of the black-box in expectation:

$$\mathbb{E}(y^w) = m(y^{bb}). \tag{3.14}$$

We propose to use a decomposition of the concentration parameter in two terms to relate the output of the black-box classifier[2], y^{bb}, with the concentration parameter, α, in the Dirichlet distribution of the wrapper. To that effect, we recall some basic statistics of the Dirichlet distribution.

Given a Dirichlet random variable $\mathbf{x} \in \mathbb{R}^C$ with concentration parameter $\alpha \in \mathbb{R}^C$, the expected value of the distribution is defined as $\mathbb{E}(\mathbf{x}_i) = \alpha_i / \sum_{k=1}^{C} \alpha_k$.

Observe that the expected value has the same properties as a probability distribution and that the output of the black-box $y^{bb} \in \mathbb{R}^C$ can also be regarded as a probability distribution. In this sense, we could directly use the output of the black box as the concentration parameter. However, by taking advantage from the fact that each term of the concentration parameter is not necessarily constrained to the interval $[0,1]$, we introduce a new scalar parameter, $\beta \in \mathbb{R}$ that allows the wrapper to adapt the distribution properties to the input data:

$$\alpha = \beta y^{bb}. \tag{3.15}$$

Introducing this parameter fulfills the constraint in Eq.3.14, i.e. $\mathbb{E}(y^w) = y^{bb}$.

By means of this decomposition, while the output of the black-box classifier stands for the mean, parameter β defines the shape of the distribution. This approach share similarities to the decompositions present in other works in a different context[3] Malinin and Gales [2018]; Chen et al. [2018]; Gast and Roth [2018].

This decoupling allows to effectively isolate the contribution of the black-box and the value of parameter β. Figure 3.2 shows the integration of the wrapper (in light green colour) with the black-box classifier (in grey). Observe that the wrapper consists of two blocks:

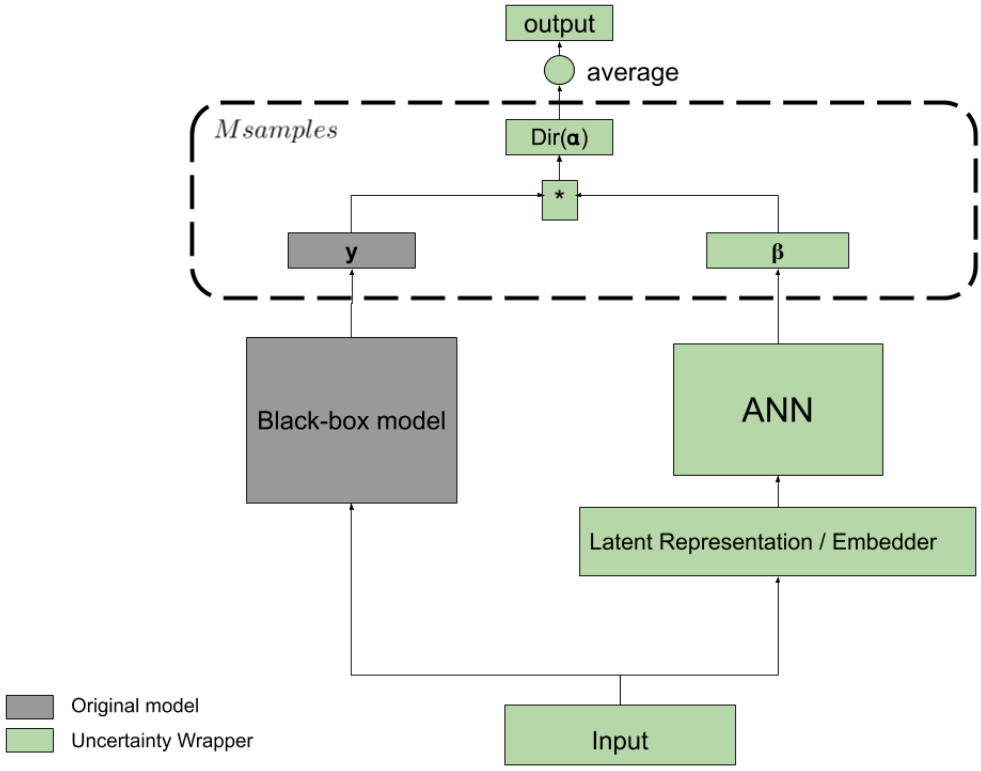
- The Dirichlet reparameterization layer of the wrapper that decouples the influence of the black-box model from the rest (see the dashed line)
- A deep learning architecture[4] which aims to compute the scalar value of β.

[2] For the sake of simplicity. Without loss of generalization, in this section, we assume an identity m function so that $m(y^{bb}) = y^{bb}$

[3] It is worth noting that in the context of those works, there is a degradation in performance when using Dirichlet. This degradation does not happen in our case since the black-box model is non-mutable.

[4] The architecture used in this figure corresponds to the one used in the experimental section.

Figure 3.2: Model used to estimate the aleatoric uncertainty from the original black-box model



Moreover, the figure also shows a possible decomposition of the most usual building blocks:

- A latent representation block that adapts the to the nature of the input data (e.g. recurrent models or embeddings for natural language processing, or convolutional blocks for image data).

- A standard feedforward/convolutional block that squeezes the information into the single output parameter β.

3.2.2 Inference in the Dirichlet setting

Similarly to the Gaussian approach, we approximate the expected value of the classification probabilities using Monte Carlo sampling. The main difference here is the fact that we sample the learned Dirichlet distribution to directly obtain the output instead of each of the logits individually for each sample, as follows:

$$y^w \sim Dir(\alpha), \quad \mathbb{E}[y^w] = \frac{1}{M}\sum_{m=1}^{M} y_m^w, \tag{3.16}$$

,

where M is the number of times that we sample from the Dirichlet distribution. Still, to be able to apply backpropagation to this setting, where we sample from the Dirichlet distribution, we need a way to compute the gradients for the cumulative distribution function. As in the Gaussian version, where we applied the reparametrization trick, here we apply the Implicit Reparameterization Gradients method proposed in Figurnov et al. [2018]. Thus, now it is possible to obtain the gradient of the Dirichlet parameters.

The trick here is to use a standardization function $\mathcal{S}_\phi(z)$ that when applied to a sample from $q_\phi(z)$, removes its dependence on the distribution parameters. This function allows computing the gradient of the expectation as the expectation of the gradients, making it possible to obtain the gradients by sampling from the distribution. In the case of using Dirichlet functions, the proposed standardization is a Gamma function, $z_i \sim \text{Gamma}(\alpha_i, 1)$, and then $\left(\frac{z_1}{\sum_{j=1}^D z_j}, \cdots, \frac{z_D}{\sum_{j=1}^D z_j}\right) \sim \text{Dirichlet}\ (\alpha_1, \ldots, \alpha_D)$.

This expectation obtained from the samples drawn from the resulting Dirichlet distribution defines the outcome of the model. This is used to define the loss function for our learning stage. Given a set of N training samples, we propose to use a regularized version of the cross-entropy loss function as follows,

$$\begin{aligned}\mathcal{L}(W) &= -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{C}\sum_{c=1}^{C}\mathbf{y}_{i,c}\log\mathbb{E}[\mathbf{y}_i^{bb}]_c + \lambda\|\beta\|_2 \\ &= -\frac{1}{N}\frac{1}{C}\sum_{i=1}^{N}\sum_{c=1}^{C}\mathbf{y}_{i,c}\log\left(\frac{1}{M}\sum_{m=1}^{M}\mathbf{y}_{m,i,c}^{bb}\right) + \lambda\|\beta\|_2.\end{aligned} \tag{3.17}$$

Observe that we introduce the norm of the β value in the minimization function. This term is required since the unregularized cross-entropy forces the value of β to grow unbounded. By adding this term, we control its growth and govern the trade-off with a scalarization parameter λ.

3.2.3 Measuring uncertainty

Modelling the output of the black-box model with the described Dirichlet layer allows studying the variability of the parameters of the black-box output distribution. Using Monte Carlo simulation, we can characterize the heteroscedastic aleatoric uncertainty.

Similarly to what we did with the Gaussian variables associated with the logits, in this case, we can directly sample from the Dirichlet distribution.

Then, we can use standard techniques for measuring uncertainty like *variation ratios* or *predictive entropy*.

To measure *Variation ratios*, in the Dirichlet version of the wrapper, for a given data point $\mathbf{x}$ we sample M output vectors from the Dirichlet wrapper $\mathbf{y}_x^t, t = 1 \dots M$, and compute the variation ratios as usual,

$$VR = 1 - \frac{f_{c^*}}{M}$$

where $f_{c^*} = \sum_t 1[\mathbf{y}_x^t = c^*]$, and c^* corresponds to the sampled majority class,

$$c^* = \arg \max_{c=1,\dots,C} \sum_{t=1}^{M} 1[\mathbf{y}_x^t = c]. \tag{3.18}$$

Alternatively, for *predictive entropy* considers the average amount of information contained in the predictive distribution. In our case, this corresponds to the vectorial output of the black-box. Results with low entropy values correspond to confident predictions, whereas high entropy leads to large uncertainty. Since the wrapper allows us to model the variability of the parameters of the black-box output distribution, we can compute a score takes into account the variability of the predicted value by sampling from the wrapper inferred distribution, the sampled predictive entropy, defined as

$$\mathbb{H} = -\sum_c \mathbb{E}[\mathbf{y}^{bb}]_c \log \mathbb{E}[\mathbf{y}^{bb}]_c. \tag{3.19}$$

At this point, we would like to remark two relevant aspects of the Dirichlet version of the wrapper:

- It is fully compatible with a black-box setting. By operating directly with the output of the original classification system, the wrapper abstracts completely of any implementation or training details. This capacity of abstraction allows applying the wrapper for any BB, no matter whether the model implementation uses the last state-of-the-art DL architecture, a simple logistic regression or whether the outputs come from human evaluations. As illustrated in Figure 3.2, as long as we have a way to represent inputs in the latent representation layer, we can apply the ANN and the Dirichlet composition layers to obtain the corresponding output probability functions.

- The present method is not constrained to probabilistic classification systems. The Dirichlet wrapper can consider the output of categorical classifiers as a probability distribution centred around the predicted class and apply the same method as for probabilistic outputs. This compatibility with hard predictors adds a layer of abstraction and makes the proposed method fit in scenarios where confidence based methods are not applicable.

All in all, we showed how the Dirichlet wrapper here presented provides practitioners with means for obtaining the aleatoric uncertainty of the predictions of a given classification API. Once trained to the target use case and deployed, the method will be able to identify inputs where the BB is uncertain. A potential issue of this method may come with possible variations in the inputs of the system once deployed in the wild. The apparition of out-of-distribution examples that are from different distributions of both the original and the target data sets may cause a degradation of the system. In the next section, we see how to extend the proposed method to deal with these OOD points.

3.3 Out-of-distribution

In the previous section, we introduced a deep learning wrapper, based on the Dirichlet distribution, that given any black-box classification system allows aims to measure three different uncertainty sources, namely:

- uncertainty associated with cases that are close to the decision boundary,
- uncertainty associated with cases that, while being far from the decision boundary, get overconfident but erroneous predictions,
- uncertainty associated with out of source distribution cases.

These measures may serve as a prophylactic measure against the possible changing environment or distribution shifts in the data when the system is deployed. With regards to the last one, the OOD detection, we align with the definition given in Malinin and Gales [2018], where they explicitly separate the three types of uncertainty, i.e. epistemic, aleatoric and distributional. In this section, we describe a method for tackling the third one, where the goal is to identify samples that do not belong to the distributions of data used for training nor the original black box neither the wrapper.

The method proposed here for furnishing the Dirichlet wrapper with OOD detection consists of inserting, during the training stage, OOD samples and teaching the model to tell from in and out of distribution examples. To illustrate the concept, we can think of the process as a sort of vaccine that inoculates OOD samples to the model to develop antibodies that allow their detection on prediction time. To ease this detection, we force the model to attribute maximum entropy to the OOD points detected.

During the learning phase, we use the minimization of an extended version of the loss function presented in equation 3.17. In this OOD version, this loss function is split into two terms. The goal of this separation is to use an alternating learning schema that will allow to train the estimation of the uncertainty for the target training set and also model out-of-distribution points. For the former, we use a regularized version of the cross-entropy loss function described in equation 3.17:

$$\begin{aligned} \mathcal{L}_{ale}(W) &= -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{C}\sum_{c=1}^{C}\mathbf{y}_{i,c}\log\mathbb{E}[\mathbf{y}_i^w]_c + \lambda\|\beta\|_2 \\ &= -\frac{1}{N}\frac{1}{C}\sum_{i=1}^{N}\sum_{c=1}^{C}\mathbf{y}_{i,c}\log\big(\frac{1}{M}\sum_{m=1}^{M}\mathbf{y}_{m,i,c}^w\big) + \lambda\|\beta\|_2 \end{aligned} \tag{3.20}$$

The second term of the loss minimizes the Kullback-Leibler divergence between the expected output and a Uniform distribution over the labels Lee et al. [2018]:

$$\mathcal{L}_{ood}(W) = \mathcal{KL}(\mathcal{U}(\mathbf{y})\|\mathbb{E}[\mathbf{y}^w]) = \mathcal{KL}(\mathcal{U}(\mathbf{y})\|\big(\frac{1}{M}\sum_{m=1}^{M}\mathbf{y}_m^w\big)) \tag{3.21}$$

Both losses are combined into the final training loss as follows,

$$\mathcal{L}(W) = (y_o)L_{ale}(W) + (1-y_o)L_{ood}(W) \tag{3.22}$$

In order to enable the alternating learning scheme when modelling the out-of-distribution, we include an additional label y_o that is used to change among both losses and identifies whether the sample corresponds to the target data set or to an additional set that is modelling the out-of-distribution. To this effect, the original training set gets augmented with a set of inputs that are intentionally out of the original distribution. For example, in natural language processing, the added elements might belong to a domain different from the target problem. In computer vision, they might be images from categories not present in the target application. The label, y_o, will tell whether the input belongs or not to the distribution.

As mentioned, the KL term in equation 3.21 will force the model to emit a maximum entropy for OOD. Later on, by applying the same metrics for obtaining an uncertainty score, we will be able to identify those points, allowing the practitioner to intervene on these problematical predictions.

3.4 Illustration of the Dirichlet Wrapper and Uncertainty Measurement

In this section, we introduce a toy scenario to illustrate the intuition of the proposed method. Recall that we are dealing with a black-box, single-label probabilistic classifier. The outcome of this system consists of a probability distribution of C different possible categories. We assume an original black-box, trained to match the categorical distribution of the labels in an unknown source domain.

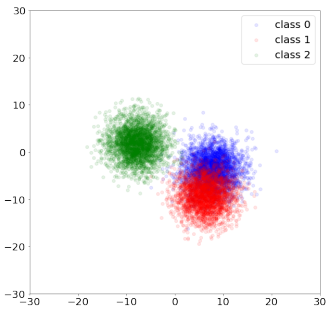
This experiment aims to illustrate how the methods presented in this work, i.e. Gaussian, Dirichlet and OOD versions of the uncertainty wrapper. The first part of the experiment focuses on comparing Gaussian and Dirichlet implementations of the wrapper. Results will show how the Dirichlet version delivers better results than its Gaussian counterpart in addition to being applicable in actual black-box scenarios.

We generate a total of 20000 two dimensional points corresponding to three different classes, each of them modelled as a Gaussian distribution (Figure 3.3(a)). These points are split into train, validation and test sets. We randomly select 9600 points for training a classification model for the three classes. The accuracy of the resulting classifier reaches 88.86%. We consider the trained classification model as the simulated black-box.

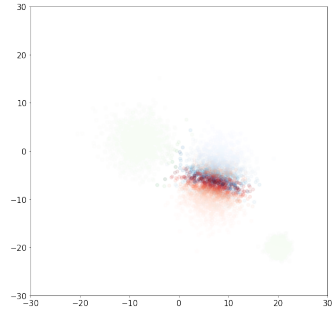
In order to exemplify potential changes in the data distribution when applying the black-box to a different data domain, we introduce in the test set a new set of points, labelled as class 2, far from the class 2 original distribution. We can consider these points as a new cluster from class 2 that was not present in the source domain. Figure 3.3(d) shows the result of applying the trained classifier to the test dataset. Note that the new cluster falls in the class 1 region, inducing an erroneous prediction for all these points. Now the black-box classifier achieves a 71.76% accuracy score in the test data set.

Observing the decision regions defined by the classifier, one can see three potential locations where predictions are prone to be wrong. The first error-prone region corre-

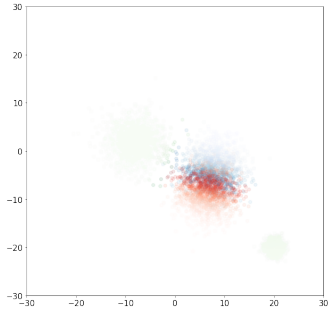
Figure 3.3: Result of applying the black-box classification to the toy test dataset



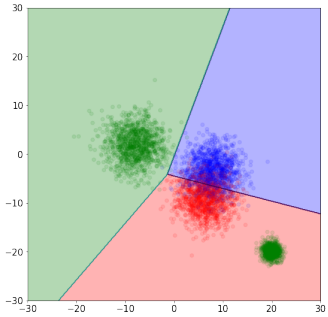
(a) Original synthetic dataset used to train the black-box classifier (BB)



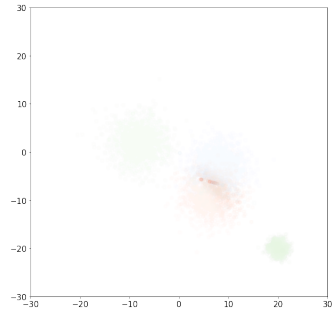
(b) Uncertainty as measured using the softmax response of the BB



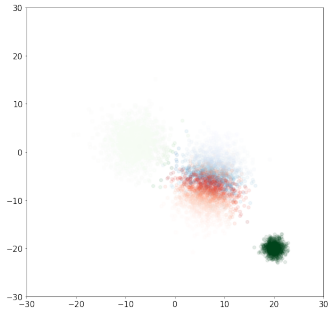
(c) Uncertainty as measured using predictive entropy using a Gaussian RV for the logits



(d) Learned BB applied to the test synthetic dataset



(e) Uncertainty as measured using variation ratios using a Gaussian RV for the logits



(f) Uncertainty as measured using predictive entropy resulting from the application of the proposed Dirichlet wrapper

sponds to the decision boundary between class 2 against the rest of the classes. The second one corresponds to the boundary between class 0 and class 1. Points close to these regions produce uncertain predictions. The third region of high uncertainty can be found in the area of the new class 2 cluster.

Figure 3.3 shows the results of applying the different methods for computing the uncertainty on this simple problem. Figure 3.3(b) shows the uncertainty measured by using the softmax-response of the predictions obtained with the black-box classifier. Figures 3.3(c) and 3.3(e) show the uncertainty obtained by computing the predictive entropy and variation-ratios respectively, by using a Gaussian random variables for the logits. Finally, Figure 3.3(f) displays the uncertainties obtained by using the pro-

posed Dirichlet wrapper. Figures show the uncertainty metrics computed for each class, where the lighter the colour of the point is, the more confident the associated prediction. Looking at the results, we can see that the Dirichlet method exhibits the best performance, detecting all enumerated sources of uncertainty.

Looking in detail to Figure 3.3(b), we observe how this metric can detect the uncertainty from the decision boundaries, but it is not able to identify the new cluster. We have to recall that in this case, we consider the output of the classifier as a single point estimate, and we try to compute the confidence of the prediction based only on the output probabilities. We, therefore, apply a method based on the distance to the decision boundary of the predictions obtained, the softmax response, traditionally used as the confidence score for probabilistic classification systems.

In figures 3.3(c) and 3.3(e), we apply a the Gaussian version of the wrapper, as presented in section 3.1, for capturing the aleatoric uncertainty. We would like to recall that in this case, we need to access the logits as they model the noise of the output predictions with Gaussian random variables with a diagonal variance term for each of the logits, $\mathbf{u}$, as follows $\mathbf{u} \sim \mathcal{N}(f^w(\mathbf{x}^*), diag(\sigma^2(\mathbf{x}^*)))$. This approach models uncertainty through the σ parameter expressing the variance of the distribution of the logits. As detailed in section 3.2.3, after modelling the logits as Gaussian random variables, we still need to obtain a numerical score for the uncertainty. Hence, we use two different methods for obtaining such score: predictive entropy (Figure 3.3(c)) and variation ratios (Figure 3.3(e)). If we analyse the results obtained with the two uncertainty metrics, we see that each one focuses on a different type of uncertainty. Predictive entropy can capture the uncertainty of points near the decision boundary, similarly to softmax response. Variation ratios are able to capture the uncertainty associated with the noisy class two labels, while at the same time capturing some of the uncertain points at the decision boundaries.

In order to understand this behaviour, we should examine the value of the standard deviation for each random variable. Observing the values for the σ parameter, we distinguish three different behaviours. First, for points that are far from the decision boundary between the original class 2 samples and the rest, the values for the sigma tend to be close to 0. As it gets closer to this boundary, the value of sigmas starts to grow until 20. Finally, the class 2 points that correspond to the new cluster at the bottom right, they all exhibit the higher σ values, from 20 to more than 40.

Based on these observations, we can conclude that the entropy of the original prediction heavily determines predictive entropy. As such, it can detect the uncertainty

at the decision boundary. However, it is not able to identify the new class 2 cluster. All those points have a very confident (wrong) prediction, resulting in a minimal probability for the correct class. Then, even if the uncertainty associated with those points is high, the effect of the sampling on those points is diluted when averaging for computing the entropy value.

Regarding variation ratios, we observe more sensitivity for the different kind of uncertainties included in this toy problem. First, for points in the decision boundaries, subtle variations in their logit values likely forces a change in the predicted class for some cases. This accounts for larger uncertainty values associated with points close to the decision boundaries. More interesting is the new class 2 cluster. Changes in the predicted label have a significant impact in this case. This is since we are merely counting changes and not averaging them as in the case of predictive entropy. Thus, even when the number of prediction changes is below 10%, the associated uncertainty is higher compared to those points that have no modifications.

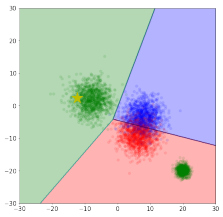
Finally, Figure 3.3(f) shows the uncertainty score obtained by using the Dirichlet wrapper. Next subsection explicitly analyses this figure and builds the intuition of the different types of uncertainty found.

3.4.1 Analysis of the measured uncertainties when using the wrapper

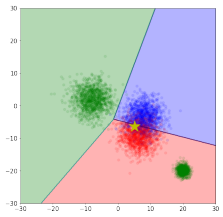
Now we analyse some examples to see how the Dirichlet method works on different cases. To that effect, we will refer to Figure 3.4, where different cases are plotted jointly with their predicted Dirichlet distribution. The bottom line figures show the contour and concentration of the points distributed in the simplex of the three classes. Each class corresponds to one of the corners of the triangle: the bottom left corner one corresponds to class zero, the bottom right one to class one and the top corner to class two.

3.4.1.1 Example of a very confident prediction

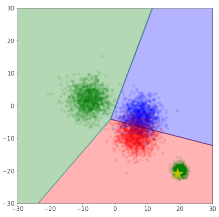
First, we consider a point from class 2 that belongs to the original class 2 set at the left side of the figure. In this case, we would expect it to have low uncertainty, as it corresponds to a confident and right prediction. Figure 3.4(a) shows an example of this case as a yellow star. The point belongs to class 2 and the classifier outputs a prediction of $[3.01 \times 10^{-10}, 2.75 \times 10^{-11}, 1.00 \times 10^{0}]$, very confident of belonging to



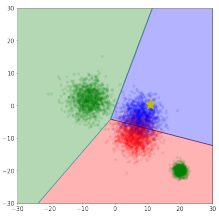
(a) Example of a point at very confident region from class 2



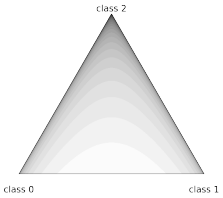
(b) Example of a point at the decision boundary between class 0 and 1



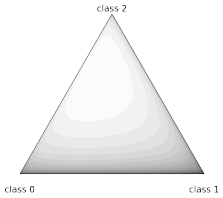
(c) Example of a point that belongs to class 2 but is predicted by the BB as class 1 with high confidence.



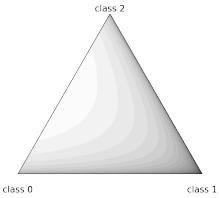
(d) Example of a point that belongs to class 1 but is in the class 0 region



(e) We observe how, as the prediction is right, the concentration of the distribution is around class 2.



(f) We observe how the distribution concentrates in the line between the class 0 and 1.



(g) We observe how the distribution concentrates around class 1 even though the point belongs to class 2

(h)We observe how the distribution concentrates around class 0 even though the point belongs to class 1

Figure 3.4: Examples of points from the toy problem together with the corresponding Dirichlet distribution.

class 2. After applying the wrapper, we obtain a beta value of 0.11, quite high for the operational regime used. The alphas in this case are $[3.03 \times 10^{-11}, 2.77 \times 10^{-12}, 1.00 \times 10^{-1}]$. Figure 3.4(e) shows the corresponding Dirichlet distribution. In this example, we see that the points concentrate around the top corner, that correspond to a very confident prediction of class 2, which aligns with their mean value $[2.54 \times 10^{-8}, 2.54 \times 10^{-8}, 9.99 \times 10^{-1}]$. In this case, the entropy value is very low: 3.86×10^{-10}.

3.4.1.2 Example of a prediction at the decision boundary

Next, we examine a class 1 point that corresponds to the set at the bottom, close to the decision boundary between class 0 and 1. In this case, because the prediction is not that confident, we would expect it to have a higher uncertainty than before. The chosen point corresponds to the yellow star in Figure 3.4(b).

The point belongs to class 0, and the classifier outputs a prediction of $[4.12 \times 10^{-1}, 5.87 \times 10^{-1}, 2.67 \times 10^{-6}]$, predicting the point as class 1 but with less confidence. After applying the wrapper, we obtain a beta value of 0.084, quite high for the operational regime used, but less than before. The alphas in this case are $[3.45 \times 10^{-2}, 4.91 \times 10^{-2}, 2.23 \times 10^{-7}]$. As before, we observe the distribution in Figure 3.4 (f), where we see a higher variance. We can see that there is a higher probability concentrated around the line that connects class 0 and class 1. This is a clear indication that the confidence in that prediction is small. In this case, the entropy is a bit higher, 0.68.

So far, the uncertainty detected does not differ a lot from what a softmax response in a well-calibrated classifier can identify. The critical point of the proposed method lies in its ability to detect overconfident predictions; this is high confidence samples with wrong predictions.

3.4.1.3 Example of a prediction for wrongly labelled points

Consider now an example from class 2 in the new cluster. The chosen point corresponds to the yellow star in Figure 3.4(c). In this case, the point belongs to class 2, but the classifier outputs a prediction of $[1.13 \times 10^{-3}, 9.98 \times 10^{-1}, 8.86 \times 10^{-24}]$, considering it to be class 1 with high confidence. Note that despite the high confidence value, the prediction is wrong if we are in the target domain. After applying the wrapper, we obtain a beta value of 1.06×10^{-4}. The alphas, in this case, are $[1.21 \times 10^{-7}, 1.06 \times 10^{-4}, 9.46 \times 10^{-28}]$ that produce the density displayed in Figure

3.4(g). In this case, the distribution shows large probabilities in the connections between class 1 and 0 and 2. Computing the expected value in this case, we obtain $[0.33, 0.33, 0.33]$ that corresponds to a value of entropy of 1.09. This assigns the point with considerable uncertainty.

3.4.1.4 Example of a non-captured uncertain point

Finally, we would like to discuss a limitation of the proposed method for the toy scenario described here. Let us analyse, for example, a point that belongs to class 1 but is surrounded by class 0 points. The point corresponds to the yellow star in Figure 3.4(d). The point belongs to class 1, but the classifier outputs a prediction of $[9.93 \times 10^{-1}, 6.60 \times 10^{-3}, 3.07 \times 10^{-8}]$ and considers it to be class 0 with high confidence. After applying the wrapper, we obtain a beta value of 0.087, not very small. The alphas, in this case, are $[8.73 \times 10^{-2}, 5.80 \times 10^{-4}, 2.70 \times 10^{-9}]$ that produces the density distribution shown in Figure 3.4(h).

In this case, we observe that the higher density values are found around class 0 and 1. We can further see this effect by observing that the values corresponding to those classes are more significant in the expected value of the distribution, $[9.92 \times 10^{-1}, 7.60 \times 10^{-3}, 7.88 \times 10^{-10}]$. In this case, the entropy is quite low, 0.044, and the wrapper fails to assign this point a substantial uncertainty value. The Dirichlet wrapper is not able to capture the wrong point, since, despite belonging to class 1, it is in a zone with a very high concentration of points labelled as 0.

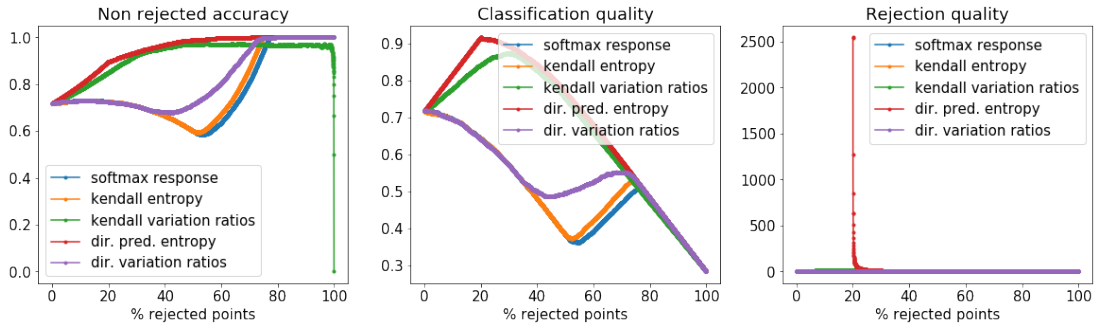
3.4.1.5 Example wrap-up

To sum up, we observe that for confident predictions, the value of the beta parameter is much higher than for those with high uncertainty. Here we would like to highlight that, in the proposed model, β belongs to the interval $[0, 1)$. This interval differs from other approaches that use Dirichlet for modelling the uncertainty, Sadowski and Baldi [2018]; Malinin and Gales [2018]; Chen et al. [2018], where they work with unbounded values bigger than 1 for the β value. Working with values of $\beta > 1$ produces simpler uni-modal distributions. On the contrary, in the regime of work used in this work, the distribution is originally multi-modal, thus avoiding stability problems regarding the lack of control in the growth of the parameter.

Working in the interval $[0, 1)$, we observe that confident predictions usually take values of βs close to 0.1. When drawing samples from the corresponding Dirichlet

distribution, this value alters a bit the expectation of the output, but still produces similar values as before. For points where the black-box itself is uncertain, for example, those close to the decision boundary, the wrapper assigns a β a bit lower, around 0.05. This β still does not affect too much the generated prediction samples drawn from the Dirichlet. Thus, the final prediction keeps being uncertain. Finally, for those predictions that are very confident but correspond to very erroneous outcomes, the model estimates a very low β. In this case, the Dirichlet is forced to output points with high entropy, hence marking them as uncertain predictions.

Figure 3.5: Classifier with rejection using different uncertainty metrics: softmax response, Gaussian predictive entropy and variation ratios and Dirichlet wrapper predictive entropy.



Consequently, we claim that the proposed method seems to better capture the notions of uncertainty in this simple problem when working with the soft predictions of the black-box classification system. We now compare the results of applying the proposed uncertain parameter for the rejection system against the rest of the metrics analysed. Figure 3.5 shows the results of the three performance metrics described in section 2.5.3. From left to right, the chart displays the value of the corresponding metric after rejecting the percentage of points indicated. We observe how the softmax-response and the Gaussian wrapper entropy exhibit similar results, as they struggle on detecting the noisy class 2 data points. The variation ratios obtained with the wrapper shows a bit better result than the previous two, but it is still insufficient to detect wrong predictions at low rejection ratios. The two best results correspond to variation ratios of the Gaussian version and the predicted entropy obtained with the Dirichlet wrapper. In the case of the later, by rejecting only 20% of the less confident points, we can increase the accuracy of the preserved data points in almost 20 points, outperforming the results obtained with Gaussian RV. Besides, we want to highlight that in the case of the

wrapper, the model has no access to the internal details of the black-box.

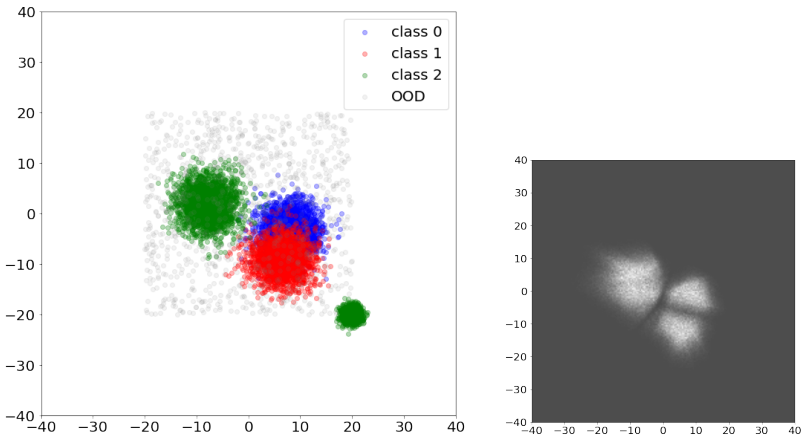
3.4.2 Evaluating out-of-distribution samples

Following the motivation of the Dirichlet wrapper and the role of uncertainty as an excellent proxy for capturing noisy labels when using black-box models in target applications, in this section, we motivate the addition of the out-of-distribution loss presented in section 3.3.

Taking advantage of the toy problem introduced in the present section, we will show how to train the model to enable the detection of out-of-distribution data inputs. First, we generate random points drawing samples from a Uniform distribution centred around the original training distribution of the wrapper. The goal of adding these points is to simulate potential out-of-distribution inputs that might arise in prediction time. Next, we add these synthetic data points to the training dataset defined as in figure 3.6 (a). To differentiate between the two types of points, we add an extra label with values $y_o = 1$ for the original data points and $y_o = 0$ for the newly generated data inputs. With the augmented dataset, we trained the wrapper model using the out-of-distribution loss function as defined in equation 3.22.

Figure 3.6(b) shows the results of applying the out-of-distribution wrapper, the darker, the more uncertain the prediction is. In order to validate the proposal, we further draw the samples from a broader Uniform distribution, corresponding to a 100×100 bounding box around the test points, instead of the 20×20 one used for training. In this respect, we validate that the method generalises for out-of-distribution samples not seen during the training stage. The results show that the out-of-distribution wrapper preserves the ability to detect uncertain predictions, showing similar results to those obtained in 3.3. Moreover, we also note how the model identifies the new out-of-distribution points by assigning a high uncertainty to those points.

To sum up, we conclude that the out-of-distribution wrapper here proposed is a handy tool for assessing the performance of a black-box classifier given a new target application. With only a small fraction of labelled data from the target domain, we can train the model to estimate the uncertainty of future predictions. This uncertainty can help with the evaluation of different black-box APIs or can provide an actionable mechanism to improve the performance of them in production through rejection strategies. We also show how by augmenting the target datasets and using a new loss function, we can deliver a more robust method of uncertainty estimation that can prevent the model



(a) (b)

Figure 3.6: (a) Training set including out-of-distribution points from $U[-20,20]$ and (b) Uncertainty of points in $U[-40,40]$ (The darker, the more uncertain)

from degrading in case of future changes in the distribution of the inputs.

The experiment described in the present section validates the proposed method as illustrated with a toy scenario. In the next section, we showcase how the wrapper is also capable of modelling uncertainty for more complex situations dealing with real data.

Chapter 4

Experiments

Contents

This section contains the experiments carried out to validate the effectiveness of the proposed model using real data. The tests included aim at simulating the central use case of this work, i.e. to evaluate the performance of a pre-trained black-box in a target scenario, and how to respond when this performance is not satisfactory.

In all the cases, we take a source domain with a large number of labelled training samples and proceed to simulate a black-box classification model for this source domain. Next, we freeze this model and pretend its exposition as it was an API part of an MLaaS platform.

We then change roles and act as a consumer of this API. At this stage, we assume that we do not have any information about the details of the API and we want to evaluate it when applied to a given target domain with a small number of labelled training samples. The first thing to do in this case is to assess whether or not the output of the API, the predicted categories, fit our problem or whether a mapping function to adapt the API results to the target use case is required. Then, we proceed to call the API for our target dataset and obtain the output predictions, calling the mapping function when needed.

In order to simulate the case when the classifier outputs hard predictions, we generate a new set of predictions based on the probabilities obtained with the black-box. However, only the class with maximum probability is reported as output.

Next, we take the target training samples together with the predictions obtained to train the Dirichlet wrapper, after what we can call it to estimate the uncertainty score associated with each prediction. At this point, we can use the score to reject uncertain predictions or to evaluate the performance on each class and decide whether to keep using the API, try another one or collect more data to train our model.

In particular, in the experiments described, we make use of the uncertainty score for building a rejection system. To evaluate the performance of the resulting rejection classifier, we follow the measures proposed by Condessa et al. [2015], as described in section 2.5.3.

Scenarios proposed to validate the method include use cases in Natural Language Processing and Computer Vision. In particular, we consider the problems of sentiment analysis and image classification. In both cases, pre-trained models are applied to solve different tasks than those they were trained for. This scenario shows how the use of uncertainty to detect problematic examples can help in increasing the overall quality of the predictions obtained for the new task.

4.1 Sentiment Analysis

The impact of domain shift and the consequent domain adaption of machine learning models is very relevant in the field of natural language processing. Many previous works have studied this issue in NLP tasks in general Mou et al. [2016] and sentiment analysis in particular Tan et al. [2009]; Glorot et al. [2011]. The central goal of these contributions was to train models that are capable enough to capture the differences between domains and be able to generalize across them. Differently from these words, our proposal focuses on the problem of detecting this shift between the training dataset of an original black-box model and the data of a given application and provide a means to identify difficult examples that may lead to overconfident and wrong predictions.

In order to validate the method proposed, we applied the wrapper to an NLP-based sentiment analysis system applied to product reviews. The experiment consists of training a sentiment analysis classifier using reviews of a given domain and evaluating the uncertainty of the predictions obtained when applying the pre-trained model to a new domain.

To illustrate real scenarios, we include two types of experiments: in the first one, we train the black-box with Amazon products of a section with multiple reviews. These reviews correspond to 1 to 5 stars assessments made by Amazon's users. Then we apply the black-box to reviews of another department and evaluate the performance of the model. The second experiment consists of training a model using reviews of Yelp venues, rated as 1 to 5 stars, and apply the model for IMDB movie reviews in a binary setting. In this case, we need to provide the mapping function that transforms the original five classes to a binary problem. Finally, for the sake of completeness, we also apply the Yelp classifier to the Amazon's product reviews and the Amazon classifier to the SST-2, adapting in each case for the number of target classes. In total it adds to 30 experiments.

The experiments are conducted using the following datasets:

- Amazon Multi-Domain Sentiment dataset contains product reviews taken from Amazon.com from many product types (domains) Blitzer et al. [2007]. The dataset consists of ratings from 1 to 5 stars despite 3-star reviews, i.e. reviews with neutral sentiment were not included in the original.

- Yelp challenge 2013[1], the goal is to classify reviews about Yelp venues where their users rated them using 1 to 5 stars. This dataset is used to train the black-box applied to the SST-2 problem. As mentioned, we need to transform the Yelp dataset from a multiclass set to a binary problem, grouping the ratings below three as a negative review, and as positive otherwise.

- Stanford Sentiment Treebank Socher et al. [2013], SST-2, binary version where the task is to classify a movie review is positive or negative.

Table 4.1 lists the datasets used for the NLP experiments.

We train a deep learning classifier for the big datasets(Yelp2013 and Amazon's music and DVD product reviews) using a model based on a representation obtained averaging the Word2vec embeddings Mikolov et al. [2013b] on the review words. Once trained, we use the model to predict the sentiment using the rest of datasets.

After obtaining the predictions for each target domain, we used the proposed Dirichlet wrapper to estimate the uncertainty score for each prediction. Finally, we evaluate the accuracy of the predictions using different values of the uncertainty score for rejecting uncertain examples.

[1] https://www.yelp.com/dataset/challenge

Table 4.1: Datasets used for NLP experiments

	Train	Validation	Test	N. classes
Amazon apparel	5,828	648	2,776	4
Amazon camera	4,666	519	2,223	4
Amazon computer	1,994	222	555	4
Amazon dvd	89,593	9,955	24,884	4
Amazon electronics	14,495	1,616	6,903	4
Amazon health care	4,551	506	2,168	4
Amazon kitchen	12,509	1,390	5,957	4
Amazon magazines	2,639	294	1,258	4
Amazon music	109,733	12,193	52,254	4
Amazon toys	9,465	1,032	2,630	4
Amazon videos	22,793	2,533	10,854	4
SST-2	65,538	872	1,821	2
YELP2013	186,189	20,691	22,991	5

Figure 4.1 compares the performance metrics of the three rejection measures: the entropy of the predictions of the black-box (*baseline*), the predictive entropy of the wrapper (*pred. entropy*) and the predictive entropy of the wrapper when applied to a categorical output of the black-box (*pred. entropy cat.*).

First, we compare the two uncertainty metrics derived from the wrapper when applied to soft and hard predictions with the predictive entropy of the black-box. This metric is obtained directly from the predictions of the pre-trained model by computing the entropy over the softmax probabilities.

It is worth noting that one can replace predictive entropy by other different metrics without differences in the results. In Figure 4.2 we compare results obtained using different baseline metrics,, namely:

- Predictive entropy, Dagan and Engelson [1995], understood as the entropy of the softmax probability distribution.

- Softmax response, as the maximal neuronal response of the softmax layer.

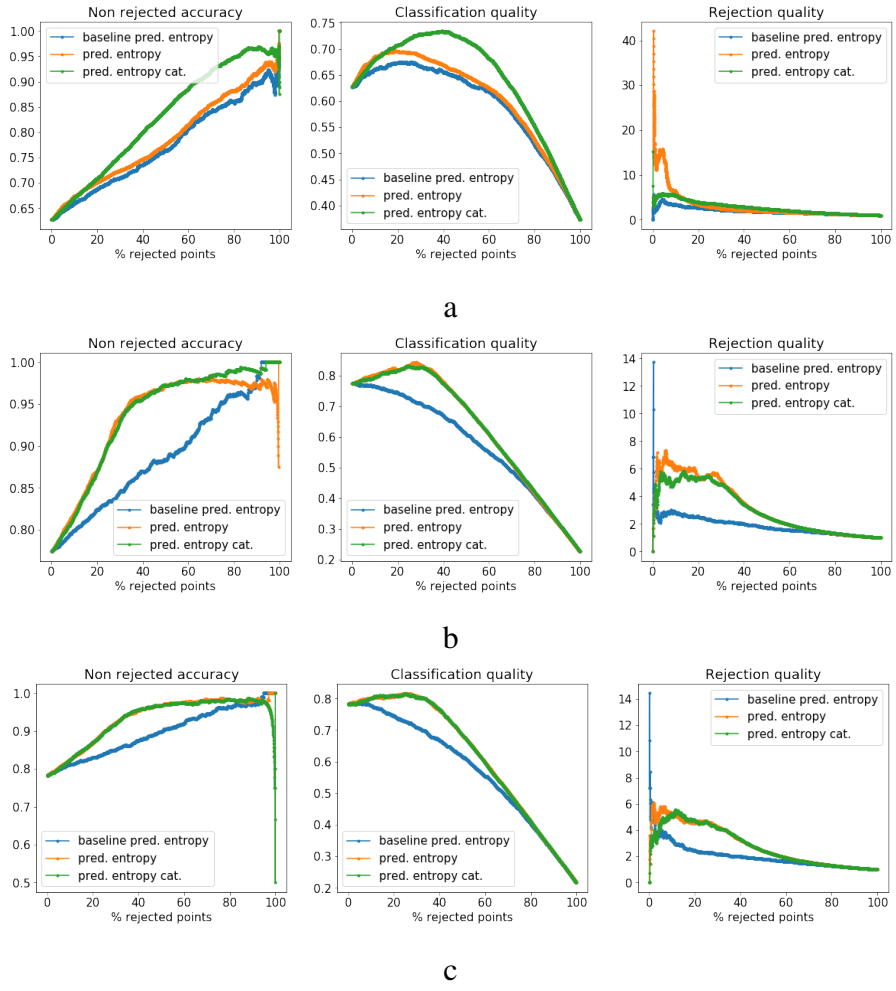


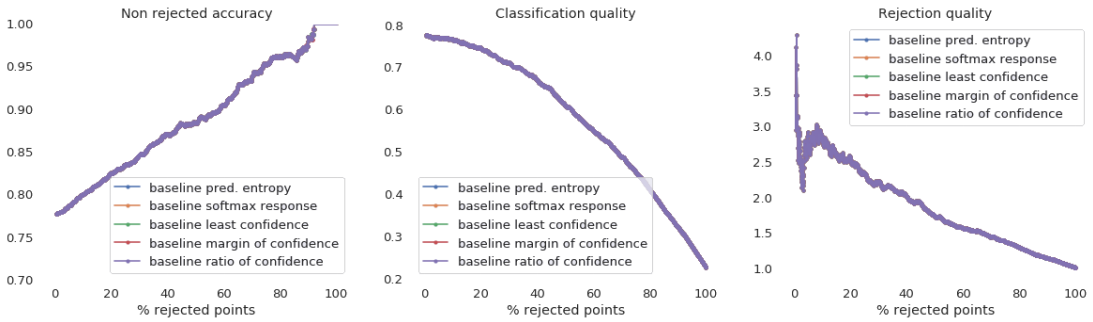
Figure 4.1: Apply Amazon Music BB to Video reviews, Yelp 2013 BB to SST-2 movie reviews and Amazon video BB to SST-2 movie reviews

- Least confidence, as defined in Culotta and McCallum [2005].
- Margin of confidence and ratio of confidence, as defined in Scheffer et al. [2001], understood as the margin and ratio respectively between the first and second more confident output units.

Moreover, we would like to remark that the comparison with the predictive entropy obtained for categorical predictions is included as a reference, taking advantage of the fact that here we simulate the API. In the real world, the baseline used here will not be available as it is based on the output probability distribution.

On each experiment, the three pictures refer to the three performance metrics (non-rejected accuracy, classification and rejection quality) on each rejection point. The X-axis corresponds to the percentage of points rejected at a given point, and the Y-axis

Figure 4.2: Comparative of different baseline metrics that obtain an uncertainty score directly from the black-box prediction, as applied for the Yelp black-box to SST-2 scenario. The baselines compared show similar performance when used for sorting uncertain predictions in the rest of the experiments.



depends on each performance metric. For the non-rejected accuracy, the value displays the accuracy of the predictions for the non-rejected points. For the classification quality, the value shows the percentage of points that are right and not rejected plus the wrong and rejected versus all the points. Finally, the rejection quality corresponds to a proportion between the rejection and the misclassification. In the first and second metrics, the closer the value to 1 the better, while in the third one values above 1 mean a good rejection point. An optimal rejection point, therefore, will be one that combines excellent performance in the three metrics.

To apply the rejection, we proceed as follows. For each point in the test set, we proceed to apply the uncertainty function. In the case of the black-box predictive entropy, the value is obtained directly from the prediction, computing the entropy of the probability distribution of the softmax. For the Dirichlet predictive entropy and the variation ratios, we apply the wrapper to the input and the corresponding black-box prediction to obtain the uncertainty. Once we have a value of the uncertainty for each point, we sort from more to less uncertain, and we start to reject those more uncertain. Each point in the X-axis shows the performance of the resulting classifier after rejecting $x\%$ points in the test dataset. Each line in the plots displays the performance of each of the uncertainty functions compared.

According to the results obtained, the predictive entropy obtained after the proposed method shows better behaviour in all scenarios and metrics. As we remove more samples according to the uncertainty, the proposed method displays much better accuracy than its counterparts. These results validate the hypothesis that the aleatoric

uncertainty computed by the wrapper effectively captures the confidence in the prediction and the samples prone to error. Note that, although our proposal performs much better, its absolute gain depends on the scenario. In domains where there is a need to adapt the original model to the particular target application, there is more to gain by using the wrapper. If we observe the classification quality (plot at the centre of each figure) and the rejection quality, we can see that the proposed metric is also excellent at rejecting the miss-classified points.

Concerning the predictive entropy for the categorical version of the black-boxes, we observe that when compared with the performance of the baseline and the predictive entropy of the soft predictions, in some scenarios we can observe that the performance is a bit lower. However, we have to recall that this comparison is not fair as in categorical scenarios, we will not have access to the probabilities. Even though we still appreciate that it is a good rejector, even when the predictions are categorical.

Figure 4.3 displays a summary of the results obtained. On each plot, the x-axis corresponds to the accuracy of the black-box as applied to a target domain[2]. Y-axis shows the accuracy of the predictions obtained by applying the wrapper model, and the non-rejected accuracy for 10%, 20%, and 30% of the points using the predictive entropy. We show the results for both the probabilistic and categorical versions of the black-boxes. We also plot a line that displays the accuracy of the BB to compare the results.

In all the experiments, the accuracy obtained when using rejection outperforms the original black-box, and we can observe how by increasing the number of rejected points, the accuracy increases accordingly. Even though the gains are higher for the probabilistic version, we still can appreciate how the predictive entropy for the categorical predictions behaves like a good rejector too.

Moreover, we would like to remark the value in the first plot, that displays the accuracy of the wrapper applied to the target data source. In this case, to predict the values, we follow the same approach as for the entropy: once we have modelled the output as a Dirichlet distribution, we sample from this distribution N times and take the expected value as the prediction. In this case, we would like to highlight how just applying the wrapper on top of the original black-box, in some applications, we see an increase in the accuracy without the need of rejecting any sample.

Table 4.2 and Table 4.3 shows a detail of the numerical results obtained during the

[2]The gap between accuracy scores correspond to the difference of results obtained from the binary and multi-label classifiers applied for movies and Amazon products respectively

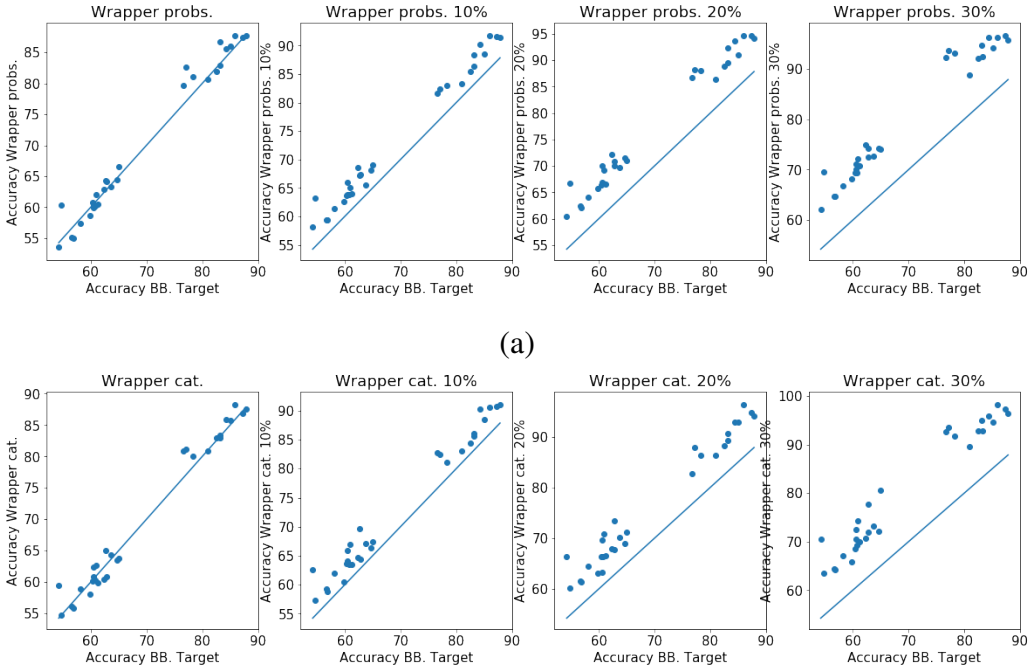


Figure 4.3: Non-rejected accuracy of the wrapper compared to original BB with both probabilistic (a) and categorical (b) outputs

experiments for the different combinations tested, both for probabilistic and categorical outputs. The first column, black-box source acc, describes the accuracy obtained for the source dataset after training the original classifier. Next, column black-box target acc. describes the accuracy obtained when applying the black-box to the target dataset. The third column displays the result of obtaining predictions by applying the wrapper model, sampling from the learned Dirichlet output distribution. The rest of the columns show the non-rejected accuracy and the classification and rejection quality after rejecting 10, 20 and 30% of the points, using the proposed predictive entropy as a rejector.

4.1.1 Relation with calibration methods

As mentioned in section 2.4.1, calibration methods ensure that the output of probabilistic classification systems describes a proper probability distribution. Thus, one can take advantage of these probabilities to not only emit a prediction but also to estimate the confidence of such prediction. Calibration methods, as described in Guo et al. [2017], are similar to the proposed uncertainty wrapper in the sense that they actuate

on already trained models, working on the output of the classifier to deliver a measure of confidence. Nevertheless, some differences make the proposed method to be more generic and applicable.

First of all, in the present work, we proposed a method for evaluating the predictions obtained using a pre-trained black-box system, checking whether the obtained results fit for the current problem or not. It is not the goal of this method to alter the predictions obtained, but to evaluate their appropriateness for the given domain. Calibration, on the other hand, alters the probabilities of the output. This alteration, though, will not affect the results when using those predictions for rejecting unconfident points, as it is the order of this probabilities what matters.

However, the main difference comes from the constraint imposed by black-box systems: we do not have access to the internals of the model. The calibration methods described in Guo et al. [2017] all work at the logits level[3], which prevent them from being applied in the scenario here described. Moreover, as outlined in the results, the proposed method also applies when the output of the black-box system is a hard categorical result and not a probabilities distribution, situation in which the use of calibration is not possible.

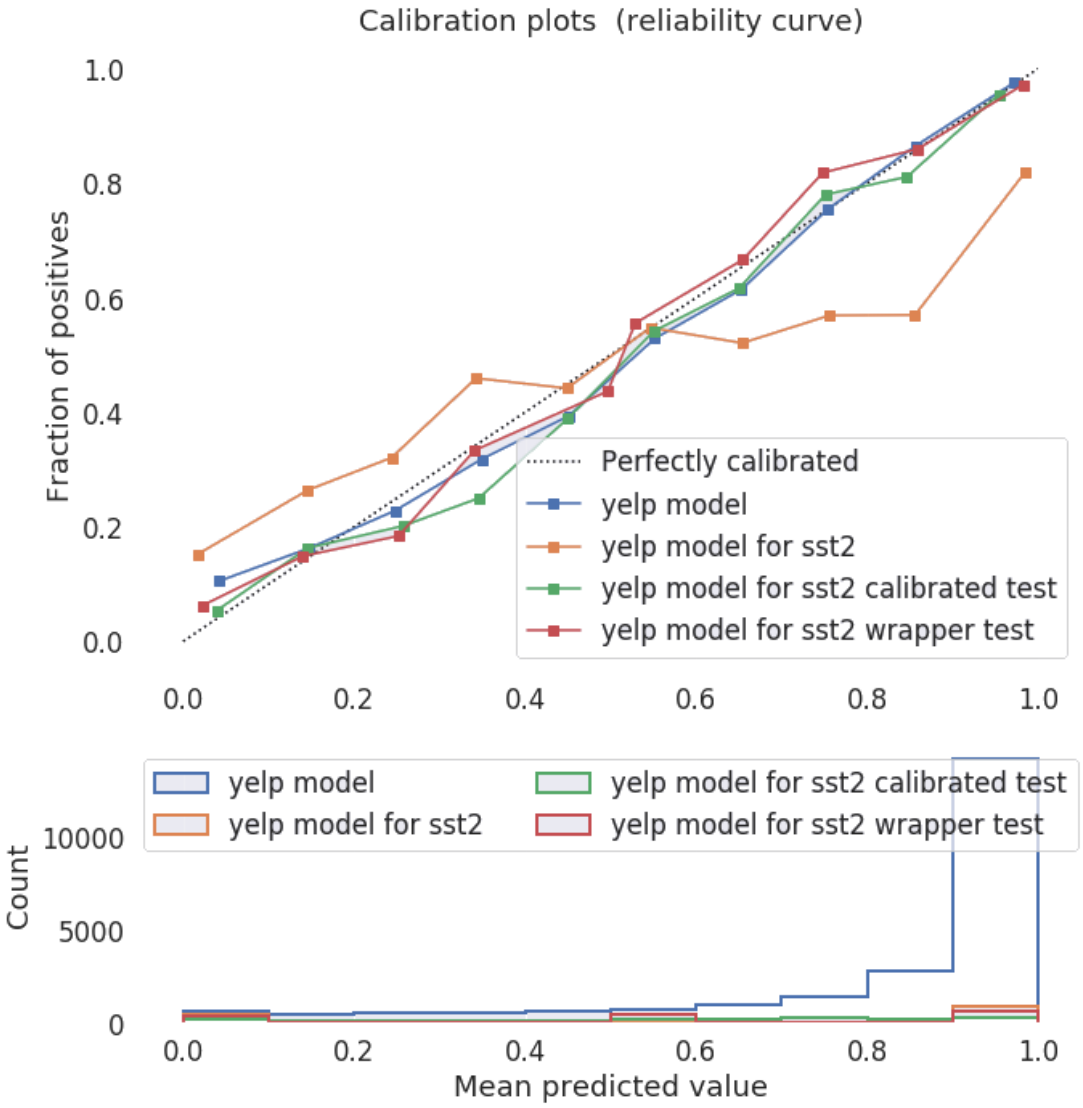
To illustrate the relationship between calibration and the proposed uncertainty wrapper, we take advantage of the simulation of a black-box scenario in the NLP test cases. There, we have access to the logits of the original model, so we can apply a calibration method and observe how it relates to the wrapper.

In the case of training a reviews classifier using Yelp reviews and apply it to classify movie reviews from the SST-2 data set as positive or negative, the first thing that we observe in figure 4.4 is that the model trained with the source dataset is well calibrated. We also notice that when applied to the target domain, the calibration is lost, as expected, as the distributions of both datasets are different. The most relevant result observed in this plot is the fact that the result after applying a recalibration method, Temperature scaling Guo et al. [2017], to the SST-2 predictions are similar to the calibration of the probabilities obtained directly from the wrapper, so the proposed method already yields calibrated predictions.

Table 4.4 shows the Expected Calibration Error, ECE, Naeini et al. [2015], computed for each of the experiments included in the present paper in the following scenarios:

[3]Logit activations are those previous to the last activation function, usually a softmax function in neural networks applied to classification.

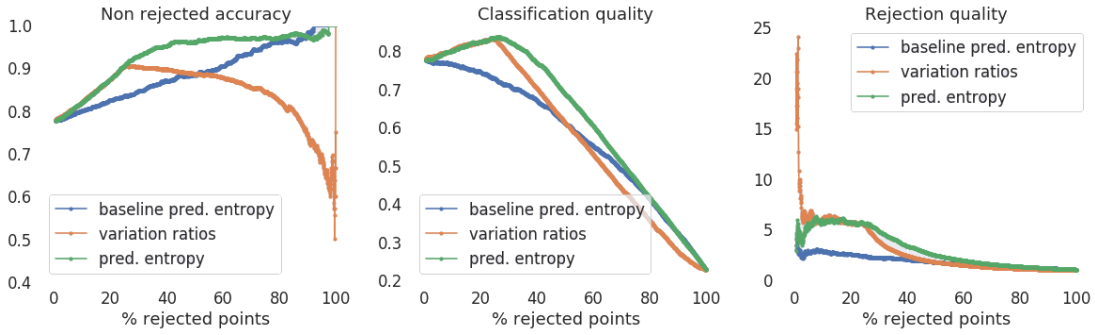
Figure 4.4: Reliability diagram displaying the calibration of the black-box applied to the source, target domain and after calibrating for the target domain, compared to the calibration obtained by the predictions from the wrapper



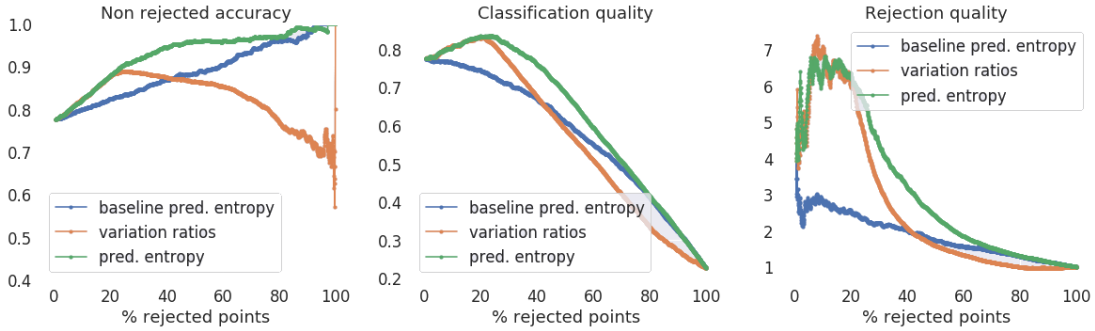
- ECE of the Black-box applied to the original data source, except for the case of image classification, where we do not have access to the initial training set.
- ECE of the Black-box applied to the target data source.
- ECE of a calibrated version of the Black-box using the Temperature scaling method applied to the target data set.
- ECE of the predictions obtained from the wrapper applied to the target data

source by drawing 1000 samples from each Dirichlet distribution.

Results show that predictions obtained using the wrapper are similar to those obtained after applying the calibration or even better, as in the case of image classification.



(a)



(b)

Figure 4.5: (a) Apply SST-2 BB to Yelp with calibrated black-box probabilities for baseline. (b) Apply SST-2 BB to Yelp with calibrated probabilities for training the beta

Next, we validate that the calibration has no direct impact on the outcomes of the wrapper. We use temperature scaling Guo et al. [2017] for calibrating the predictions for the target dataset. We then run two different experiments to evaluate the impact of calibration in the proposed method:

- Compare the wrapper uncertainty with the predictive entropy of the calibrated output. In this experiment, we validate that even with calibrated probabilities, the confidence of those probabilities alone is not enough to capture the uncertainty of overconfident and erroneous predictions. Figure 4.5 (a) displays the rejection performance comparing the proposed uncertainty obtained using the wrapper against the predictive entropy using the calibrated probabilities from the original model.

- Compare the wrapper uncertainty using calibrated and non-calibrated probabilities as input for the modelling of the uncertainty. Figure 4.5 (b) shows the performance of the wrapper trained using the calibrated version of the probabilities.

In both cases, we do not appreciate substantial changes in performance when using and not using calibrated probabilities compared to the uncalibrated version. The only observation that we can remark is the fact that when using the calibrated probabilities as the input for training the wrapper, the training process seems to be smoother than when using the non-calibrated predictions. Moreover, we find that using calibrated probabilities improves the performance of the variation ratios metric in the first stages of the rejection process. Nevertheless, we want to remark that the wrapper itself already has the effect of producing well-calibrated predictions by itself.

4.2 Image Classification

Similar to the NLP case, in computer vision applications, the adaptation of the trained models to new domains is essential. In the literature, one can find different works that tackle the problem using different approaches Chen et al. [2020]; Liang et al. [2019], including the observation of discrepancies in the distributions of both source and target domains or proposing a conditional adaptation network for cross-domain image classification. In our work, we focus on estimating the uncertainty of every single prediction by applying the proposed wrapper and use this uncertainty to discard overconfident and erroneous predictions.

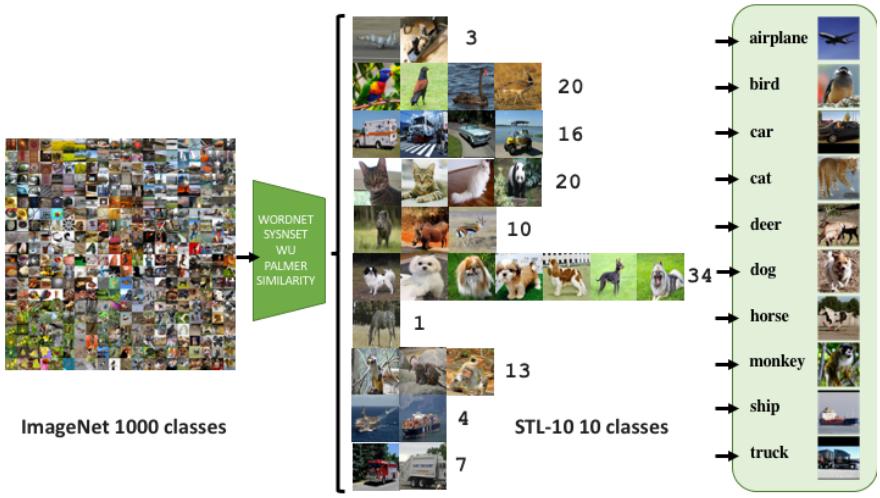
In this use case, we show how the proposed wrapper applies in transfer learning scenarios. Here the simulated API, the reference pre-trained model, are different Keras[4] implementations of a MobileNet.V2 Sandler et al. [2018], DenseNet Huang et al. [2017] and Inception.V3 Szegedy et al. [2016] models, trained for classifying 1000 classes of the ImageNet dataset Deng et al. [2009].

As the target domain we use the STL-10 dataset Coates et al. [2011]. Due to the difference in the output categories(1000 to 10), a mapping between both domains is needed. Figure 4.6 illustrates the mapping process used to adapt the ImageNet pre-trained models to the 10 STL-10 classes. Taking advantage of the fact that each ImageNet label belongs to a SynSet in WordNet, we assign a SynSet for each of the 10

[4] https://keras.io/applications/

STL labels. Next, we use the Wu-Palmer Similarity Wu and Palmer [1994] to find those ImageNet Sysnet that are more similar to the chosen STL ones.

Figure 4.6: Mapping from the 1000 classes of ImageNet to the 10 of STL-10 based on the Wu-Plamer similarity of the ImageNet sysnet as definied in WordNet to the corresponding STL-10 cateogry



Thanks to this mapping method now we can assign ImageNet predictions to the STL-10 labels. For each new example, we use the ImageNet models to obtain the probabilities for each of the 1000 categories. Then we use the mapping to get only the probabilities of the categories mapped with the corresponding STL-10 classes. For each STL-10, we compute the weighted average, considering the number of assigned ImageNet classes for each STL-10 label. In addition to the labels mapping, we needed to rescale the size of STL-10 images, which were 96x96x3, to the ImageNet size, 224x224x3. Later on, we use these images together with the corresponding predictions as the input for the Dirichlet method to estimate the uncertainties.

Finally, we proceed in the same way as done in the Sentiment analysis case, using the uncertainty score to evaluate the performance of different rejection points.

Similarly to the NLP experiments, Figure 4.7 displays the performance metrics of the three rejection measures (*baseline*), *pred. entropy* and *pred. entropy cat.*) for the tests run in the computer vision case.

In this domain, we can see that by only rejecting 10% of the more uncertain predictions, we increase in 8 points the accuracy for the remaining images. This increase goes over 12 points when rejecting 20% of the less confident predictions. In all the cases, the results using the predictive entropy of the wrapper outperform the baseline of using the softmax response of the BB. Observing the values for the classification

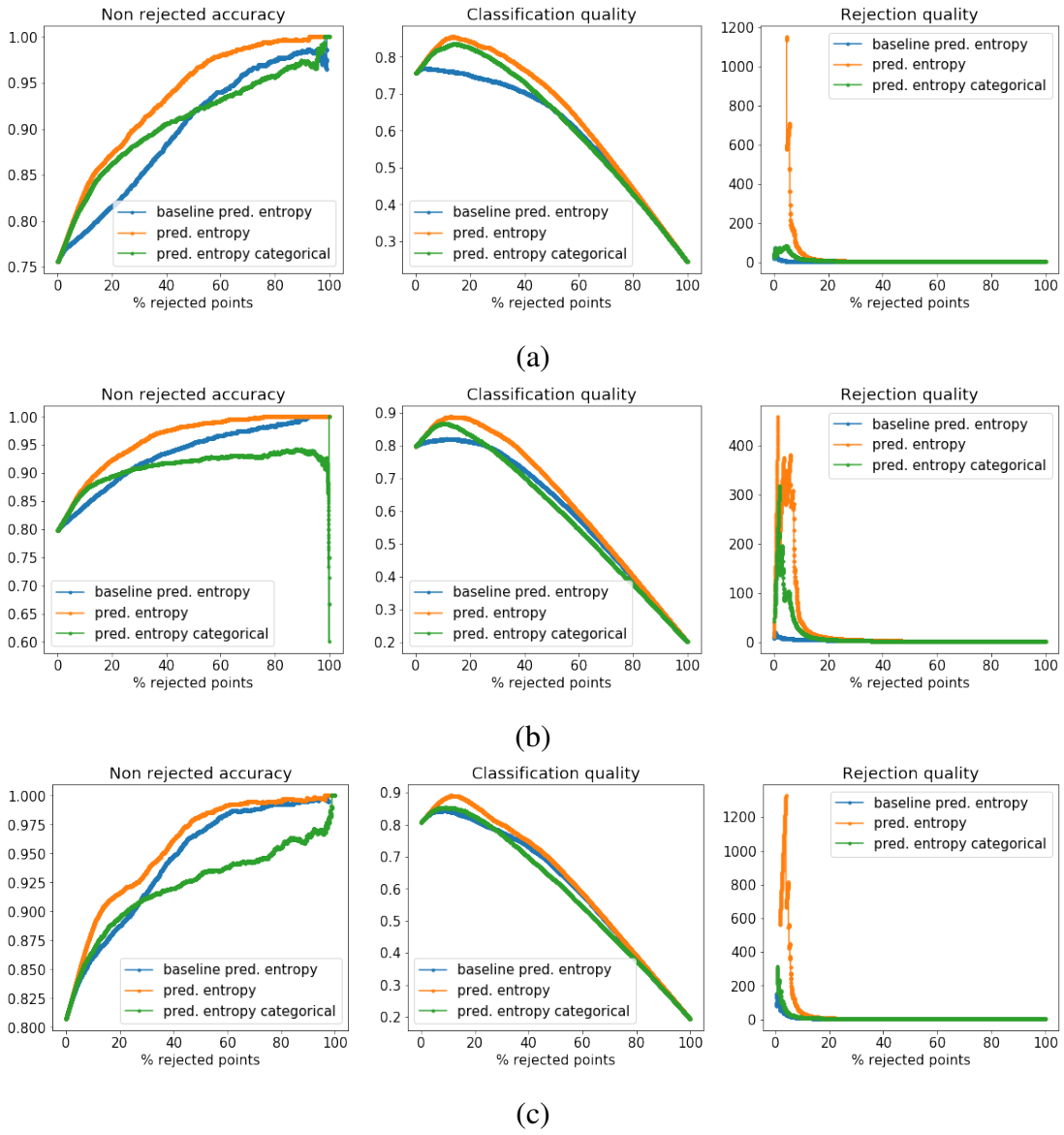


Figure 4.7: (a) Apply ImageNet BB trained with MobileNet V2 model to STL-10; (b) Apply ImageNet BB trained with DenseNet model to STL-10; (c) Apply ImageNet BB trained with Inception V3 model to STL-10

quality, we can determine that it takes its maximum when discarding 20%, indicating that this could be a good rejection point for these cases.

About the categorical version, we observe that it behaves differently depending on the original black-box model. For MobileNet.V2, we see that it outperforms the baseline for the first 40% predictions. For the rest of the models, we observe that this threshold falls to around 30%. This difference might be explained by the fact that during the selection of the model used for implementing the wrapper, we observe

that MobileNet showed the best results. However, as commented for the NLP case, uncertainty still appears as a good rejector, especially if we bear in mind the fact that there is no chance to use the softmax response or similar metrics in such scenarios.

Figure 4.8 illustrates the overall results obtained in the computer vision scenario for both probabilistic and categorical versions of the black-box. In this domain, even though the results of rejection are outstanding, we observe how the accuracy of the prediction obtained from sampling the output from the wrapper, with no rejection, does not improve the results obtained from applying the original black-box. This lack of improvement might be caused by the fact that for this domain, and due to the need for mapping the BB and target labels, the cases of over-confident predictions might be very relevant. For preventing this overconfidence, rejection is a handy tool, as the results show.

	Target BB.	Wrapper prb.	Wrapper prb. 10%	Wrapper prb. 20%	Wrapper prb. 30%
MobileNet V2	75.73	74.76	83.35	87.32	90.55
Inception V3	80.68	80.02	88.57	91.11	93.25
DenseNet	79.81	79.69	87.85	92.5	95.62

	Target BB.	Wrapper cat.	Wrapper cat. 10%	Wrapper cat. 20%	Wrapper cat. 30%
MobileNet V2	75.73	74.62	82.22	86.19	88.78
Inception V3	80.68	80.55	86.99	89.57	91.13
DenseNet	79.81	78.87	86.68	89.29	90.79

Figure 4.8: Results of applying the wrapper to the target STl-10 dataset with different rejection points.

After analysing the results obtained, now it is time to design the rejection mechanism for the classification model. Take, for example, the use case where we are using the ImageNet model to predict STL-10 images. Looking at the values of the classification and rejection quality measures, they show their maximum around 10% of rejected points, (84.67% and 54.32% respectively). Observing the accuracy of the kept examples, 83.54%, we conclude that only by rejecting this 10% we increase the accuracy of the model in almost 8 points.

If we examine the value of the uncertainty score predicted at the selected rejection

point, 2.3024, we can see that this value is quite close to the maximum predictive entropy for ten classes, 2.3026. Thus we can see that the proposed wrapper is taking the more uncertain and erroneous points to the maximum entropy, as we saw in the toy scenario for the wrong labelled class 2 points. Filtering the predictions with uncertainty scores higher than this 2.3024 value, we discarded 798 of the 8000 test data points.

In case we wanted to put this system in production for classifying new STL-10-like images, for those predictions that reached an uncertainty score higher than 2.3024, we could warn the user about the uncertainty of the prediction, or directly discard those examples.

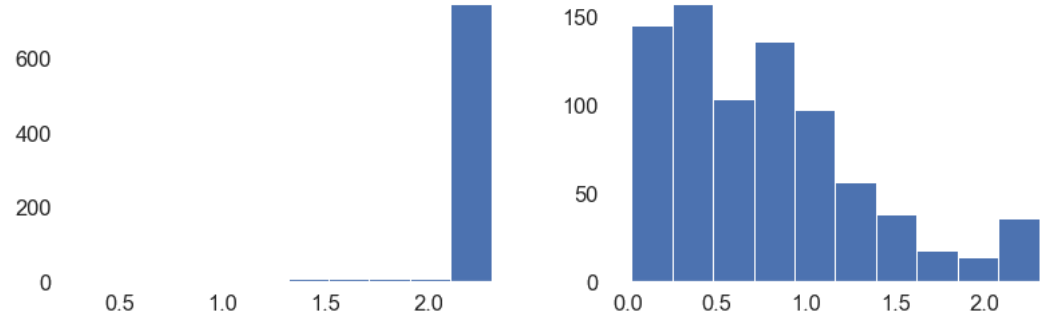
To find the right balance between the number of rejected points and the accuracy of the kept data will depend on each use case, as in some cases we would rather high accuracy no matter the number of discarded points. In contrast, in some other circumstances, the system will have to give a prediction for all the data points. In any case, the proposed method will always be able to indicate a level of confidence for the prediction, leaving it in the hands of the user to make the final decision.

4.2.1 Uncertainty evaluation of unknown classes

During the description of the image classification use case, we mention the necessity of developing a mapping between the 1000 ImageNet labels and the 10 used in STL. By using similarities between the WordNet SynSets that defined the ImageNet categories and those selected for STL, we created a correspondence between each STL category and N matching ImageNet labels.

Although in many cases the mapping can identify clear matchings for each STL-10 category in those predicted in ImageNet, there are some cases where the equivalence is not that evident. In Figure 4.6, we see the number of ImageNet categories that are bound to each STL-10 category. One can see that there are categories well represented, dogs mapped to Japanese spaniel, Maltese dog, Pekinese, Shih-Tzu, up to 34 different categories; others like horses have an inferior representation. In the latter case, the only mapping identified with horses is the zebras. Beyond the biological discussion of their similarities, it is clear that in this case, the level of uncertainty might be high when applying the ImageNet to identify horses, as it shows when looking at the distribution of the uncertainties obtained for the "horses" category displayed in Figure 4.9(a).

Moreover, in classes that exhibit more matches, some of those matches are not very accurate. Take, for example, the deers category. In the WordNet mapping, we



(a) (b)

Figure 4.9: (a) Distribution of uncertainty scores for the "horses", and (b) "deers" categories in STL-10

find similarities with hartebeests, impalas, gazelles, but also with wild boars, warthogs, hippopotamus, water buffalos, bison, Arabian camels or llamas. Figure 4.9 (b) shows the distribution of uncertainties for this class. In this case, we see that the model is more confident than in the case of horses. We conjecture that this may be because we consider the weighted contributions of the mappings, and this is helping to identify deers as a combination of those animals.

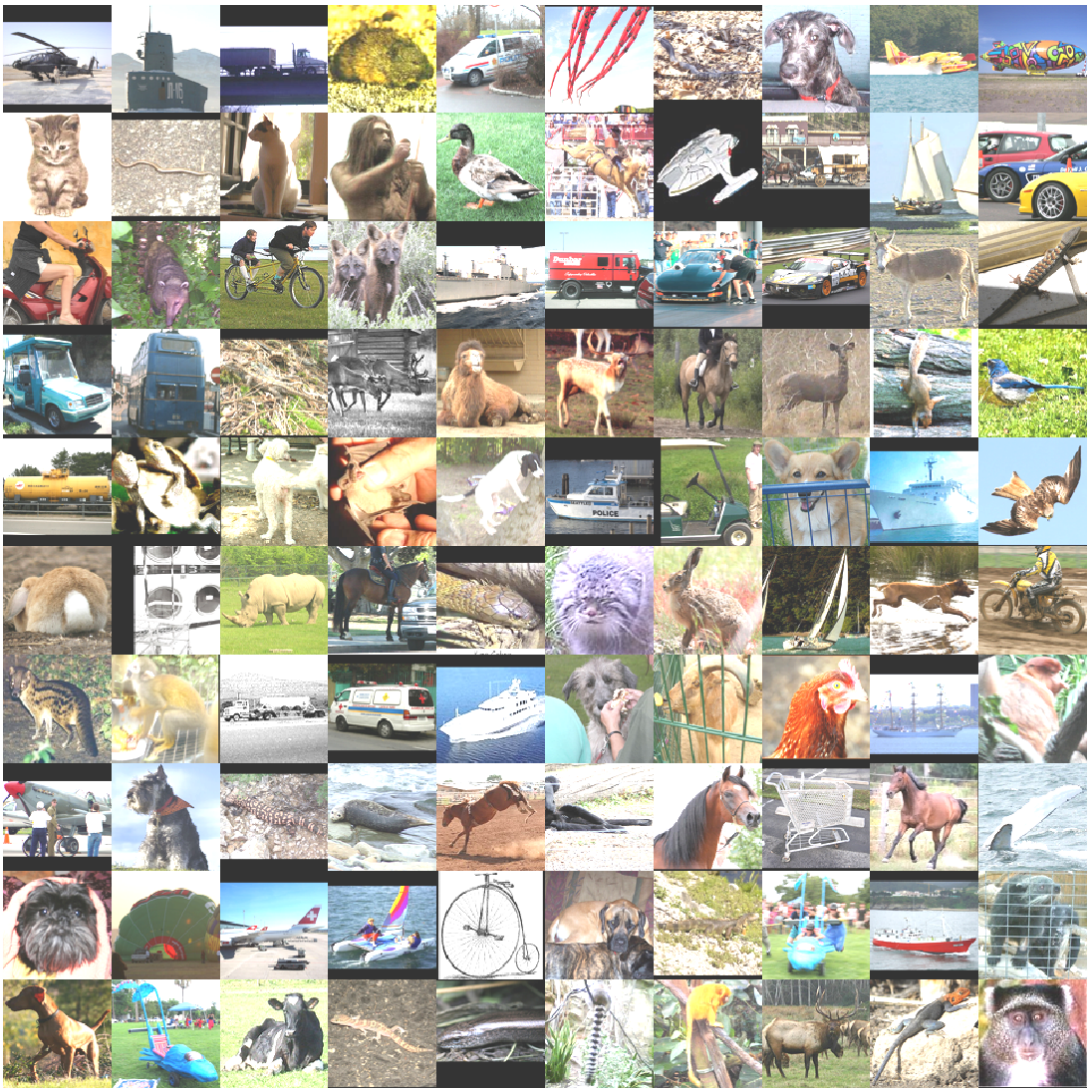
4.2.2 Out of Distribution Image Case

Beyond the uncertainties induced by the possible mismatches in the label mapping, in this section, we also analyse the role of the out-of-distribution term added to the loss function of the wrapper. To validate the out-of-distribution method, we take advantage of the fact that the STL-10 data set incorporates, according to the description, "100000 unlabeled images for unsupervised learning". These examples are extracted from a similar but broader image domain. Figure 4.10 show some examples of these images that contain other types of animals (bears, rabbits, etc.) and vehicles (trains, buses, etc.) in addition to the ones in the labelled set.

In training time, we added 1000 of these unlabeled images to the STL-10 training set for learning the uncertainties, labelling them according to whether they belong to the distribution or not. After training the out-of-distribution wrapper, we take 1000 images more from the unlabeled data set to test the method.

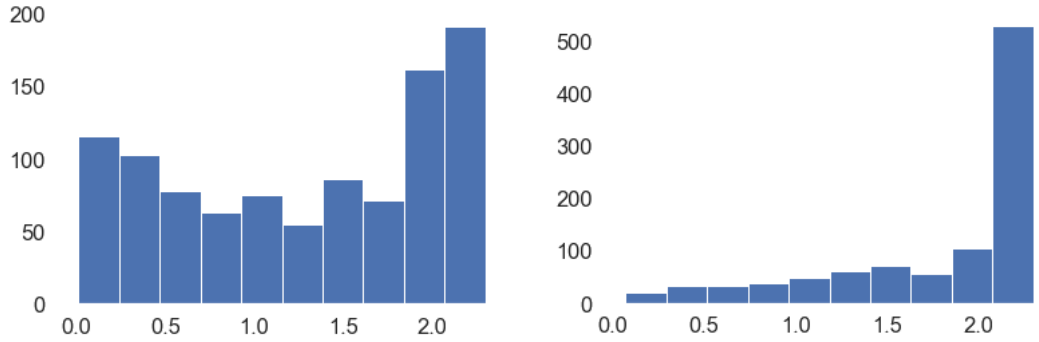
In figure 4.11, we can see the results of applying the original wrapper and the new out-of-distribution version to the test unlabelled images. We can observe how the new version outperforms the previous one detecting the out-of-distribution images

Figure 4.10: Examples of images included in the unlabeled set of STL-10 images



by assigning a high entropy to them. Thus, on prediction time, the out-of-distribution version of the wrapper can detect not only uncertainties associated with noisy labels of images included in the training set but also can protect against the addition of images that are away from the original distribution. By analysing the uncertainty of the predictions, the practitioner will be able to identify such new categories and decide on whether discard them or train a specific model for them.

Moreover, in figure 4.12, we analyse the performance of the new method compared to the original one with regards to the rejection ability when applied to the labelled test images. In this part of the experiment, we use both the original wrapper and the out-of-distribution version to the test set in STL-10 to compute the prediction uncertainty.



(a) Original wrapper (b) Out-of-distribution version

Figure 4.11: Histogram of the distribution of uncertainties of the test unlabeled images from STL-10 obtained using the two wrapper versions

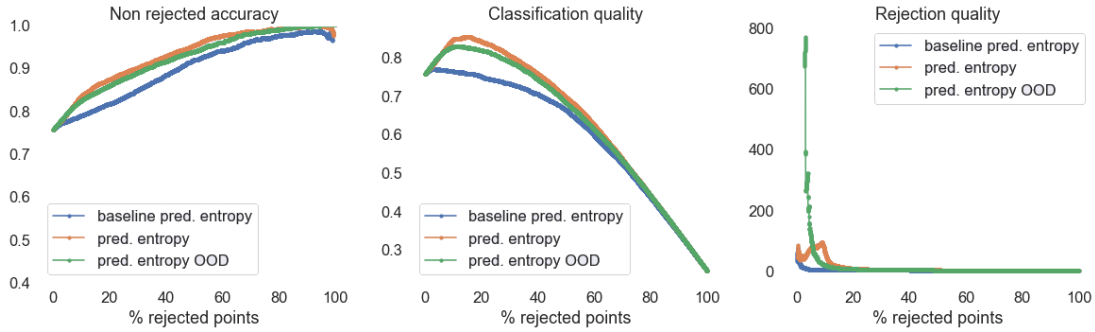


Figure 4.12: Rejection performance metrics comparing the results from applying the softmax response, the predictive entropy of the original wrapper and the OOD version

Next, we use these uncertainties for rejecting uncertain predictions, as in section 4.2. We observe how, despite a small reduction of the rejection performance, the wrapper still retains its ability to detect the uncertainty in erroneous predictions, outperforming the softmax response.

Table 4.2: The accuracy obtained by training a standalone classifier, applying the API and the proposed wrapper for each domain

	BB source acc.	BB target acc.	Wrapper target acc.	Non-reject. acc. (10/20/30%)	Class. quality (10/20/30%)	Reject. quality (10/20/30%)
Apply Yelp BB to SST-2	89.18±0.08%	77.13±0.52%	82.62±0.49%	82.43±0.22% 88.19±0.50% 93.60±0.16%	80.04±0.39% 83.11±0.80% 83.05±0.23%	6.03±0.45 6.04±0.51 4.97±0.07
Apply Yelp BB to Amazon apparel		87.86±0.02%	87.68±0.02%	$91.37\pm4\times10^{-4}$% $94.15\pm2\times10^{-3}$% $95.75\pm3\times10^{-3}$%	$86.18\pm9\times10^{-4}$% $82.38\pm3\times10^{-3}$% $75.82\pm5\times10^{-3}$%	5.25±2.5% 4.18±0.4 3.13±0.2%
Apply Yelp BB to Amazon camera		87.31±0.04%	87.44±0.00%	$91.58\pm7\times10^{-4}$% $94.68\pm2\times10^{-3}$% $96.62\pm1\times10^{-3}$%	$87.09\pm1\times10^{-3}$% $83.07\pm3\times10^{-3}$% $77.57\pm1\times10^{-3}$%	6.66±4.4 4.84±0.4 3.54±0.0
Apply Yelp BB to Amazon computer		84.32±0.02%	85.58±0.00%	$90.24\pm1\times10^{-3}$% $93.61\pm3\times10^{-3}$% $96.24\pm4\times10^{-3}$%	$86.16\pm2\times10^{-3}$% $83.56\pm6\times10^{-3}$% $78.66\pm7\times10^{-3}$%	7.29±6.5 5.03±0.7 3.75±0.3
Apply Yelp BB to Amazon electronics		82.52±0.04%	81.87±0.00%	$85.39\pm1\times10^{-4}$% $88.81\pm1\times10^{-3}$% $92.04\pm2\times10^{-3}$%	$81.06\pm2\times10^{-4}$% $79.46\pm3\times10^{-3}$% $76.21\pm3\times10^{-3}$%	3.54±0.7 3.47±0.2 3.08±0.1
Apply Yelp BB to Amazon health		80.95±0.02%	80.69±0.00%	$83.24\pm6\times10^{-4}$% $86.36\pm7\times10^{-4}$% $88.70\pm1\times10^{-3}$%	$78.35\pm1\times10^{-3}$% $76.74\pm1\times10^{-3}$% $72.78\pm1\times10^{-3}$%	2.59±0.7 2.81±0.0 2.45±0.0
Apply Yelp BB to Amazon kitchen		83.22±0.04%	82.90±0.00%	$86.43\pm4\times10^{-4}$% $89.47\pm2\times10^{-3}$% $92.36\pm2\times10^{-3}$%	$82.15\pm9\times10^{-4}$% $79.75\pm4\times10^{-3}$% $75.91\pm4\times10^{-3}$%	4.03±2.8 3.51±0.3 3.02±0.1
Apply Yelp BB to Amazon magazines		83.14±0.03%	86.70±0.01%	$88.44\pm1\times10^{-3}$% $92.33\pm6\times10^{-3}$% $94.58\pm6\times10^{-3}$%	$85.05\pm3\times10^{-3}$% $83.59\pm1\times10^{-2}$% $78.37\pm1\times10^{-2}$%	7.10±3.2 5.17±1.4 3.62±0.5
Apply Yelp BB to Amazon toys		85.09±0.02%	86.05±0.01%	$88.53\pm5\times10^{-3}$% $91.08\pm1\times10^{-4}$% $94.08\pm1\times10^{-4}$%	$83.74\pm1\times10^{-3}$% $81.27\pm3\times10^{-3}$% $76.15\pm3\times10^{-3}$%	3.03±0.9 4.43±0.2 3.93±0.1
Apply Yelp BB to Amazon videos		85.91±0.03%	87.65±0.00%	$91.75\pm1\times10^{-4}$% $94.65\pm2\times10^{-3}$% $96.26\pm9\times10^{-4}$%	$89.15\pm2\times10^{-4}$% $85.38\pm4\times10^{-3}$% $78.77\pm1\times10^{-3}$%	11.92±0.2 5.78±1.0 3.76±0.0
Apply DVD BB to SST-2	89.18±0.08%	78.31±0.03%	81.13±0.00%	$82.99\pm1\times10^{-3}$% $88.03\pm2\times10^{-3}$% $93.05\pm5\times10^{-3}$%	$80.47\pm2\times10^{-4}$% $81.92\pm4\times10^{-3}$% $81.36\pm8\times10^{-3}$%	5.41±5.6 5.14±0.4 4.41±0.4
Apply DVD BB to apparel	64.87±0.02%	64.73±0.09%	64.38±0.00%	$68.19\pm4\times10^{-4}$% $71.54\pm1\times10^{-3}$% $74.30\pm4\times10^{-4}$%	$67.83\pm8\times10^{-4}$% $69.54\pm2\times10^{-3}$% $69.09\pm5\times10^{-3}$%	3.37±1.8 2.96±0.1 2.45±0.1
Apply DVD BB to camera		54.25±0.12%	53.62±0.00%	$58.21\pm5\times10^{-4}$% $60.38\pm1\times10^{-3}$% $62.19\pm2\times10^{-4}$%	$59.94\pm8\times10^{-4}$% $61.77\pm2\times10^{-3}$% $62.23\pm5\times10^{-3}$%	3.59±1.7 2.47±0.3 2.00±0.1
Apply DVD BB to computer		60.54±0.07%	59.96±0.00%	$66.04\pm2\times10^{-3}$% $69.67\pm5\times10^{-3}$% $71.16\pm6\times10^{-3}$%	$67.56\pm4\times10^{-3}$% $70.09\pm9\times10^{-3}$% $68.28\pm9\times10^{-3}$%	5.76±1.7 3.88±0.3 2.52±0.1
Apply DVD BB to electronics		56.83±0.02%	54.93±0.00%	$59.40\pm1\times10^{-4}$% $62.02\pm4\times10^{-3}$% $64.69\pm1\times10^{-4}$%	$60.06\pm2\times10^{-4}$% $62.37\pm8\times10^{-4}$% $63.70\pm3\times10^{-3}$%	2.54±1.9 2.31±0.0 2.09±0.0
Apply DVD BB to health		59.82±0.03%	58.69±0.00%	$62.61\pm4\times10^{-4}$% $65.76\pm3\times10^{-3}$% $68.23\pm3\times10^{-3}$%	$62.69\pm7\times10^{-4}$% $65.19\pm5\times10^{-4}$% $65.49\pm4\times10^{-3}$%	2.59±0.6 2.53±0.3 2.16±0.1
Apply DVD BB to kitchen		61.28±0.03%	60.41±0.00%	$63.94\pm4\times10^{-4}$% $66.52\pm2\times10^{-3}$% $70.82\pm3\times10^{-3}$%	$63.94\pm7\times10^{-4}$% $66.52\pm5\times10^{-4}$% $67.77\pm4\times10^{-3}$%	2.71±3.8 2.69±0.3 2.45±0.1
Apply DVD BB to magazines		62.79±0.05%	64.16±0.00%	$67.38\pm4\times10^{-4}$% $70.88\pm2\times10^{-3}$% $74.19\pm6\times10^{-4}$%	$68.04\pm7\times10^{-4}$% $70.14\pm4\times10^{-4}$% $70.58\pm1\times10^{-2}$%	4.65±0.3 3.51±0.4 2.82±0.3
Apply DVD BB to toys		60.76±0.03%	60.19±0.00%	$63.83\pm5\times10^{-4}$% $66.67\pm2\times10^{-3}$% $69.32\pm5\times10^{-3}$%	$63.94\pm1\times10^{-4}$% $65.71\pm4\times10^{-4}$% $66.09\pm4\times10^{-2}$%	2.87±2.1 2.53±0.2 2.20±0.1
Apply DVD BB to video		64.98±0.03%	66.49±0.00%	$69.11\pm5\times10^{-4}$% $71.09\pm3\times10^{-3}$% $74.04\pm5\times10^{-3}$%	$69.93\pm1\times10^{-4}$% $70.05\pm7\times10^{-4}$% $68.64\pm8\times10^{-2}$%	4.72±0.1 3.10±0.1 2.37±0.1
Apply Music BB to SST-2	93.08±0.03%	76.65±0.15%	79.71±0.03%	$81.61\pm7\times10^{-4}$% $86.67\pm1\times10^{-3}$% $92.25\pm2\times10^{-3}$%	$79.67\pm2\times10^{-4}$% $81.43\pm5\times10^{-3}$% $81.91\pm7\times10^{-3}$%	5.85±3.3 5.26±0.6 4.65±0.4
Apply Music BB to apparel	71.88±0.01%	63.68±0.32%	63.31±0.03%	$65.53\pm7\times10^{-4}$% $69.73\pm1\times10^{-3}$% $72.62\pm2\times10^{-3}$%	$65.90\pm1\times10^{-4}$% $67.70\pm1\times10^{-3}$% $67.80\pm3\times10^{-3}$%	2.69±1.1 2.61±0.1 2.30±0.0
Apply Music BB to camera		59.96±0.09%	60.35±0.00%	$63.16\pm7\times10^{-4}$% $66.66\pm3\times10^{-3}$% $69.59\pm2\times10^{-3}$%	$63.59\pm1\times10^{-3}$% $66.54\pm6\times10^{-3}$% $67.30\pm3\times10^{-3}$%	3.12±1.3 2.91±0.4 2.45±0.1
Apply Music BB to computer		62.34±0.13%	62.81±0.00%	$68.64\pm2\times10^{-3}$% $72.12\pm3\times10^{-3}$% $74.92\pm4\times10^{-3}$%	$70.30\pm4\times10^{-3}$% $72.11\pm5\times10^{-3}$% $71.60\pm6\times10^{-3}$%	7.71±1.2 4.28±1.2 3.01±0.3
Apply Music BB to electronics		56.64±0.12%	55.17±0.00%	59.35±0.22% 62.34±0.32% 64.77±0.61%	60.46±0.40% 63.36±0.51% 64.29±0.86%	3.05±0.30 2.67±0.15 2.22±0.13
Apply Music BB to health		58.16±0.26%	57.34±0.00%	$61.35\pm1\times10^{-3}$% $64.09\pm2\times10^{-3}$% $66.85\pm3\times10^{-3}$%	$62.10\pm2\times10^{-3}$% $64.19\pm4\times10^{-3}$% $65.23\pm5\times10^{-3}$%	3.02±4.2 2.54±0.3 2.23±0.1
Apply Music BB to kitchen		60.53±0.10%	60.29±0.00%	$63.76\pm2\times10^{-4}$% $66.81\pm3\times10^{-3}$% $70.08\pm4\times10^{-3}$%	$64.16\pm4\times10^{-4}$% $66.29\pm5\times10^{-3}$% $67.50\pm7\times10^{-3}$%	3.25±1.0 2.76±0.4 2.45±0.2
Apply Music BB to magazines		60.96±0.14%	62.06±0.01%	$65.07\pm1\times10^{-3}$% $69.25\pm4\times10^{-3}$% $72.23\pm5\times10^{-3}$%	$65.77\pm2\times10^{-3}$% $69.39\pm7\times10^{-3}$% $69.71\pm8\times10^{-3}$%	3.95±3.6 3.65±0.6 2.79±0.3
Apply Music BB to toys		60.38±0.11%	60.69±0.00%	$63.73\pm6\times10^{-4}$% $66.44\pm2\times10^{-3}$% $69.45\pm4\times10^{-3}$%	$64.15\pm1\times10^{-3}$% $65.73\pm4\times10^{-3}$% $66.65\pm6\times10^{-3}$%	3.21±3.5 2.60±0.8 2.32±0.1
Apply Music BB to video		62.73±0.03%	64.27±0.00%	$67.17\pm1\times10^{-4}$% $70.05\pm2\times10^{-3}$% $72.52\pm7\times10^{-4}$%	$68.13\pm2\times10^{-4}$% $69.32\pm4\times10^{-3}$% $68.76\pm1\times10^{-3}$%	5.62±0.8 3.32±0.3 2.52±0.0

Table 4.3: The accuracy obtained by training a categorical standalone classifier, applying the API and the proposed wrapper for each domain

	BB source acc.	BB target acc.	Wrapper target acc.	Non-reject. acc. (10/20/30%)	Class. quality (10/20/30%)	Reject. quality (10/20/30%)
Apply Yelp BB to SST-2	89.18±0.08%	77.13±0.52%	81.14±0.00%	82.38±2 × 10^{-3}%	80.27±4 × 10^{-3}%	5.87±2.5
				87.89±3 × 10^{-3}%	82.58±6 × 10^{-3}%	5.70±0.4
				93.61±3 × 10^{-3}%	83.01±4 × 10^{-3}%	4.97±0.3
Apply Yelp BB to Amazon apparel		87.86±0.02%	87.60±0.00%	91.01±2 × 10^{-3}%	85.54±4 × 10^{-1}%	4.62±2.5
				94.06±2 × 10^{-3}%	82.24±3 × 10^{-3}%	4.12±0.1
				96.37±1 × 10^{-3}%	76.69±1 × 10^{-3}%	3.34±0.0
Apply Yelp BB to Amazon camera		87.31±0.04%	86.90±0.00%	90.78±4 × 10^{-4}%	85.67±6 × 10^{-4}%	5.03±1.9
				94.79±2 × 10^{-3}%	83.94±4 × 10^{-3}%	4.93±0.5
				97.36±8 × 10^{-3}%	77.60±1 × 10^{-3}%	3.81±0.0
Apply Yelp BB to Amazon computer		84.32±0.02%	85.87±0.00%	90.20±1 × 10^{-3}%	86.09±2 × 10^{-3}%	7.16±4.4
				92.91±3 × 10^{-3}%	82.48±1 × 10^{-3}%	4.55±0.2
				95.92±4 × 10^{-3}%	78.23±2 × 10^{-3}%	3.65±0.1
Apply Yelp BB to Amazon electronics		82.52±0.04%	82.99±0.00%	84.40±1 × 10^{-4}%	79.27±2 × 10^{-4}%	2.44±0.7
				88.22±2 × 10^{-3}%	78.51±4 × 10^{-3}%	3.15±0.2
				92.86±1 × 10^{-3}%	77.37±2 × 10^{-3}%	3.34±0.1
Apply Yelp BB to Amazon health		80.95±0.02%	80.88±0.00%	83.00±8 × 10^{-4}%	77.97±1 × 10^{-3}%	2.40±0.7
				86.34±2 × 10^{-3}%	76.72±3 × 10^{-3}%	2.80±0.2
				89.59±2 × 10^{-3}%	74.02±3 × 10^{-3}%	2.68±0.1
Apply Yelp BB to Amazon kitchen		83.22±0.04%	82.90±0.00%	85.84±4 × 10^{-4}%	81.11±9 × 10^{-3}%	3.41±3.2
				89.30±9 × 10^{-3}%	79.47±1 × 10^{-2}%	3.43±1.4
				92.78±8 × 10^{-3}%	76.49±1 × 10^{-2}%	3.15±0.5
Apply Yelp BB to Amazon magazines		83.14±0.03%	83.35±0.01%	86.03±9 × 10^{-4}%	80.79±1 × 10^{-3}%	3.24±1.4
				90.66±2 × 10^{-3}%	80.98±4 × 10^{-3}%	4.02±0.2
				95.00±3 × 10^{-3}%	78.95±5 × 10^{-3}%	3.76±0.2
Apply Yelp BB to Amazon toys		85.09±0.02%	85.72±0.00%	88.42±7 × 10^{-4}%	83.53±1 × 10^{-3}%	4.31±0.8
				92.20±4 × 10^{-3}%	81.91±8 × 10^{-3}%	4.18±0.1
				94.57±1 × 10^{-3}%	76.82±2 × 10^{-3}%	3.27±0.1
Apply Yelp BB to Amazon videos		85.91±0.03%	88.21±0.00%	90.51±1 × 10^{-4}%	86.91±2 × 10^{-4}%	7.46±4.0
				96.35±2 × 10^{-3}%	88.15±3 × 10^{-3}%	7.64±0.5
				98.29±3 × 10^{-3}%	81.61±5 × 10^{-3}%	4.57±0.4
Apply Music BB to SST-2	89.18±0.08%	78.31±0.03%	80.09±0.00%	81.11±1 × 10^{-3}%	78.79±2 × 10^{-3}%	4.92±0.6
				86.34±3 × 10^{-3}%	80.90±6 × 10^{-3}%	4.98±0.3
				91.69±3 × 10^{-3}%	81.13±6 × 10^{-3}%	4.40±0.1
Apply Music BB to apparel	64.87±0.02%	64.73±0.09%	63.54±0.00%	66.37±4 × 10^{-4}%	65.61±8 × 10^{-4}%	2.55±0.8
				68.91±2 × 10^{-3}%	66.39±3 × 10^{-3}%	2.29±0.1
				72.07±2 × 10^{-3}%	67.03±4 × 10^{-3}%	2.18±0.1
Apply Music BB to camera		54.25±0.12%	59.48±0.00%	62.66±4 × 10^{-3}%	62.69±7 × 10^{-3}%	2.59±0.4
				66.38±6 × 10^{-3}%	66.09±1 × 10^{-2}%	2.80±0.3
				70.45±8 × 10^{-3}%	68.51±1 × 10^{-2}%	2.67±0.2
Apply Music BB to computer		60.54±0.07%	62.37±0.00%	65.88±2 × 10^{-3}%	65.47±4 × 10^{-3}%	2.83±0.5
				69.58±6 × 10^{-3}%	68.14±1 × 10^{-2}%	2.84±0.2
				72.48±5 × 10^{-3}%	62.88±8 × 10^{-3}%	2.41±0.1
Apply Music BB to electronics		56.83±0.02%	55.83±0.00%	58.76±1 × 10^{-3}%	59.38±1 × 10^{-3}%	2.39±0.1
				61.43±3 × 10^{-3}%	61.90±5 × 10^{-3}%	2.27±0.0
				64.18±3 × 10^{-3}%	63.47±6 × 10^{-3}%	2.09±0.0
Apply Music BB to health		59.82±0.03%	58.11±0.00%	60.51±1 × 10^{-3}%	60.60±4 × 10^{-4}%	2.21±0.1
				63.17±3 × 10^{-3}%	62.73±4 × 10^{-3}%	2.18±0.1
				65.80±2 × 10^{-3}%	63.78±3 × 10^{-3}%	2.01±0.1
Apply Music BB to kitchen		61.28±0.03%	59.92±0.00%	63.45±9 × 10^{-4}%	63.61±1 × 10^{-3}%	2.86±0.0
				66.62±3 × 10^{-3}%	65.99±5 × 10^{-3}%	2.67±0.1
				70.03±1 × 10^{-3}%	67.43±2 × 10^{-3}%	2.54±0.0
Apply Music BB to magazines		62.79±0.05%	60.90±0.00%	64.42±3 × 10^{-3}%	64.62±6 × 10^{-3}%	3.09±0.3
				67.80±7 × 10^{-3}%	67.10±1 × 10^{-2}%	2.86±0.3
				71.98±9 × 10^{-3}%	69.36±1 × 10^{-2}%	2.73±0.2
Apply Music BB to toys		60.76±0.03%	60.28±0.00%	63.43±4 × 10^{-4}%	63.61±6 × 10^{-3}%	2.87±0.3
				66.45±3 × 10^{-3}%	65.74±2 × 10^{-3}%	2.59±0.1
				69.31±1 × 10^{-3}%	66.45±5 × 10^{-3}%	2.28±0.0
Apply Music BB to video		64.98±0.03%	63.71±0.00%	67.40±5 × 10^{-5}%	68.55±3 × 10^{-3}%	6.32±0.7
				71.25±2 × 10^{-3}%	71.23±7 × 10^{-3}%	4.17±0.3
				80.51±4 × 10^{-3}%	73.28±8 × 10^{-2}%	3.51±0.2
Apply DVD BB to SST-2	93.08±0.03%	76.65±0.15%	80.91±0.03%	82.76±2 × 10^{-3}%	80.07±5 × 10^{-3}%	5.02±0.4
				87.76±5 × 10^{-3}%	81.50±9 × 10^{-3}%	4.94±0.4
				92.59±5 × 10^{-3}%	80.71±7 × 10^{-3}%	4.22±0.1
Apply DVD BB to apparel	71.88±0.01%	63.68±0.32%	64.35±0.03%	67.17±1 × 10^{-4}%	66.00±2 × 10^{-4}%	2.34±0.1
				70.16±3 × 10^{-3}%	67.34±4 × 10^{-3}%	2.37±0.1
				73.24±4 × 10^{-3}%	67.61±5 × 10^{-3}%	2.21±0.0
Apply DVD BB to camera		54.74±0.09%	54.67±0.00%	57.40±1 × 10^{-3}%	58.50±2 × 10^{-3}%	2.55±0.1
				60.16±2 × 10^{-3}%	61.43±3 × 10^{-3}%	2.38±0.1
				63.46±4 × 10^{-3}%	63.99±4 × 10^{-3}%	2.26±0.0
Apply DVD BB to computer		62.34±0.13%	60.43±0.00%	64.77±2 × 10^{-3}%	65.33±6 × 10^{-3}%	3.49±0.4
				68.00±3 × 10^{-3}%	67.49±9 × 10^{-3}%	2.95±0.2
				70.74±4 × 10^{-3}%	67.71±1 × 10^{-2}%	2.43±0.1
Apply DVD BB to electronics		56.64±0.12%	56.14±0.00%	59.24±7 × 10^{-4}%	59.77±1 × 10^{-3}%	2.39±0.1
				61.56±2 × 10^{-3}%	61.63±4 × 10^{-3}%	2.14±0.1
				64.47±5 × 10^{-3}%	63.38±7 × 10^{-3}%	2.05±0.1
Apply DVD BB to health		58.16±0.26%	58.94±0.00%	61.98±2 × 10^{-3}%	61.56±3 × 10^{-3}%	2.07±0.1
				64.48±3 × 10^{-3}%	63.16±5 × 10^{-3}%	2.06±0.1
				67.14±6 × 10^{-3}%	63.98±9 × 10^{-3}%	1.96±0.1
Apply DVD BB to kitchen		60.53±0.10%	60.79±0.00%	64.14±1 × 10^{-3}%	64.10±3 × 10^{-3}%	2.78±0.1
				67.31±3 × 10^{-3}%	66.33±5 × 10^{-3}%	2.64±0.1
				70.51±4 × 10^{-3}%	67.35±6 × 10^{-3}%	2.38±0.0
Apply DVD BB to magazines		60.96±0.14%	62.60±0.01%	66.97±3 × 10^{-3}%	67.31±1 × 10^{-3}%	3.99±0.4
				70.90±3 × 10^{-3}%	70.17±6 × 10^{-3}%	3.50±0.5
				74.23±1 × 10^{-3}%	70.65±4 × 10^{-3}%	2.83±0.1
Apply DVD BB to toys		60.38±0.11%	60.15±0.00%	63.67±3 × 10^{-3}%	63.67±5 × 10^{-3}%	2.73±0.3
				66.30±3 × 10^{-3}%	65.12±4 × 10^{-3}%	2.38±0.1
				68.61±4 × 10^{-3}%	65.10±6 × 10^{-3}%	2.06±0.1
Apply DVD BB to video		62.73±0.03%	64.98±0.00%	69.71±5 × 10^{-4}%	70.45±1 × 10^{-3}%	6.23±0.1
				73.40±2 × 10^{-3}%	72.41±3 × 10^{-3}%	4.03±0.1
				77.75±2 × 10^{-3}%	73.82±3 × 10^{-3}%	3.40±0.1

Table 4.4: ECE scores for the different combinations of black-boxes and target applications

	BB applied to source	**BB applied to target**	**Calibrated BB to target**	**Wrapper to target**
Apply Yelp BB to SST-2	0.0104067	0.150570	0.032845	0.039284
Apply Music BB to Electronics	0.0165734	0.035458	0.035698	0.055868
Apply ImageNet to STL-10	N/A	0.612243	0.206683	0.092530

Chapter 5

A real-life use case: Overqualification analysis.

Contents

In this chapter, we present a practical application of the proposed uncertainty wrapper, applied in this case at a classification system for detecting overqualification in job adverts.

5.1 Context of the project

The project that frames the task of overqualification detection is BDFP, Big Data for Vocational and Educational Training (VET [1], FP in Catalan). The project was an initiative of Eurecat in the context of the Big Data Center of Excellence in Barcelona[2],

[1] training for a specific occupation through a combination of theoretical teaching and practical experience

[2] https://www.bigdatabcn.com/

a public-private partnership aiming to assist and boost a *data culture* at all types of organisations operating in Catalonia.

Big Data for VET aimed to analyse the role of Vocational and Educational Training (VET) as one of the critical factors for current societal challenges. The work proposed a data-driven approach with a combination of different data analysis algorithms including data mining, rule-based systems and machine learning. The goal was to analyse the evolution of the labour market (extracted by the free text job vacancies posted in the leading job portal) and the VET offering (extracted by the official training curricula in the whole territory) in Catalonia during the last few years. Special attention was devoted to two main strategic market sectors for the region, namely ICT and Industry 4.0, and the transformation of the particular skill set demanded by employers.

The main objectives of the project were two-fold:

- Provide valuable insights about the evolution of the skill-sets offered in the VET contents in all Catalan territory as well as the ones demanded by the job market during the same period, 2015-2018. This information is of great value for the Catalan Education Department in charge of designing the curricula of the VET courses trying to satisfy the local job market near future requirements.

- Showcase the need and benefits of joining efforts by different institutions to solve societal challenges, with different knowledge and assets, ranging from datasets, data science experts, technology providers, domain experts, decision-makers and facilitators.

Different meetings happened among the stakeholders of the project, after which those two objectives translated to five particular research topics, namely:

- T1: Geographic characterisation of the labour demand, the evolution of the labour demand based on the contracts registered in Catalonia during the last five years.

- T2: Analysis of the relationship between Dual VET and sectorial labour demand, a particular study case of the type of VET training includes an internship in a company.

- T3: Temporal evolution of the VET skills, the evolution of the demanded kills focusing on ICT and Industry 4.0.

- T4: Comparison of labour supply and demand in ICT and Industry 4.0,

- T5: Identification of overqualification, required in a job post concerning the responsibilities described.

During the execution of the project, we compiled data from both the description of VET studies, occupations and job demands. To be able to compare supply and demand, we selected to link both worlds by the skills acquired during the training and required by companies. In the case of job advertisements, we worked with millions of them, making it necessary to develop automated tagging mechanism to deal with such amount of unstructured data. We develop this automatic tagging leveraging on ML classification models that automatically tag the skills required for each job description. Figure 5.1 (a) illustrate the type of models used for the skills classifier.

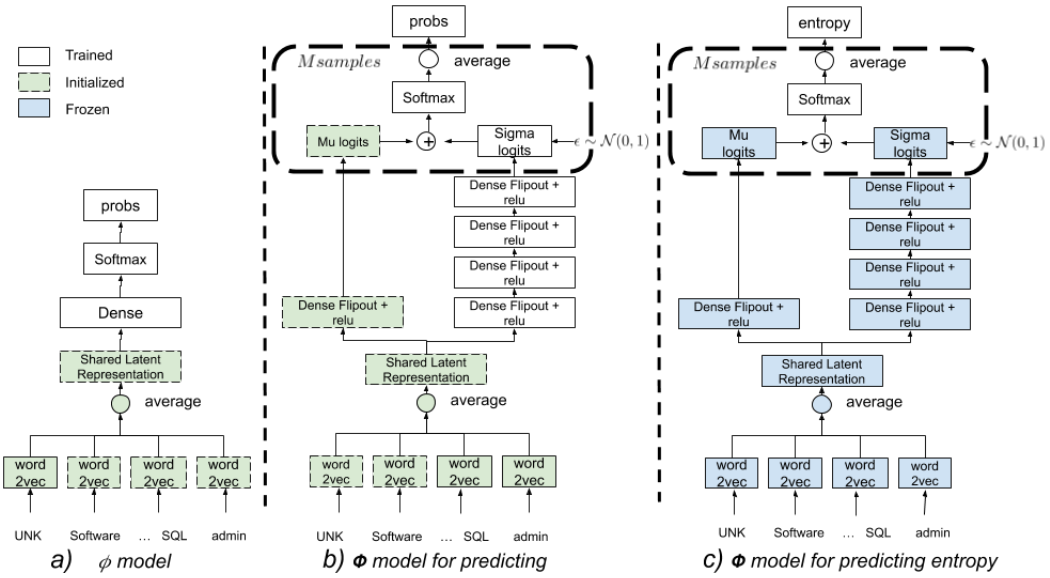


Figure 5.1: Models used in the project. (a) Deep learning model used for challenge 1, (b) Uncertainty Bayesian wrapper for deep black-box models used in challenge 2.

To develop those ML models needed having access to labelled job offers were experts manually tagged the skills demanded. The effort required for labelling and the limited access to experts prevented the inclusion of all the skills defined in a preliminary analysis, so, finally, the study of the evolution of skill demand and the matching with VET studies was limited to those skills more asked by companies.

5.2 Task 4: Identification of overqualification

During the labelling process, we took advantage of having access to experts to ask them for an additional labelling task. In this case, while identifying the skills demanded in a

job vacant, we asked them also to tell whether the description of the work suggested an overqualification issue or not. In many cases, we have detected that the level of studies required in a job description was higher than what was actually needed to fulfil the position. The overqualification issues seemed to be especially relevant in the case of VET and high school engineering curriculums in both ICT and Industry 4.0 positions.

Similarly to the case of detecting the skills present in the job offers, in the case of detecting overqualification in demand, it requires to deal with unstructured textual information, which motivates us to use advanced NLP models.

The model used in this case was the same used for identifying skills, shown in Figure 5.1(a). The input for the models is the description of the job, including different attributes like the level of education, minimum requirements, description of the position, title or previous knowledge. In order to fight the lack of positive labels for some of the skills, we try different combinations of those fields for building different samples out of the same labelled job. We also combine samples obtained from the labelling tool with some from a rule-based model to obtain balanced datasets. The resulting text for each job position is transformed into a sequence of words with a limited length of 120 words, using left padding for shorter texts.

We used an artificial neural network, where the first layer of the model is an embedding layer. In this work, we use a pre-trained Word2vec embedding Mikolov et al. [2013a] trained with a billion of Spanish words Cardellino [2019]. We average the embeddings of the words included in the job description to obtain a latent representation for it. This latent representation is the input for a fully connected layer with a softmax activation, used for obtaining the probabilities of each class predicted. For some of the classifiers, we substitute the average of the embeddings for an LSTM layer plus a hidden dense layer, showing better accuracy scores. Sometimes, though, this leads to overfitting, despite having 50% of dropout, probably due to the lack of sufficient data samples, so we have to stick to the original model.

By using this model, the prediction accuracy only reaches 72.28%. This value, though acceptable in some contexts, is inadequate for a careful analysis of the problem. Our hypothesis for justifying this value considers the difficulty of defining the concept of over-qualification and the subjectivity involved in those definitions. There are different levels of overqualification. For example, they are asking a university degree for a job that a VET student can perform, or asking for a high level of VET when it can be carried out by medium degree VET graduates.

These two different properties introduce noise in the labelling process, and con-

sequently, affect the performance of the machine learning classifiers. In this setting, the degree of confidence in the prediction can help to refine the results, making this problem very suitable for modelling uncertainty using the wrapper method proposed in this thesis.

5.3 Computing the uncertainty score

The method used in this use case applied for the two terms in equation 2.5: the one that depends on the application of the model to the input data, the aleatoric component, and a second one that is measuring how the model may vary depending on the training data, the epistemic component.

For the epistemic uncertainty, we model the weights as random variables by introducing Gaussian perturbations and Flipout as introduced in Wen et al. [2018], to estimate the conditioned probability by using Montecarlo (MC) sampling. The model trained up to this model is a deterministic model, except for the epistemic components. As such, it does not allow to infer the aleatoric component of the uncertainty.

Moreover, for computing the aleatoric uncertainty, we leverage on the wrapper described in section 3.1. While the wrapper output will stand for the mean value of that distribution, we will let the assistance NN to work out the standard deviation corresponding to each input sample. As in the former case, we use Montecarlo approximations to sample from the latent layer and approximate the output distribution.

As a result of both Montecarlo samplings, we obtained a probability density function of the output values that served for computing the overall uncertainty.

It is remarkable that just by training the model with the new architecture for 20 epochs(compared to the 500 used for training the original model), the resulting accuracy increases up to 84,25%.

Model in Figure 5.1(c) depicts the model used to predict the uncertainty score. In this case, we used predictive entropy as defined in Gal [2016]. The predictive entropy, introduced by Shannon [1948], measures the dispersion of the predictions around the mode. In this case, we combine the computation of the aleatoric and epistemic uncertainty. Thus it is necessary to combine the random variables that learned the variability of the model, the epistemic component, together with the random variables assigned to the output logits that model the aleatoric uncertainty.

In order to obtain the uncertainty score, we carry out two consecutive Montecarlo simulations: first, we sample a model $\mathbf{W}$. We then use this model to sample the output

prediction for each logit, following the distribution parameterised by the outputs of the ANN and the assistance NN. Using the resulting probabilities, we compute the predictive entropy as follows,

$$\mathbb{H}[\mathbf{y}^*|\mathbf{x}^*, D_{train}] := -\sum_c \mathbb{E}(\mathbf{y}^*) \log \mathbb{E}(\mathbf{y}^*) \tag{5.1}$$

For obtaining the uncertainty score of the overqualification classifier, we sample 10 models and for each model 100 probabilities.

5.4 Uncertainty-based rejection classifier

Despite the increment in accuracy obtained after training the ANN with uncertainty, we could further exploit the uncertainty scored to improve the results by filtering out the less confident predictions. As a result, the quality of the filtered predictions obtained increases as it only outputs confident predictions. Then, we used this score as a rejection mechanism for the classification.

The test dataset was sorted by the rejection value, from higher to lower scores. If there were a correlation between the uncertainty score and the misclassification value of the input, by discarding the more uncertain data points, we would increase the performance of the classification system. We considered different rejection points corresponding to descending values of the rejector from including all points to discard all of them in the last iteration. The analysis of the three performance measures Condessa et al. [2015] for each rejection point considered a trade-off between the number of samples rejected and the quality of the classifications.

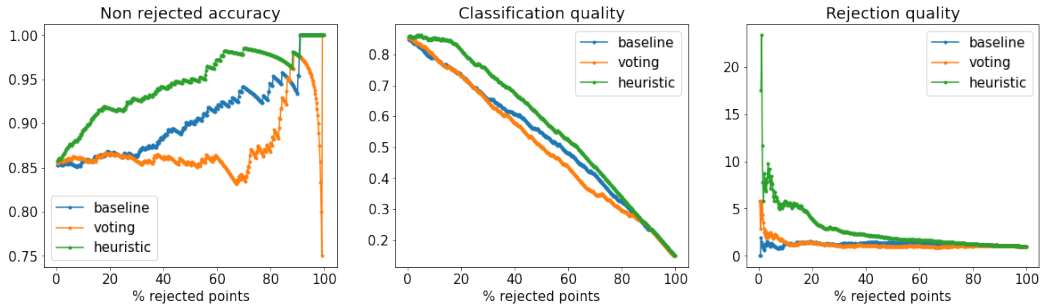


Figure 5.2: Performance measures showing the accuracy of kept points, how correct predictions are kept and wrong discarded, and the ability of rejecting wrong samples.

Figure 5.2 compares the results obtained after applying three different models for computing uncertainty: predictive entropy for the aleatoric model, as proposed in this

work, the predictive entropy of the original model, used as a baseline, and an additional uncertainty score, variation ratios, as described in Gal [2016]. The plot shows how only by rejecting the 10% of predictions with higher uncertainty scores, the model increases its performance up to 88% — reaching 92% prediction accuracy when rejecting the 20% more uncertain predictions.

5.5 Adequacy between required functions/skills and the level of studies

Finally, the research topic T5: Identification of overqualification was answered based on the results obtained from the uncertainty-based rejection method described in section 5.3. Taking the 222,957 job posts that corresponded to ICT and Industry 4.0 positions, we applied a rejection ratio that, on test time, reached a 92.5% of accuracy. Thus, we discarded 137,157 job descriptions, focusing the study on the remaining 85,800. From those, we estimated only 17,388 as including overqualification in the job position.

The analysis of those overqualified job descriptions resulted in 19% overqualification in ICTs and 24% in IND4.0. The trend of under qualification in ICTs is decreasing, while in IND4.0 it is increasing. The "computing and communications" category has the highest number of overqualified ads. We note that the programming subcategory is the most affected with 4,300 cases followed by "analysis", "ERP", "Project Management" and "Systems" that are around 800. In general, those categories with a higher rate of overqualification, coincide with the most demanded ones.

The "engineering and technical" category is not only the one with the most overqualified ads in the industrial area, but its presence in such ads is 40% higher than in the total Industry 4.0 ads. A similar case occurs in the category of Quality and R + D production, whose weight in ads with overqualification increases by 8%. Regarding the "Engineering and Technical" subcategories, the industrial category is the most prone to overqualification. With 2,240 cases, it presents 33% more overqualification than Industry 4.0 at a whole. Next comes "electronic and industrial automatisation" with 941, "other engineering" with 393 and "electronics" with 316. In these cases, the demands of employment request higher degrees to develop skills included in VET. At the other extreme, the Professions, Arts and Crafts category reduces its presence by 35% in ads with overqualification, which tells us that, in general, these types of ads are better made

and better match their requirements.

5.6 Lessons learned

The overqualification detection use case was appealing for the development of the present work, as it presented an opportunity for showcasing the methods proposed in a different setting from the black-box scenario. In this case, the combination of both epistemic and aleatoric uncertainty showed astonishing results in a severe problem such as this one.

The definition of the problem itself makes it very prone to subjective opinions on what is and what is not overqualification. This subjectiveness led the labelling process to introduce noise in the training data set. Besides, the scarcity of job descriptions annotated, made it necessary to have a method that filter out uncertain and noisy examples, enabling the posterior analysis of the overqualification to focus on compelling examples only.

Chapter 6

Unsupervised, Uncertainty-based Text Readability Assessment

Contents

In this chapter, we present another application of the uncertainty wrapper proposed in this thesis, which is the assessment of readability in texts. The appraisal of text readability is an open problem. There is not, and will not be, a unique model of readability, and therefore, it is necessary to develop methods that allow assessing multiple models. This application presents an unsupervised method that leverages a set of traditional readability scores to train a Deep Learning model capable of estimating a readability score for a text together with the uncertainty of this prediction. By applying explainability methods, we analyse the uncertainty found and attribute the words that contributed more to it. We run experiments with texts of different levels of complexity and demonstrate that, by modifying these words, we can increase their readability.

In the context of the new regulations of data privacy like the European General Data Protection Regulation (GDPR), the importance of legal texts like privacy policies or consent forms is vital to communicate to the end-users about their rights. A good

example is the SMOOTH project[1], that aims to assist micro-enterprises on the assessment of GDPR. In SMOOTH, one of the key activities is the analysis of the readability of this type of texts, to ensure that final users understand how their data is handled and what rights they do have over it, and how to exercise them.

Usually, the level of complexity of a text is measured using readability indexes that analyse the text and look at things like the number of words, their length or other syntactic aspects. Alternatively, Machine Learning (ML) models are trained to predict the complexity of texts using a labelled text corpus based on the criteria of domain experts that tag each text depending on its difficulty.

Both approaches have significant limitations. In the first case, Redish [1981], the type of measures used usually does not include semantic aspects of the words. Also, due to many readability formulas, one can get wide variations in results of the same text. On the other hand, the ML approach requires a meaningful amount of labelled texts to train the model, which may prevent its applicability to some scenarios with scarce resources.

In this case, we opt for a combination of both. First, we consider a set of existing readability indexes that we use to label texts according to their reading complexity. Next, we use a Deep Learning (DL) Wrapper to estimate the uncertainty of these predictions. The wrapper learns this uncertainty from two sources: the potential disagreement between the distinct readability scores, and the uncertainty inherent in the text itself.

Later on, we apply explainability techniques to identify the words that contribute more to the uncertainty score predicted. By examining the results obtained for texts of high complexity, we observe how by changing those words in the original text, we can reduce its complexity and improve the overall readability.

6.1 Readability assessment

In the literature, we can find different readability scores. Following the seminal work defined in the Dale-Chall Readability Score Dale and Chall [1948], we find the result of studies carried out in the U.S. navy Senter and Smith [1967]; Kincaid et al. [1975]. In those works, they provided diverse readability scores: Automated Readability Index, the Flesch–Kincaid readability tests or the Flesch-Kincaid Grade Level. These scores included different formulas that combine the number of words, sentences and syllables

[1] https://smoothplatform.eu/

to map a given text to the required grade level. Similar formulas were developed in different studies like the Gunning fog index Gunning [1968], or variations like the SMOG index Mc Laughlin [1969], the Coleman-Liau Index Coleman and Liau [1975], and the Linsear Write Formula Klare [1974].

Alternatively, some works leverage machine learning models such as SVM, Schwarm and Ostendorf [2005]; Petersen and Ostendorf [2009], or KNN, vor der Brück et al. [2008] to predict the reading ease of texts. Other works propose more sophisticated features than those used in the "classic" approaches, like in Feng et al. [2009]; Štajner et al. [2012], based on cognitive or linguistic models. More recent models propose the use of Knowledge Graphs, Štajner and Hulpuş [2018]. All in all, these approaches are based on some existing corpus of labelled texts, like articles from the Encyclopedia Britannica Barzilay and Elhadad [2003], Weekly Reader [2], or OneStopEnglish corpus, Vajjala and Lučić [2018], among others.

Even though ML approaches seem to be more effective than "classic" formulas, François and Miltsakaki [2012], they require the non-trivial effort of labelling texts. Still, they do not improve on the explainability of these models. Besides, the use of generic models in more specific use cases might induce domain shifting issues as described in Elsahar and Gallé [2019].

6.2 Explainability

One of the goals of this application is to provide explanations to the predicted readability score for a text, and how to attribute it to the words that compose it. In the case of model explainability, the goal is to be able to produce an explanation of the output in human terms, usually by proposing a simplified version of the model. For example, LIME Ribeiro et al. [2016a], approximates a simpler model that locally explains each prediction. Other methods, like DeepLIFT Shrikumar et al. [2017] or Layer-wise relevance propagation Bach et al. [2015], focus on DL models and analyze the changes in the gradients to study the relevance of the different input features. SHAP Lundberg and Lee [2017] proposes a generalization for some of these methods based on the fact that all of them share an additive structure in the function applied to the feature attribution based on a linear function of binary variables.

In this article, we apply SHAP to produce an explanation of the prediction uncertainty. By applying the DeepExplainer method and the visualization tools provided in

[2] http://www.weeklyreader.com

Lundberg et al. [2018], we obtain an explanation for how each feature, words, in this case, contributes to the estimated uncertainty.

6.3 Method

Building the dataset. In this case, we do not rely on a human-annotated data source. Instead, our ground-truth is a consensus between the readability metrics studied. We then consider each readability score as a black-box classifier that emits a prediction, the uncertainty of which we want to estimate. Here, the model input consists of sentences, x, and the different readability score functions, $f_r(x)$, that predict a category of complexity, y^*, where $y^* = f_r(x)$.

Here, y will take one of the values defined in The Flesch Reading Ease formula, as shown in Table 6.1. We use the readability score functions present in the library Texstats [3], i.e. Flesch Reading Ease formula, Flesch-Kincaid Grade Level, Fog Scale, SMOG Index, Automated Readability Index, The Coleman-Liau Index, Linsear Write Formula and Dale-Chall. For the ground-truth, we use the consensus score included in the same library as "the estimated school grade level required to understand the text".

Table 6.1: Mapping of grade levels and Reading ease

School level	Reading Ease	Class
$<$ 5th grade	Very easy to read.	0
6th grade	Easy to read.	1
7th grade	Fairly easy to read.	2
8th & 9th grade	Plain English.	3
10th to 12th grade	Fairly difficult to read.	4
College (13th to 15th)	Difficult to read.	5
College graduate (>15th)	Very difficult to read.	6

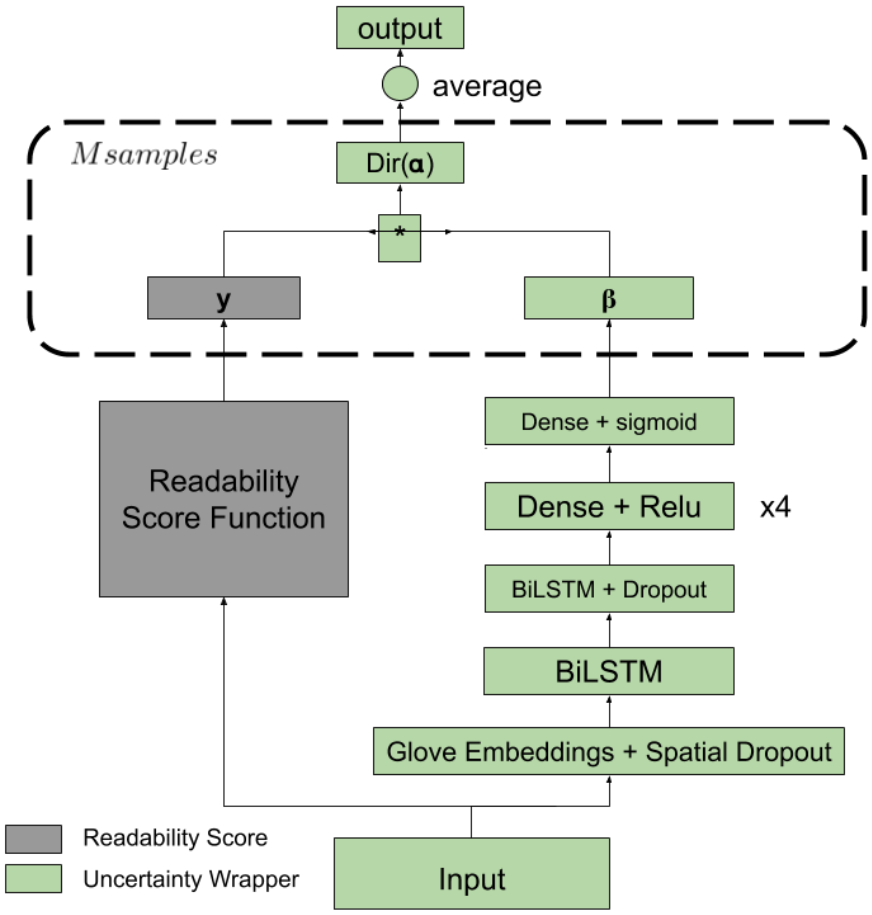
As outputs of each readability score may differ from one to another, it was necessary to develop a mapping function that transforms the result of each type of score to one of the seven categories described in Table 6.1.

[3] https://github.com/shivam5992/textstat

To build the dataset, we create individual samples as the triplet $\langle x, m_r(f_r(x)), m_{con}(f_{con}(x))\rangle$, where f_r correspond to each readability function and m_r the required mapping function. The subscript *con* refers to the consensus output. Hence, for each text in the corpus, we create seven inputs, one for each different readability function.

Estimating the uncertainty. Using the described triplets, we now proceed to apply the wrapper described in section 3.2. The output of the wrapper y^w transforms the score of each readability score $m_r(f_r(x))$ into a RV that models $p(y^w|x, m_r(f_r(x)), w) \sim Dir(\alpha)$. Here, the concentration parameter α is decomposed into $\alpha = \beta \cdot m_r(f_r(x))$, where β corresponds to the output of the wrapper ANN as shown in Figure 6.1.

Figure 6.1: Model used to estimate the aleatoric uncertainty



The wrapper is trained applying the loss as defined in equation 3.17, where the label corresponds to $m_{con}(f_{con}(x))$ in this case. Finally, the output of the wrapper can be obtained by sampling from the resulting Dirichlet distribution as defined in equation 3.16.

Word attribution. In this case, contrary to what they propose in the original paper,

we are not analyzing the whole uncertainty as measured by the predictive entropy of the resulting Dirichlet distribution. Here, we understand that this predictive entropy is combining two sources of uncertainty: one derived from the disagreement among the different readability metrics applied, and a measure of uncertainty due to the variability inherent in the texts analyzed. It is this later source of uncertainty what we use here to attribute the complexity of the input sentence.

Hence, in this work, we apply the DeepExplainer method to the wrapper model, in green in Figure 6.1, to analyze the variations induced by the different input elements in the activations of the wrapper NN. Next, we use the visualization tools provided by the SHAP library [4] to analyze which words are burdening more the readability of a sentence.

6.4 Experiments

To validate the consistency of the proposed method, we run experiments using RACE dataset Lai et al. [2017]. The dataset was meant for benchmark evaluation of methods in the reading comprehension task. RACE consists of 27,933 passages and 97,687 questions collected from English exams for middle and high school Chinese students. For analyzing the readability of the texts, we only took the 27,933 text passages. We observed the same split as defined in the original RACE paper, with 25,137, 1,389 and 1,407 examples for train, validation and test, respectively.

Using RACE, we generated the dataset, as explained in section 6.3. We applied a pre-processing of texts, including the removal of punctuation symbols, numbers and expansion of English contractions. Next, we trained the Dirichlet wrapper for 40 epochs in Google Colab, using a learning rate of 1e-3 and a λ parameter of 0.1, optimized manually observing the output of the wrapper. We also set a maximum sequence length of 60 words and 30 units for the hidden dense layers. Once the wrapper was trained, we applied DeepExplainer, using a sample of 1000 examples from the training set to learn the Shap values.

6.4.1 Results

In this section, we discuss the results obtained when applying the method to the RACE test dataset. First of all, we wanted to analyse whether the wrapper was able to capture

[4] https://github.com/slundberg/shap

uncertainties beyond the disagreement among the set of readability scores used for training. Because the wrapper is using the input text and the corresponding readability score, it is expected to identify uncertainties associated with the text itself, such as the ambiguity of some words.

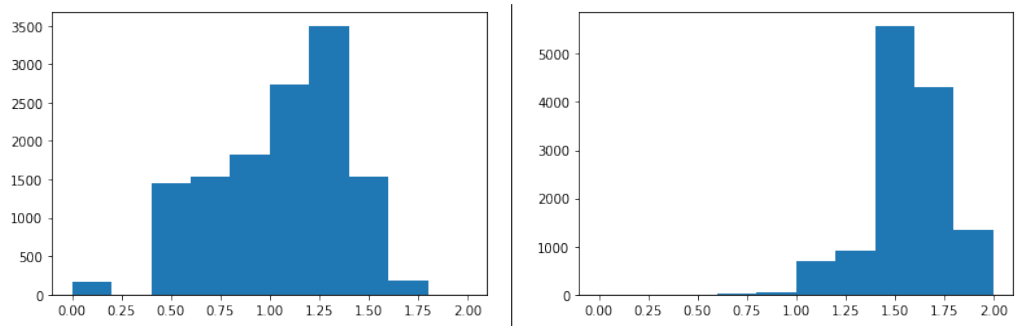


Figure 6.2: (a) Entropy of the disagreement among the scores. (b) Entropy predicted by the wrapper

In figure 6.2, we observe how the wrapper uncovers higher uncertainty scores than those obtained using the disagreement among the readability values[5]. We hypothesize that this behavior is due to the wrapper detecting rare or ambiguous words contributing to the uncertainty of the predicted readability score.

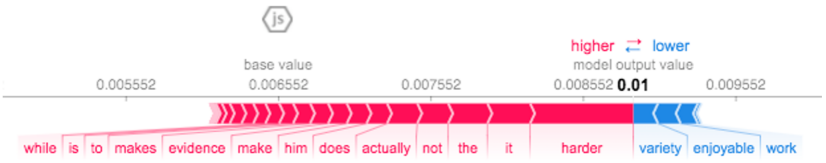
To confirm this hypothesis, we carried out a qualitative analysis of the predictions of some sentences. For texts with high uncertainty and reading complexity, we plot the corresponding SHAP values and substitute those words that contributed more to the uncertainty score for more frequent or simpler expressions. Figure 6.3 displays some examples of the results obtained:

We can observe that by replacing those words, we reduce the value of the readability score, i.e. the consensus. Thus, we observe a correlation between the uncertainty measure obtained with the wrapper and the readability complexity, allowing to identify which parts of the text are more related to this complexity, being possible, therefore, to modify them to ease the text complexity.

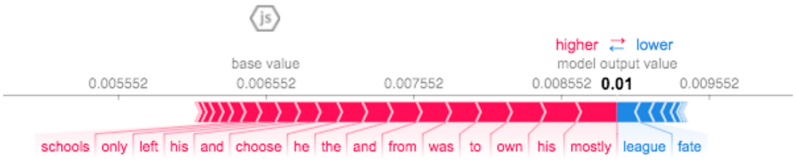
Potential applications of the method include online editors that can use the readability score only when there is low uncertainty on its prediction, or that can highlight which words to change to ease the comprehension of a text. Even though the results obtained are preliminary, we think that this work can serve as the basis for future studies. Potential extensions may include the automatization of these alterations by applying a

[5] maximum entropy for seven classes, as in this use case, would correspond to 1.9459

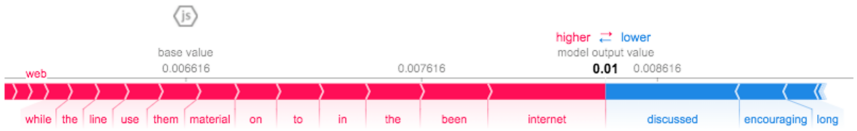
Figure 6.3: Modifying the more uncertain part of the sentence, the complexity decreases.



There is evidence to suggest that while variety certainly makes the worker's life more enjoyable, it does not actually make him work harder.	16th and 17th grade
There is evidence to suggest that while variety certainly makes the worker's life cooler, it does not actually make him work harder.	11th and 12th grade



In the end, only MIT rejected his application and he was left to choose his own fate from the selective prestigious and mostly Ivy League schools.	16th and 17th grade
In the end, only MIT rejected his application and he was left to choose his own fate from the selective prestigious schools.	10th and 11th grade



How to protect children Web fans from unsuitable material on-line while encouraging them to use the Internet has long been discussed in the US.	15th and 16th grade
How to protect children Web fans from unsuitable material on-line while encouraging them to use the Internet has long been talked in the US.	10th and 11th grade

framework like TextAttack Morris et al. [2020]. Moreover, we foresee other potential research lines that might include the analysis of other languages beyond English or the incorporation of alternative methods for readability assessment.

Chapter 7

Conclusion

Contents

With the advent of the Internet, Big Data and the success of technologies like Deep Learning, many companies are currently including AI to their digital solutions. However, not all companies are able to develop their own AI technology, as requirements of this type of technology in terms of talent and computational resources can be very demanding. Many technological enterprises have identified this demand and are providing out-of-the-box AI solutions through MLaaS platforms. These platforms deliver a portfolio of ML services or APIs that companies can consume on the cloud by providing their data and receiving the corresponding predictions.

In this work, we take the point of view of practitioners that have to make use of these APIs. Sometimes there may be different alternatives for a given problem, e.g., different Sentiment Analysis APIs, and they have to choose the more convenient to their problem and data. Sometimes the problem is that the definition of the service is slightly different from the particular need, maybe the image categories identified by the API are not the same as in their use case.

In this thesis, we propose to use the uncertainty of the predictions to assess the quality of a given classification API when applied to a target application. Thus, the goal of this work is to enrich the output of those APIs, considering them as black-boxes, with an uncertainty measure. Once equipped with this measure of uncertainty, practitioners will be able to make more informed decisions on whether to trust or not the predictions of those third-party APIs.

In the literature, many works exist based on Bayesian methods that can estimate the different types of uncertainty, both related to the trained model or the noise inherent in the data. We studied the use of different types of RV for modelling the output of the classification systems, mainly following Gaussian or Dirichlet distributions. Even though some of those methods present excellent results on estimating uncertainty in different scenarios, they all require to have access to the internals of the classifier. This constraint compelled us to develop a method that acts as a wrapper around the black-box classifier, operating only with the input passed to the black-box, and the result obtained. Even though the wrapper has neither access to the internals of the model nor the possibility of modifying it, it can obtain the uncertainty associated with the prediction. A relevant feature of the proposed method is its ability to operate with any classification system, both categorical or probabilistic, without imposing any architecture, working for DL, ML or even human-annotated classifications.

7.1 Contributions

As mentioned above, this thesis focuses on classifications systems and aims to furnish practitioners with an uncertainty wrapper for assessing existing APIS when applied to a target application. The proposed wrapper operates on top of the given API, enriching its predictions with an uncertainty score that practitioners can use to evaluate the performance of that API into their use case.

The main advantages of this wrapper are:

- **Black-box compatible**. Differently from previous works that measure uncertainty in Deep Learning models, the novelty of the proposed method relies on the fact that it can operate in the constrained setting of pure black-box models[1].

- **Model agnostic**. Even though the examples included correspond to DL architectures, the wrapper can operate on top of any classification system. These include traditional ML models, such as logistic regressions, ensemble models or SVMs, or even human-based annotations.

- **Works with categorical predictions**. One advantage of the present work is its ability to work also when the black-box model returns hard predictions, i.e. just the predicted class instead of a distribution over the different classes. The

[1] Defined by not having access to the internals of the black-box model

fact that it can measure the uncertainty even in the lack of a probabilistic output makes the proposed work to stand out in front of other approaches like calibration methods, that relay on those probabilities to adjust the predictions.

- **Domain agonistic**. As we have seen in the experiments, the model performs well in different domains. We can apply the wrapper to any problem where we can obtain a latent representation of the input.

- **Out-of-distribution detection**. In addition to the uncertainty estimation for the black-box results, the proposed Dirichlet wrapper incorporates mechanisms to capture the uncertainty even for out-of-distribution points. This feature brings extra-robustness to the proposed method as it will act as a vaccine against future changes in the distribution of the target data set.

In the Experiments chapter, we have seen how we applied the proposed wrapper to different domains such as NLP or CV. In those experiments, we have validated how the wrapper captured a more comprehensive set of uncertainties than traditional methods for uncertainty estimation or calibration, and how we increased the performance of the original black-boxes by implementing a rejection system leveraging those uncertainties obtained. Besides, in the CV example, we included an analysis of how the wrapper was able to detect uncertainty when facing OOD examples of unknown image categories.

Moreover, to illustrate the application of the proposed method, we included two more contributions addressing two real-life use cases. In the first one, we applied the wrapper to the analysis of over-qualification in job descriptions. This example was very interesting as it showed an application of the method where the classifier can be modified. In this case, by combining both aleatoric and epistemic uncertainties, we were able to improve the performance of the over-qualification classifier in a scenario with limited access to labelled data.

Last, we include an example of how to apply the method to classification systems different from ML or DL models. In this use case, we first applied the wrapper to analyse the disagreement among different "classic" readability metrics, using the uncertainty obtained as a measure of this disagreement. Then, by employing explainability tools, we analysed the uncertainty obtained, being able to identify those words that contributed more to this uncertainty, and offering the possibility to the editor of changing these words in order to relax the complexity of the resulting text.

7.2 Future Research

In addition to the work presented in this thesis, we see potential lines of future research that could be build on top of it. In the first place, one could study how to apply the wrapper to different types of classification scenarios, like multi-label systems. Moreover, options could include assessing its performance when applied to use cases with high dimensionality in the labels, like those typical in NLP such as Language Modeling dealing with an output of thousands of words from a vocabulary.

Other types of applications could require an analysis to see how to apply the wrapper to more complex scenarios, like Question Answering, where the output of the model varies on each request. Another potential application of the uncertainty in classification systems could be in the context of chatbots. In the case of combining different chatbots engines to apply given a user's utterance, the wrapper's uncertainty can be an excellent complement to other methods like those based on Reinforcement learning, Serban et al. [2017], to make the selection on which response to emit.

Moreover, we would like to study how to automatise the application of the wrapper. The goal would be to deliver it as a plugin for classification systems. We would like to see how to adapt for each case the hyperparameters of the ANN of the wrapper. Further, this automatisation could be extended to the selection of the rejection point using the resulting uncertainty score.

Besides its usage for rejecting uncertain predictions made by ML or DL classifiers, it could be also useful to explore other applications of the uncertainty. A potential line of research could be the use of the uncertainty for evaluating the performance of human annotators. By considering each of those human oracles as a target application, we could train the wrapper to estimate the uncertainty of their predictions and include only those whose predictions are less uncertain.

Finally, some questions left opened with regards to the interpretability of the uncertainty obtained could be addressed. In the experiments carried out in the readability use case, we applied SHAP values to analyse the words that contributed more to the uncertainty on readability measures. We reckon that linking this uncertainty with explainability techniques might help on better understanding the performance of classification systems, especially for domain shifting scenarios.We firmly believe that uncertainty can be beneficial in the explanation of bias or discrimination existing in some ML models.

Hopefully, future research will shed light on some of these questions.

Publications

This thesis has led to the publications summarized below.

7.3 Journals submissions

Accepted

- Mena, J., Pujol, O., and Vitrià, J. (2020). *Uncertainty-based Rejection System for Black-box Classifiers*. *IEEE Access*, doi: 10.1109/ACCESS.2020.2996495.

Submitted

- Mena, J., Pujol, O., and Vitrià, J. (2020). *A Survey on Uncertainty Estimation in Classification Systems*. Submitted to *ACM Computing Surveys (CSUR)*. Under review.

7.4 International Conferences

Accepted

- Mena, J., Brando, A., Pujol, O., and Vitrià, J. *Uncertainty estimation for black-box classification models: a use case for sentiment analysis*. In Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA), Madrid, Spain, 2019.

- Mena, J., Pujol, O., and Vitrià, J. *Dirichlet uncertainty wrappers for actionable algorithm accuracy accountability and auditability*. In ACM Conference on Fairness, Accountability, and Transparency (ACM FAT 2020), Barcelona, Spain, 2020.

Submitted

- Mena, J., Pujol, O., and Vitrià, J. *Unsupervised, Uncertainty-based Text Readability Assessment*. Submitted to The 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020), Online Conference, 2020. Under review.

7.5 Workshops

Accepted

- Mena, J., Torrent-Moreno, M., Portell, L., González, D., Pujol, O., and Vitrià, J. *Analysis of Vocational Education and Training and the labour market in Catalonia. A Data-driven approach.*. In The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases - ECML PKDD 2019 (The 4th Workshop on Data Science for Social Good - SoGood 2019), Würzburg, Germany, 2019.

Bibliography

D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016. Cited on page 5.

AvrioAI. Avrio ai: Ai talent platform,. https://www.goavrio.com/, 2019. Accessed: 2019-08-01. Cited on page 5.

S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46, 07 2015. Cited on page 105.

D. Barber and C. M. Bishop. Ensemble learning in bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences*, 168:215–238, 1998. Cited on pages 20 and 33.

P. L. Bartlett and M. H. Wegkamp. Classification with a reject option using a hinge loss. *J. Mach. Learn. Res.*, 9:1823–1840, June 2008. Cited on page 46.

R. Barzilay and N. Elhadad. Sentence alignment for monolingual comparable corpora. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 25–32. Association for Computational Linguistics, 2003. Cited on page 105.

C. M. Bishop. Bayesian methods for neural networks. 1995. Cited on page 17.

J. Blitzer, M. Dredze, and F. Pereira. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic, June 2007. Association for Computational Linguistics. Cited on page 75.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks, 2015. Cited on pages 22, 28, 33, and 40.

N. Bostrom. Ethical issues in advanced artificial intelligence. In *ScienceFiction and Philosophy: From Time Travel to Superintelligence*, pages 277—284. Wiley-Blackwell, 2009. Cited on page 5.

D. Bouneffouf, R. Laroche, T. Urvoy, R. Féraud, and R. Allesiardo. Contextual bandit for active learning: Active thompson sampling. In *International Conference on Neural Information Processing*, pages 405–412. Springer, 2014. Cited on page 45.

A. Brando, J. A. Rodriguez, J. Vitria, and A. R. Muñoz. Modelling heterogeneous distributions with an uncountable mixture of asymmetric laplacians. In *Advances in Neural Information Processing Systems*, pages 8836–8846, 2019. Cited on page 29.

W. M. Briggs. It is time to stop teaching frequentism to non-statisticians. *arXiv:1201.2590*, 2012. Cited on page 11.

J. Bröcker and L. A. Smith. Increasing the reliability of reliability diagrams. *Weather and forecasting*, 22(3):651–661, 2007. Cited on page 37.

C. Cardellino. Spanish Billion Words Corpus and Embeddings, August 2019. Cited on page 98.

T. Chen, E. Fox, and C. Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691, 2014. Cited on pages 18 and 33.

W. Chen, Y. Shen, H. Jin, and W. Wang. A variational dirichlet framework for out-of-distribution detection, 2018. Cited on pages 31, 32, 33, 36, 55, 56, and 68.

Y. Chen, C. Yang, Y. Zhang, and Y. Li. Deep conditional adaptation networks and label correlation transfer for unsupervised domain adaptation. *Pattern Recognition*, 98: 107072, 2020. Cited on page 84.

C. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, 16(1):41–46, 1970. Cited on page 46.

A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In G. Gordon, D. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. Cited on page 84.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996. Cited on page 44.

M. Coleman and T. L. Liau. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283, 1975. Cited on page 105.

F. Condessa, J. Kovacevic, and J. M. Bioucas-Dias. Performance measures for classification systems with rejection. *CoRR*, abs/1504.02763, 2015. Cited on pages xi, 47, 48, 74, and 100.

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*, 6(5):1140–1147, 1995. Cited on page 46.

C. Cortes, G. DeSalvo, and M. Mohri. Boosting with abstention. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1660–1668. Curran Associates, Inc., 2016. Cited on page 46.

L. Cosmides and J. Tooby. Are humans good intuitive statisticians after all? rethinking some conclusions from the literature on judgment under uncertainty. *cognition*, 58 (1):1–73, 1996. Cited on page 42.

R. T. Cox. Probability, frequency and reasonable expectation. *American journal of physics*, 14(1):1–13, 1946. Cited on page 10.

A. Culotta and A. McCallum. Reducing labeling effort for structured prediction tasks. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, pages 746–751. AAAI Press, 2005. Cited on pages 34, 48, and 77.

D. G. E. D. T. P. D. V. C. D. Sculley, Gary Holt and M. Young. Machine learning: Thehigh interest credit card of technical debt. In *SE4ML: Software Engineering for Machine Learning (NIPS 2014)*, 2014. Cited on page 5.

I. Dagan and S. P. Engelson. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the Twelfth International Conference on International Conference on Machine Learning*, ICML'95, pages 150–157, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. Cited on pages 34 and 76.

E. Dale and J. S. Chall. A formula for predicting readability: Instructions. *Educational research bulletin*, pages 37–54, 1948. Cited on page 104.

C. De Stefano, C. Sansone, and M. Vento. To reject or not to reject: That is the question-an answer in case of neural classifiers. *Trans. Sys. Man Cyber Part C*, 30 (1):84–94, Feb. 2000a. Cited on page 46.

C. De Stefano, C. Sansone, and M. Vento. To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(1):84–94, 2000b. Cited on page 46.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. Cited on page 84.

A. Der Kiureghian and O. Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009. Cited on page 15.

DigitalReasoning. Healthcare ai: Digital reasoning,. https://digitalreasoning.com/solutions/healthcare/, 2019. Accessed: 2019-08-01. Cited on page 5.

R. El-Yaniv and Y. Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(May):1605–1641, 2010. Cited on page 46.

H. Elsahar and M. Gallé. To annotate or not? predicting performance drop under domain shift. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2163–2173, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. Cited on page 105.

M. H. Faber. On the treatment of uncertainties and probabilities in engineering decision analysis. 2005. Cited on pages 14 and 15.

L. Feng, N. Elhadad, and M. Huenerfauth. Cognitively motivated features for readability assessment. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 229–237, 2009. Cited on page 105.

M. Figurnov, S. Mohamed, and A. Mnih. Implicit reparameterization gradients. In *Advances in Neural Information Processing Systems*, pages 441–452, 2018. Cited on pages 23, 33, and 58.

T. François and E. Miltsakaki. Do nlp and machine learning improve traditional readability formulas? In *Proceedings of the First Workshop on Predicting and Improving Text Readability for Target Reader Populations*, PITR '12, page 49–57, USA, 2012. Association for Computational Linguistics. Cited on page 105.

L. Freeman. *Elementary applied statistics: for students in behavioral science*. Wiley, 1965. Cited on page 35.

Y. Gal. *Uncertainty in deep learning*. PhD thesis, PhD thesis, University of Cambridge, 2016. Cited on pages 28, 33, 35, 36, 43, 44, 48, 99, and 101.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation. *arXiv preprint arXiv:1506.02157*, 2015. Cited on pages 25, 28, 33, and 40.

J. Gast and S. Roth. Lightweight probabilistic deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3369–3378, 2018. Cited on pages 30, 31, 33, 37, 39, 55, and 56.

Y. Geifman and R. El-Yaniv. Selective classification for deep neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4878–4887. Curran Associates, Inc., 2017a. Cited on page 48.

Y. Geifman and R. El-Yaniv. Selective classification for deep neural networks. In *Advances in neural information processing systems*, pages 4878–4887, 2017b. Cited on page 46.

Y. Geifman and R. El-Yaniv. Selectivenet: A deep neural network with an integrated reject option, 2019. Cited on page 46.

X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 513–520, USA, 2011. Omnipress. Cited on page 74.

A. Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011. Cited on pages 21, 22, and 33.

R. Gunning. *The Technique of Clear Writing*. McGraw-Hill, 1968. Cited on page 105.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. Cited on pages 34, 80, 81, and 83.

B. Hanczar. Performance visualization spaces for classification with rejection option. *Pattern Recognition*, 96:106984, 2019. Cited on page 47.

D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016. Cited on page 40.

J. M. Hernandez-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1861–1869, Lille, France, 07–09 Jul 2015. PMLR. Cited on pages 23, 30, and 33.

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*, pages 5–13, 1993. Cited on pages 19, 20, 22, 27, and 33.

M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013. Cited on pages 22 and 33.

S. C. Hora. Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management. *Reliability Engineering & System Safety*, 54(2-3):217–223, 1996. Cited on page 15.

N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011. Cited on page 36.

G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, July 2017. Cited on page 84.

Ideal. Ideal: Ai for recruiting software,. https://ideal.com/, 2019. Accessed: 2019-08-01. Cited on page 5.

M. Jankowiak and F. Obermeyer. Pathwise derivatives beyond the reparameterization trick. *arXiv preprint arXiv:1806.01851*, 2018. Cited on pages 23 and 33.

E. T. Jaynes. Bayesian methods: General background. 1986. Cited on page 10.

S. Jonna, K. K. Nakka, and R. R. Sahay. My camera can see through fences: A deep learning approach for image de-fencing. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 261–265. IEEE, 2015. Cited on page 42.

A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017. Cited on pages 14, 28, 29, 30, 31, 33, 43, and 51.

J. P. Kincaid, R. P. Fishburne Jr, R. L. Rogers, and B. S. Chissom. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. *STARS*, 1975. Cited on page 104.

G. R. Klare. Assessing readability. *Reading research quarterly*, pages 62–102, 1974. Cited on page 105.

D. A. Knowles. Stochastic gradient variational bayes for gamma approximating distributions. *arXiv preprint arXiv:1509.01631*, 2015. Cited on pages 23 and 33.

G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017. Cited on page 108.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017. Cited on pages 26, 33, and 40.

T. C. Landgrebe, D. M. Tax, P. Paclík, and R. P. Duin. The interaction between classification and reject performance for distance-based reject-option classifiers. *Pattern Recognition Letters*, 27(8):908–917, 2006. Cited on page 46.

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*, 2018. Cited on page 61.

D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994. Cited on page 44.

Y. Li, J. M. Hernández-Lobato, and R. E. Turner. Stochastic expectation propagation. In *Advances in neural information processing systems*, pages 2323–2331, 2015. Cited on pages 24 and 33.

J. Liang, R. He, Z. Sun, and T. Tan. Exploring uncertainty in pseudo-label guided unsupervised domain adaptation. *Pattern Recognition*, 96:106996, 2019. Cited on page 84.

A. Loquercio, M. Segu, and D. Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020. Cited on pages 31 and 33.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017. Cited on pages 42 and 105.

S. M. Lundberg, B. Nair, M. S. Vavilala, M. Horibe, M. J. Eisses, T. Adams, D. E. Liston, D. K.-W. Low, S.-F. Newman, J. Kim, et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering*, 2(10):749, 2018. Cited on page 106.

D. J. MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992a. Cited on pages 17, 19, and 43.

D. J. MacKay. The evidence framework applied to classification networks. *Neural computation*, 4(5):720–736, 1992b. Cited on pages 13, 17, 33, 43, and 44.

D. J. MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992c. Cited on pages 17, 21, and 33.

A. Malinin and M. Gales. Predictive uncertainty estimation via prior networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7047–7058. Curran Associates, Inc., 2018. Cited on pages 31, 33, 36, 41, 54, 56, 60, and 68.

G. H. Mc Laughlin. Smog grading-a new readability formula. *Journal of reading*, 12 (8):639–646, 1969. Cited on page 105.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a. Cited on page 98.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b. Cited on page 75.

D. Milios, R. Camoriano, P. Michiardi, L. Rosasco, and M. Filippone. Dirichlet-based gaussian processes for large-scale calibrated classification. In *Advances in Neural Information Processing Systems*, pages 6005–6015, 2018. Cited on page 23.

J. X. Morris, E. Lifland, J. Y. Yoo, and Y. Qi. Textattack: A framework for adversarial attacks in natural language processing. *arXiv preprint arXiv:2005.05909*, 2020. Cited on page 110.

L. Mou, Z. Meng, R. Yan, G. Li, Y. Xu, L. Zhang, and Z. Jin. How transferable are neural networks in nlp applications? *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016. Cited on page 74.

M. S. A. Nadeem, J.-D. Zucker, and B. Hanczar. Accuracy-rejection curves (arcs) for comparing classification methods with a reject option. In *Machine Learning in Systems Biology*, pages 65–81, 2009. Cited on page 46.

M. P. Naeini, G. Cooper, and M. Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. Cited on page 81.

R. M. Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012. Cited on pages 18, 25, and 33.

A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632, 2005. Cited on pages 37 and 39.

I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016. Cited on pages 26 and 33.

S. E. Petersen and M. Ostendorf. A machine learning approach to reading level assessment. *Computer speech & language*, 23(1):89–106, 2009. Cited on page 105.

I. Pillai, G. Fumera, and F. Roli. Multi-label classification with a reject option. *Pattern Recogn.*, 46(8):2256–2266, Aug. 2013. Cited on page 46.

J. C. Redish. Understanding the limitations of readability formulas. *IEEE Transactions on Professional Communication*, PC-24(1):46–48, 1981. Cited on page 104.

M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938, 2016a. Cited on page 105.

M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016b. Cited on page 42.

C. Riquelme, G. Tucker, and J. Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018. Cited on page 40.

S. Rossi, P. Michiardi, and M. Filippone. Good initializations of variational bayes for deep models. *arXiv preprint arXiv:1810.08083*, 2018. Cited on page 23.

F. R. Ruiz, M. T. R. AUEB, and D. Blei. The generalized reparameterization gradient. In *Advances in neural information processing systems*, pages 460–468, 2016. Cited on pages 23 and 33.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. Cited on page 17.

D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, Z. Wen, et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018. Cited on page 45.

P. Sadowski and P. Baldi. Neural network regression with beta, dirichlet, and dirichlet-multinomial outputs. 2018. Cited on pages 29, 30, 33, and 68.

M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. Cited on page 84.

T. Scheffer, C. Decomain, and S. Wrobel. Active hidden markov models for information extraction. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, IDA '01, pages 309–318, London, UK, UK, 2001. Springer-Verlag. Cited on pages 34, 48, and 77.

S. E. Schwarm and M. Ostendorf. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics, 2005. Cited on page 105.

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*, 255:16–29, 2014. Cited on page 14.

R. Senter and E. A. Smith. Automated readability index. Technical report, CINCINNATI UNIV OH, 1967. Cited on page 104.

I. V. Serban, C. Sankar, M. Germain, S. Zhang, Z. Lin, S. Subramanian, T. Kim, M. Pieper, S. Chandar, N. R. Ke, et al. A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*, 2017. Cited on page 114.

B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009. Cited on page 44.

C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948. Cited on pages 35 and 99.

S. Shekhar, M. Ghavamzadeh, and T. Javidi. Binary classification with bounded abstention rate. *arXiv preprint arXiv:1905.09561*, 2019. Cited on page 46.

A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. *CoRR*, abs/1704.02685, 2017. Cited on page 105.

J. Snoek, Y. Ovadia, E. Fertig, B. Lakshminarayanan, S. Nowozin, D. Sculley, J. Dillon, J. Ren, and Z. Nado. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13969–13980, 2019. Cited on pages 27, 40, and 41.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics. Cited on page 75.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. Cited on page 25.

S. Štajner and I. Hulpuş. Automatic assessment of conceptual text complexity using knowledge graphs. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 318–330, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics. Cited on page 105.

S. Štajner, R. Evans, C. Orasan, and R. Mitkov. What can readability measures really tell us about text complexity. In *Proceedings of the the Workshop on Natural Language Processing for Improving Textual Accessibility (NLP4ITA)*, pages 14–21. Citeseer, 2012. Cited on page 105.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. Cited on page 84.

S. Tan, X. Cheng, Y. Wang, and H. Xu. Adapting naive bayes to domain adaptation for sentiment analysis. In M. Boughanem, C. Berrut, J. Mothe, and C. Soule-Dupuy, editors, *Advances in Information Retrieval*, pages 337–349, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. Cited on page 74.

J. J. Thiagarajan, I. Kim, R. Anirudh, and P.-T. Bremer. Understanding deep neural networks through input uncertainties. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2812–2816. IEEE, 2019. Cited on pages 31 and 33.

S. Thulasidasan, T. Bhattacharya, J. Bilmes, G. Chennupati, and J. Mohd-Yusof. Combating label noise in deep learning using abstention. *arXiv preprint arXiv:1905.10964*, 2019. Cited on page 46.

Turnitin. Turnitin: Revision assistant,. http://turnitin.com/, 2019. Accessed: 2019-08-01. Cited on page 5.

S. Vajjala and I. Lučić. OneStopEnglish corpus: A new corpus for automatic readability assessment and text simplification. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 297–304, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. Cited on page 105.

T. vor der Brück, S. Hartrumpf, and H. Helbig. A readability checker with supervised learning using deep indicators. *Informatica*, 32(4), 2008. Cited on page 105.

Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*, 2018. Cited on pages 23, 33, and 99.

J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viégas, and J. Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics*, 26(1):56–65, 2019. Cited on page 42.

Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994. Cited on page 85.

Y. Xiao and W. Y. Wang. Quantifying uncertainties in natural language processing tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7322–7329, 2019. Cited on page 29.

C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018. Cited on page 27.

T. Zhang and F. J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *17th International Conference on Machine Learning*, 2000. Cited on page 45.