Bayesian Deep Learning for Mining and Analyzing Astronomical Data

Claire Theobald

UNIVERSITÉ DE LORRAINE

Thèse de Doctorat

Claire THEOBALD

Mémoire présenté en vue de l'obtention du
grade de Docteur de l'Université de Lorraine
Mention Informatique

École doctorale : IAEM

Unité de recherche : Laboratoire Lorrain de Recherche en Informatique et ses Applications UMR 7503

Soutenue le 6 juin 2023

Bayesian Deep Learning for Mining and Analyzing Astronomical Data

JURY

Présidente du jury :	**Marianne Clausel**, Professeure des universités, IECL – Site de Nancy, Faculté des sciences et Technologies, F-54000 Nancy, France
Examinatrice :	**Marie-Jeanne Lesot**, Maîtresse de conférences, Sorbonne Université, 75005, Paris, France
Rapporteur :	**Mário Figueiredo**, Professeur des universités, Université de Lisbonne, Instituto de Telecomunicações, Instituto Superior Técnico, 1049-001, Lisboa, Portugal
Rapporteur :	**Sébastien Destercke**, Directeur de recherche, CNRS, Université de Technologie de Compiegne, 60205, Compiegne, France
Directeur de thèse :	**Miguel Couceiro**, Professeur des universités, Université de Lorraine, CNRS, LORIA, F-54000 Nancy, France
Co-directeur de thèse :	**Frédéric Pennerath**, Maître de conférences, CentraleSupélec, LORIA, F-57000 Metz, France
Invité :	**Brieuc Conan-Guez**, Maître de conférences, Université de Lorraine, LORIA, F-57000 Metz, France

Résumé

Dans cette thèse, nous abordons le problème de la confiance que nous pouvons avoir en des systèmes prédictifs de type réseaux profonds selon deux directions de recherche complémentaires. Le premier axe s'intéresse à la capacité d'une IA à estimer de la façon la plus juste possible son degré d'incertitude liée à sa prise de décision. Le second axe quant à lui se concentre sur l'explicabilité de ces systèmes, c'est-à-dire leur capacité à convaincre l'utilisateur humain du bien fondé de ses prédictions.

Le problème de l'estimation des incertitudes est traité à l'aide de l'apprentissage profond bayésien. Les réseaux de neurones bayésiens admettent une distribution de probabilité sur leurs paramètres, qui leur permettent d'estimer différents types d'incertitudes. Tout d'abord, l'incertitude aléatoire qui est liée aux données, mais également l'incertitude épistémique qui quantifie le manque de connaissance que le modèle possède sur la distribution des données. Plus précisément, cette thèse propose un modèle de réseau de neurones bayésien capable d'estimer ces incertitudes dans le cadre d'un problème de régression multivarié. Ce modèle est appliqué dans le contexte du projet ANR "AstroDeep" à la régression des ellipticités complexes sur des images de galaxies. Ces dernières peuvent être corrompues par différences sources de perturbation et de bruit qui peuvent être estimées de manière fiable par les différentes incertitudes. L'exploitation de ces incertitudes est ensuite étendue à la cartographie de galaxies, puis au "coaching" du réseau de neurones bayésien. Cette dernière technique consiste à générer des données de plus en plus complexes durant l'apprentissage du modèle afin d'en améliorer les performances.

Le problème de l'explicabilité est quant à lui abordé via la recherche d'explications contrefactuelles. Ces explications consistent à identifier quels changements sur les paramètres en entrée auraient conduit à une prédiction différente. Notre contribution dans ce domaine s'appuie sur la génération d'explications contrefactuelles basées sur un autoencodeur variationnel (VAE) et sur un ensemble de prédicteurs entraînés sur l'espace latent généré par le VAE. Cette méthode est plus particulièrement adaptée aux données en haute dimension, telles que les images. Dans ce cas précis, nous parlerons d'explications contrefactuelles visuelles. En exploitant à la fois l'espace latent et l'ensemble de prédicteurs, nous arrivons à produire efficacement des explications contrefactuelles visuelles atteignant un degré de réalisme supérieur à plusieurs méthodes de l'état de l'art.

Abstract

In this thesis, we address the issue of trust in deep learning predictive systems in two complementary research directions. The first line of research focuses on the ability of AI to estimate its level of uncertainty in its decision-making as accurately as possible. The second line, on the other hand, focuses on the explainability of these systems, that is, their ability to convince human users of the soundness of their predictions.

The problem of estimating the uncertainties is addressed from the perspective of Bayesian Deep Learning. Bayesian Neural Networks assume a probability distribution over their parameters, which allows them to estimate different types of uncertainties. First, aleatoric uncertainty which is related to the data, but also epistemic uncertainty which quantifies the lack of knowledge the model has on the data distribution. More specifically, this thesis proposes a Bayesian neural network can estimate these uncertainties in the context of a multivariate regression task. This model is applied to the regression of complex ellipticities on galaxy images as part of the ANR project "AstroDeep". These images can be corrupted by different sources of perturbation and noise which can be reliably estimated by the different uncertainties. The exploitation of these uncertainties is then extended to galaxy mapping and then to "coaching" the Bayesian neural network. This last technique consists of generating increasingly complex data during the model's training process to improve its performance.

On the other hand, the problem of explainability is approached from the perspective of counterfactual explanations. These explanations consist of identifying what changes to the input parameters would have led to a different prediction. Our contribution in this field is based on the generation of counterfactual explanations relying on a variational autoencoder (VAE) and an ensemble of predictors trained on the latent space generated by the VAE. This method is particularly adapted to high-dimensional data, such as images. In this case, they are referred as counterfactual visual explanations. By exploiting both the latent space and the ensemble of classifiers, we can efficiently produce visual counterfactual explanations that reach a higher degree of realism than several state-of-the-art methods.

Cette thèse de doctorat fut le fruit de plusieurs années de recherches intenses, et comme c'est bien souvent le cas, la progression ne fut pas "linéaire" ! Bien évidemment, rien de tout ceci n'aurait été possible sans l'aide et le soutien précieux de nombreuses personnes, dont j'aimerais dès à présent remercier.

En premier lieu, je souhaiterais bien sûr remercier le jury de thèse dans son ensemble: les rapporteurs de thèse, Mário Figueiredo et Sébastien Destercke, qui ont pris le temps de lire mon mémoire de thèse et d'apporter des critiques très pertinentes afin d'en améliorer le contenu; les examinatrices, Marie-Jeanne Lesot et Marianne Clausel, qui ont su créer des dialogues et une discussion pertinente lors de la soutenance de thèse. Et

bien évidemment mes directeurs et co-encadrants de thèse: Miguel Couceiro, Frédéric Pennerath, Brieuc Conan-Guez et Amedeo Napoli. Tous ont su me soutenir tout le long de ma thèse !

Ensuite, je voudrais remercier l'ensemble des doctorant·e·s, post-doctorant·e·s, ingénieur·e·s et stagiaires que j'ai pu rencontrer au Loria. Malgré le Covid qui a frappé très rapidement après le début de ma thèse et nous a forcés à travailler à distance, je ne peux pas oublier les moments que j'ai pu passer au laboratoire (et ailleurs !) en leur compagie ! La thèse est malheureusement très souvent un travail solitaire, et pouvoir parler à des personnes partageant cette épreuve aide énormément. Je ne vais pas citer tous les noms ici, il y en aurait beaucoup trop et je risquerais d'en oublier... Cela étant dit, si vous lisez ceci, sachez que je n'oublierai jamais les moments passés en votre compagnie, que je vous ai connu pendant seulement une journée ou pendant mes trois années et demi de thèse !!

Cette thèse fut également inscrite au sein du projet ANR AstroDeep, dans lequel j'ai pu collaborer avec le laboratoire Astroparticule et Cosmologie de l'université Paris Cité, ainsi que le Département D'Astrophysique du CEA de Saclay. Je tiens donc à remercier les personnes de ces laboratoires qui ont travaillé avec moi lors de cette thèse.

Dans le cadre de mes activités d'enseignement, j'ai aussi travaillé avec l'IDMC, la FST de Nancy, et l'IUT de Metz. Je tiens à remercier les équipes pédagogiques de ces établissements qui m'ont accompagné tout au long de mes heures en tant que vacataire ou ATER.

Enfin, je souhaite remercier toutes les personnes qui, hors du cadre de recherche, m'ont accompagné lors de ces dernières années. Cela inclut donc bien évidemment mes ami· e·s; à la fois en physique, mais aussi, et de manière plus importante, en ligne: il ne va pas sans dire que les connexions que j'ai pu faire en ligne furent d'une capitale importance pour la réussite de cette thèse, surtout dans le contexte de la pandémie et du confinement !

Et finalement, je n'oublie bien évidemment pas ma famille: mon père, ma mère, mon frère, ainsi que toute ma famille étendue. Rien n'aurait été possible sans leur soutien.

Merci énormément à tout le monde !

Everyone, thank you so much !

Contents

List of figures

List of tables

List of symbols

x	instance
y	label
z	latent variable
ω	model parameters
$\mathcal{X}$	set of instances
$\mathcal{Y}$	set of labels
$\mathcal{Z}$	latent space
$\mathcal{D}$	dataset
Ω	set of parameters
$\mathbb{R}$	Real numbers set
$\mathcal{P}(\cdot)$	power set
$p(\cdot)$	probability mass function (discrete case)/probability density function (continuous case)
$P(\cdot)$	probability measure
$\hat{X}$	realization of a random variable X
$\odot$	Hadamard (element-wise) product
$*$	convolution operator
$\mathcal{B}(p)$	Bernoulli distribution with parameter $0 \leq p \leq 1$
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with mean μ and variance σ^2

List of acronyms

AI	Artificial intelligence
AUC	Area under curve
BDL	Bayesian deep learning
BNN	Bayesian neural network
CDF	Cumulative distribution function
CF	Counterfactual
CNN	Convolutional neural network
ELBO	Evidence lower bound
FNN	Feedforward neural network
GAN	Generative adversarial networks
KL	Kullback-Leibler
LSST	Legacy Survey of Space and Time
MC	Monte Carlo
MCMC	Markov chain Monte Carlo
MLP	Multilayer perceptron
MVN	Multivariate Normal distribution
PDF	Probability density function
PID	Proportional-Integral-Derivative
ROC	Receiver Operating Characteristic
VAE	Variational autoencoder

Chapter 1.

Introduction

"Why should I trust you ?". In today's age, AI has reached performance levels in various tasks such as computer vision (Krizhevsky et al., 2012; Rombach et al., 2022), NLP (Graves, 2013; Devlin et al., 2019), speech processing (Graves et al., 2013), and so on. AI slowly became an invaluable asset in many application domains, such as biology, chemistry, medicine, autonomous driving, and physics, where datasets are becoming larger and more complex. However, even though AI models such as deep neural networks can achieve impressive results on many different tasks, there has been a growing need to make such models more transparent.

Transparency comes with explainability, interpretability, accountability and safety. As AI systems become more and more complex, it has become increasingly more difficult to understand *why* they made a particular decision. Even if these predictions can be correct 99% of the time, it does not mean that the AI uses patterns that are intelligible or fair. For instance, 2015 was the year when the first casualty due to an autonomous car has been recorded (NHTSA, 2017) when the AI piloting the car mistook a white truck for the bright sky, raising concerns in terms of safety with machine learning systems. In the same year, an image recognition algorithm made outrageous classifications of black people due to a misinterpretation of their black skin in the Google Photos app (Guynn, 2015), which put into question the fairness (or lack thereof) of AI; notably how is it possible to make sure these AI systems are unbiased in regards to discriminating behavior against race, gender, age, etc. More recently, models such as DALL-E 2 (Ramesh et al., 2022), Stable Diffusion (Rombach et al., 2022) or Midjourney[1] achieved impressive results in image generation from text (see Fig. 1.1), becoming viral in the public space. But these models also raised ethical concerns on their training process and usage.

It is then clear that performance alone is insufficient to make an AI system accepted by the public, institutions, companies or in applied research. Slowly, regulation has been put in place requiring such systems to have more transparency and accountability. The European's Union General Data Protection Regulation (GDPR)[2] states that any user has a right to obtain intelligible information about the model's prediction, and in 2019 the US government enacted the "Algorithmic Accountability Act of 2019" (Wyden, 2019).

But the need for intelligible and transparent models goes even beyond regulation and ethical issues: they are also very much valuable in the scientific domain. As the use of machine learning models continue to increase in many scientific areas, so does the need

[1] https://www.midjourney.com/
[2] https://gdpr.eu/

(a) Midjourney

(b) DALL-E 2

Figure 1.1.: Pictures generated by Midjourney and DALL-E 2. (a) Midjourney generated image, query: "A gigantic dragon looking upon an icy landscape at night with a full moon". (b) DALL-E 2 generated image, query: "A Shiba Inu dog wearing a beret and black turtleneck". Credits: (b) OpenAI[3].

for models that can be trusted and interpretable. For instance, the *Legacy Survey of Space and Time* (LSST Science Collaboration, 2009), conducted at the Vera C. Rubin Observatory, is expected to take over 15 petabytes of uncompressed pictures of the observable Universe. Such vast amounts of data are impossible to be analyzed by humans, and as such, astrophysicists are turning their eyes to machine learning systems to review the collected data.

Figure 1.2.: The Vera C. Rubin Observatory under construction in Chile. Credits: Rubin Obs./NSF/AURA.

One of the goals of the LSST is to study *dark energy*, a theorized component of unknown nature introduced in the current cosmological standard model to explain the acceleration of the Universe expansion, which can be probed by studying the mass distribution across the Universe. However, inferring this distribution requires precise analysis of the data provided by the survey, a dataset which can be corrupted by noise or outliers. Deep

[3] https://twitter.com/gdb/status/1511718170804908037

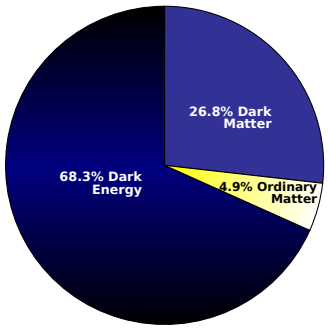


Figure 1.3.: The distribution of matter and energy in the universe. Dark energy currently represents more than 2/3 of the total in the universe. Source: ESA (2013).

neural networks learning metrics from the survey must be capable of quantify these sources of uncertainty, as these errors can propagate in the whole pipeline of the research process.

In this context, the ANR project AstroDeep[4] was put in place in November of 2019 to conduct research on the usage of *Bayesian Neural Networks* in cosmology. Bayesian Neural Networks (BNNs) are a variant of Neural Networks which are able to quantify uncertainties from different sources, in addition to their predictions which makes them very much suited for the task at hand. More precisely, BNNs separate *aleatoric* uncertainty, which is inherent from the data, from *epistemic* uncertainty, which is related to the lack of knowledge the model has on the data distribution.

However, uncertainty estimation is just one facet of the encompassing task of deconstructing the "black-box" of deep neural nets. The ability for AI systems to explain their predictions is another very important characteristic that is sought by cosmologists. Having a neural net capable of explaining its decisions in an intelligible way, both qualitatively and quantitatively, can help to detect patterns or biases that would have been missed otherwise.

More precisely, the question of "counterfactual explanations" consists of asking the model what change in the image would have caused it to make another decision. Such explanations could reveal problematic behavior from the neural network, putting its reliability into question. An interesting problematic would be to consider how these explanations could benefit from a Bayesian approach to deep learning, to make them more intelligible and plausible, especially when these explanations are applied to image data. Fig. 1.4 presents an example of a counterfactual visual explanation: how can a learned neural net can explain the prediction "brown hair" in the left picture by trying to change it to "black hair" ? The right image shows that, by default, neural networks learn patterns that are unintelligible to humans and unrelated to the actual task, causing issues regarding to its reliability.

[4] https://astrodeep.net/

Figure 1.4.: Counterfactual visual explanation example from the CelebA dataset (Liu et al., 2015). On the left: original image, a woman with brown hair. On the right: the picture has been modified by a neural network with the goal of predicting "black hair".

The problematic is then clear: how can we use Bayesian approaches to deep learning to give calibrated uncertainty estimates and reliable explanations, in the context of cosmological data ?

Answering this question will shed light to such techniques that can be applied beyond astrophysics: as we saw, these characteristics are sought in many application fields. Besides, studying machine learning systems learning on astronomical images can also help us understand complex behaviors and develop techniques that are tied to computer vision in general.

This thesis will then be presented as follows.

- Chapter 2 will present an overview of deep learning, presenting the main architectures related to the present work. More precisely, since the focus of this thesis is analyzing astronomical data which are images, we will focus our attention on convolutional neural networks for image processing. We will also point our attention to generative models such as VAEs or GANs, which are able to sample new images from a latent space, and will later be useful to generate counterfactual explanations.
- Chapter 3 focuses on Bayesian deep learning, presenting the framework to estimate aleatoric and epistemic uncertainty. The first part of this chapter focuses on presenting the current main techniques to train a BNN and quantify the uncertainties. The second part, on the other hand, presents our contribution (Theobald et al., 2020), "A Bayesian Neural Network for Dropout Regulation", which describes a Bayesian Neural Network which regulates its distribution by using an algorithm inspired by automation, in order to achieve better uncertainty estimates.
- Chapter 4 applies Bayesian Deep Learning to the estimation of galaxy ellipticities, an important step in the study of the mass distribution of the Universe in order to understand dark energy. Following our contribution (Theobald et al., 2021), "A Bayesian Convolutional Neural Network for Robust Galaxy Ellipticity Regression", we present a BNN capable of estimating the complex shape of the galaxies in a regression problem, while giving calibrated uncertainty estimates for both sources of noise and outliers. Then, we will describe a way to use epistemic uncertainty to map galaxy objects on a galaxy map. Finally, some preliminary work has been done to help the model learn from more difficult datasets by increasingly augmenting the complexity of the data during the training process. We coined this approach

"Network coaching".

- Chapter 5 introduces the state of explainable AI, by reviewing the main methods to extract explanations from a deep learning model. First, by looking at model-intrinsic and model-agnostic methods, and then some example-based explanation techniques. Finally, we will more precisely focus on counterfactual explanations, especially counterfactual visual explanations which are applied to images. We end this chapter by introducing results from the contribution (Theobald et al., 2022) "Clarity: an improved gradient method for producing quality visual counterfactual explanations", by showing the limits of the current counterfactual visual explanation algorithms.
- Chapter 6 is the final chapter where we go into more detail about the submission "Clarity: an improved gradient method for producing quality visual counterfactual explanations" (Theobald et al., 2022). We will give insights on why using a classifier to specifically exploit the structure of a latent space generates more realistic counterfactual visual explanations. Then, we present *Clarity*, a counterfactual visual explanation algorithm which uses a Bayesian neural network trained on a latent space to generate counterfactual examples. We end this chapter by looking at a variant of *Clarity* using Student's t-tests.
- Finally, Chapter 7 serves as an overall conclusion to the contributions presented in this work, with a list of future prospects which could be later developed.

Chapter 2.

A deep learning overview

In this chapter, we will give a general introduction to the deep learning tools relevant for this thesis. This includes the main architectures used for the tasks of classification, regression, and generation.

2.1. Artificial neural networks

2.1.1. The perceptron

The concept of an artificial neural network in computer science was inspired by its biological counterpart which can be found in the brain. The idea was to model units called "neurons" which receive signals from others neurons and in turn, output another signal. The first model of a neural network was introduced by Rosenblatt (1958) with the perceptron. This model is a binary linear classifier. More precisely it can be defined as a function $f : \mathbb{R}^n \to \mathbb{R}$ that maps any input vector $x \in \mathbb{R}^n$ to a real value $f(x) \in \{0, 1\}$, such that

$$f(x) = \begin{cases} 1 & \text{if } w^T x + b > 0 \\ 0 & \text{else.} \end{cases} \tag{2.1}$$

$w \in \mathbb{R}^n$ is the weight vector of the perceptron, and $b \in \mathbb{R}$ is called the bias. More precisely, we can write f as a composition of two functions : $f = H \circ g$, where $g(x) = w^T x + b$ and H is the Heaviside function. H here is the *activation function*, e.g. a function that outputs a non-zero depending on the value of the input. The idea being that the neuron "activates" only under certain combinations of input signals. Activation functions also introduce non-linearity to the model. There are other examples of activation functions, such as the rectified linear unit (ReLU) $f(x) = \max(x, 0)$, or the sigmoid $f(x) = \frac{1}{1+e^{-x}}$. ReLU also has its variants, such as Leaky ReLU: $f(x) = \max(x, 0) + \lambda \min(0, x)$ with $\lambda > 0$ a hyperparameter, and PReLU (He et al., 2015) : $f(x) = \max(x, 0) + \lambda \min(0, x)$ where this time λ is a learnable parameter.

In the case of Equation 2.1, the output of the perceptron is non-zero if and only if $w^T x > -b$: the bias here defines the threshold the linear combination of x and w has to reach to activate the neuron. However, in the case of the sigmoid activation function, the neuron can be slightly activated even if the linear combination is negative:

$w^T x + b < 0 \Rightarrow 0 < f(x) < 0.5$. Still, higher values of $w^T x + b$ will result in stronger output signals for the neuron. Fig. 2.1 shows a visual representation of the perceptron.

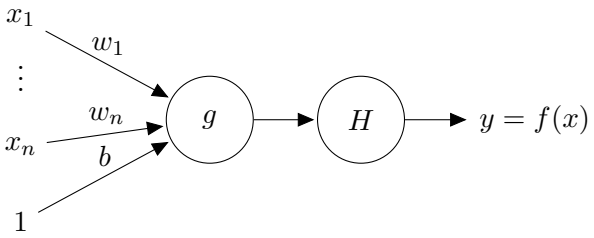


Figure 2.1.: Graphical representation of the perceptron. First, it computes the linear combination of the inputs with the weights, then the result is passed through the activation function.

Note that the perceptron separates the positive outputs from the zero outputs relative to the hyperplane $w^T x + b = 0$. A visualization of this is shown in Fig. 2.2.

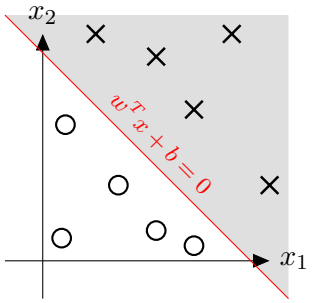


Figure 2.2.: Binary classifier in the 2D plane. The data points are classified depending on their relative position to the separating line defined by the weights w and bias b.

Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, the perceptron learns its hyperparameters $\omega = \{w, b\}$ by updating them such that every data point in $\mathcal{D}$ is classified correctly, according to the Widrow-Hoff learning rule

$$\forall i \in [\![1, N]\!], \forall j \in [\![0, n]\!], w_j \leftarrow w_j + \eta \left(y_i - f(x_i)\right) x_{i,j} \tag{2.2}$$

with $w_0 = b$ and $\forall i \in [\![1, N]\!], x_{i,0} = 1$. η is called the *learning rate* and it defines the step size of the updates. The higher the learning rate, the faster the algorithm can possibly converge, but it also increases the risk of divergence. A learning rate too small ensures convergence (when possible), at the cost of increased computation time.

This algorithm converges if and only if the dataset is linearly separable, i.e. there exists a hyperplane that separates the two classes. The perceptron was important in the sense that it introduced a mathematical model inspired biology capable of learning from data in a supervised manner (i.e. the labels y are known during training). However, even though the perceptron can be generalized to a multiclass version, by defining a weight vector for each class and selecting the maximal output, it was still underperforming when confronted to more complex datasets.

2.1.2. Multilayer perceptron

To solve the problem of non-linearly separable datasets, the main idea was to introduce so-called "hidden" layers, between the input layer x and the output layer y. This variant of the perceptron is rightfully called a *multilayer perceptron* (MLP), which is a form of *feedforward neural network* (FNN).

Let us redefine the problem of a multiclass classification. The dataset is $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ with $\forall i \in [\![1, N]\!], x_i \in \mathcal{X} \subset \mathbb{R}^n$ and $y_i \in \mathcal{Y} = \{1, \dots, C\}$. n is the dimension of the input space and C the number of classes. Like the perceptron, the MLP is a function f_ω parameterized by $\omega \in \Omega$ that aims to map any input x to its label y, i.e. $y = f_\omega(x)$.

The multilayer peceptron is defined by a series of fully-connected (or dense) layers $(l_0, ..., l_L) \in \mathbb{R}^{n_0} \times \cdots \times \mathbb{R}^{n_L}$, with $l_0 = x$, and

$$\forall k \in [\![1, L]\!], l_k = f_k(W_k\, l_{k-1} + b_k)\,. \tag{2.3}$$

The learnable parameters of the k^{th} layer are $W_k \in \mathbb{R}^{n_k \times n_{k-1}}$ and $b_k \in \mathbb{R}^{n_k}$. Finally, $(f_k)_{k=1}^{L}$ are the respective activation functions of each layer. Note that the number of weights in a given layer is proportional to the number of neurons in the previous one. With these notations, the set of learnable parameters is $\omega = \{W_1, \dots, W_L, b_1, \dots, b_L\}$.

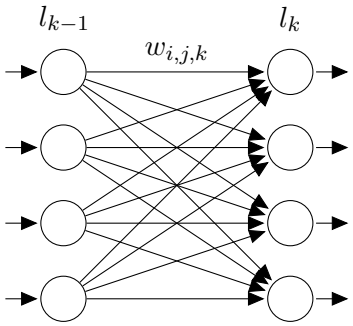


Figure 2.3.: Multilayer perceptron. Each neuron of the layer l_k is connected to the previous layer l_{k-1} through a series of weights and biases.

When dealing with a classification problem, it is generally preferable to output a predictive probability distribution $p(y \,|\, x, \omega)$, which indicates the relative confidence the model has on the prediction of x. As such, the softmax activation function is applied to the final layer, such that $\forall c \in [\![1, C]\!], f_\omega(x)_c \in [0, 1]$ and $\sum_c f(x)_c = 1$. Given a vector $z \in \mathbb{R}^C$ and a temperature parameter $T > 0$, the softmax function is defined as $\sigma(z)_i = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$. Usually the temperature parameter is set to 1 or calibrated through temperature seeking.

To learn the parameters of a MLP, we first need to define a *loss function* $\mathcal{L} : \mathbb{R}^C \times \mathbb{R}^C \to \mathbb{R}$, which quantifies how much the neural network fails to classify the data point. Common examples of loss functions include the L^2 loss $\mathcal{L}(y, y') = \|y - y'\|_2^2$ or the L^1 loss

$\mathcal{L}(y, y') = \sum_i |y'_i - y_i|$. Then, the goal of the model is to minimize the *risk* $\mathcal{R} : \Omega \to \mathbb{R}$ or expected loss

$$\mathcal{R}(\omega) := \int_{\mathcal{X}\times\mathcal{Y}} \mathcal{L}(f_\omega(x), y)\, dP(x, y)\,. \tag{2.4}$$

$P(x, y)$ is the underlying probability measure that generates the data, and it is usually unknown. As such, the integral is untractable. In order to estimate it, it is possible to use the *Monte Carlo sampling* technique (MC sampling): given a random variable x with a probability density $p(x)$, and some function g of x, the Monte Carlo estimator of $\bar{g} = \mathbb{E}_{p(x)}[g(x)]$ is

$$\hat{g}_N := \frac{1}{N}\sum_{i=1}^{N} g(x_i) \tag{2.5}$$

with $(x_i)_{i=1}^N$ being N samples from the random variable x. This estimator comes from the law of large numbers: given enough samples, the estimator will converge almost surely to the expected value[1]: $\mathbb{P}\left(\lim_{N\to+\infty} \hat{g}_N \to \bar{g}\right) = 1$.

Therefore, a common substitute to the risk is the *empirical risk*, which in the case of i.i.d. samples $(x_i, y_i)_{i=1}^N$, is estimated with MC sampling:

$$\hat{\mathcal{R}}(\omega) := \frac{1}{N}\sum_{(x,y)\in\mathcal{D}} \mathcal{L}(f_\omega(x), y)\,. \tag{2.6}$$

The goal of the training process is therefore to find the optimal solution to the empirical risk given the training data

$$\omega^* := \operatorname*{argmin}_{\omega\in\Omega} \hat{\mathcal{R}}(\omega)\,. \tag{2.7}$$

To find ω^*, we apply the stochastic gradient descent (SGD) algorithm described in Algorithm 1. The idea is to update the parameters ω following the direction of the gradient of the empirical risk. The algorithm used to update the weights given a gradient is called an optimizer. Several other optimizers exist in the literature, such as RMSProp (Tieleman and Hinton, 2012), AdaGrad (Duchi et al., 2011), or Adam (Kingma and Ba, 2015). While SGD has a fixed learning rate, these other optimizers usually introduce momentum-based optimization, with a learning rate that is time dependent.

To make the algorithm faster, the gradient is estimated on mini-batches which are randomly drawn without replacement from the dataset , until all the data points are used to update the weights. This process is called an *epoch* and the algorithm is repeated for any number of epochs T, until the algorithm has effectively converged to an optimal solution.

In practice however, it is common to consider a validation dataset $\mathcal{D}^{val}$ drawn from $P(x, y)$ and use it to evaluate the validation risk $\hat{\mathcal{R}}^{val}(\omega)$ during training. The algorithm would stop when $\hat{\mathcal{R}}^{val}(\omega)$ starts increasing and the difference with the training risk

[1] This is the strong law of large numbers, assuming that $g(x_1)$ is integrable: $\mathbb{E}(|g(x_1)|) < +\infty$.

Algorithm 1 Stochastic gradient descent

Input: Initial parameters ω, MLP f_ω, training dataset $\mathcal{D}$, batch size n_b, number of epochs T, learning rate η
Output: optimal parameters ω^*
for $t = 1$ *to* T **do**
  for $i = 1, \ldots, \lceil \frac{N}{n_b} \rceil$ **do**
    Randomly draw mini-batch $\mathcal{D}_{batch}$ of size n_b without replacement.
    Compute $\hat{\mathcal{R}}(\omega)$ on $\mathcal{D}_{batch}$
    Update parameters $\omega \leftarrow \omega + \eta \nabla_\omega \hat{\mathcal{R}}(\omega)$
  end
end
return optimal parameters $w^* = \omega$

becomes greater. This is a phenomenon known as *overfitting*, when the model loses its generalization ability and starts performing worse on unseen data coming from the same underlying distribution. An illustration of overfitting is shown in Fig. 2.4. Several techniques to prevent overfitting do exist. Adding a regularization term to the loss function helps mitigate model complexity. For instance, by using some types of loss function or regularizing terms, this will lead to a minimization of the number of non-zero weights. The new objective is then

$$\hat{\mathcal{R}}(\omega) := \frac{1}{N} \sum_{(x,y)\in\mathcal{D}} \mathcal{L}(f_w(x), y) + \lambda \operatorname{reg}(\omega)\,, \tag{2.8}$$

with $\lambda \geq 0$ and $\operatorname{reg} : \Omega \to \mathbb{R}$. Such regularization terms include the L2 regularization (Ridge Regression): $\operatorname{reg}_{L_2}(\omega) = \sum_i w_i^2$, L1 regularization (similarly to Lasso Regression): $\operatorname{reg}_{L_1}(\omega) = \sum_i |w_i|$, with $\omega = \{w_1, \ldots, w_m\}$, or Elastic net regularization which combines both Ridge and Lasso: $\operatorname{reg}_{elastic}(\omega) = \lambda_1 \sum_i |w_i| + \lambda_2 \sum_i w_i^2$ with $\lambda_1, \lambda_2 \geq 0$.

A complementary technique to prevent overfitting is by using a technique called *dropout* (Srivastava et al., 2014). When applied to a certain layer, dropout will randomly shut down neurons independently with a probability p, meaning that the neuron's value will effectively be 0, rendering it irrelevant in the optimization process. By reapplying dropout at each training step, it forces the model to not rely on the same neurons everytime, and instead forces it to learn parameters that can better generalize to a wider dataset. Formally, when using dropout on a neural network, the outputs of each layer l_k are multiplied by a binary noise

$$r_j^k \sim \mathcal{B}(1 - p_k)\, \forall j = 1, \ldots, n_{k-1} \tag{2.9}$$

$$l_{k+1} = f_k\left(W_{k+1}\,(r^{k+1} \odot l_k) + b_{k+1}\right)\,, \tag{2.10}$$

where $r^k = (r_1^k, \ldots, r_{n_{k-1}}^k)^T$ with $n_k \in \mathbb{N}$ being the numbers of neurons in layer k, and $(p_k)_{k=1}^L$ are the dropout rates in each layer. In most applications, the dropout rates are set arbitrarily to a certain value (often 0.5), or through a grid search.

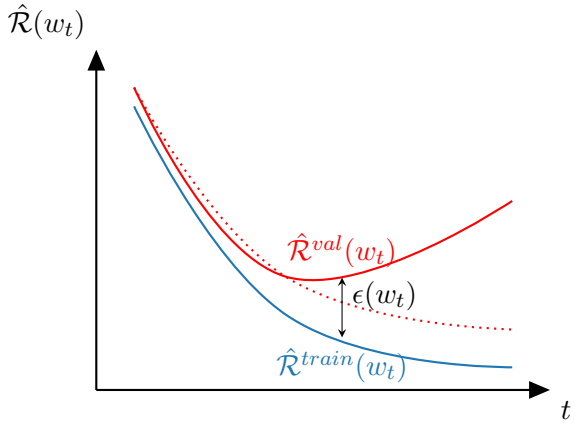


Figure 2.4.: Illustration of overfitting. As the training process progresses, the gap $\epsilon(w_t)$ between the validation set and the training set increases, reducing the generalization capability of the model. The red line represents the empirical risk on the validation data, the blue line the empirical risk on the training data. The dotted line is a desired behavior of the empirical risk on the validation data without overfitting.

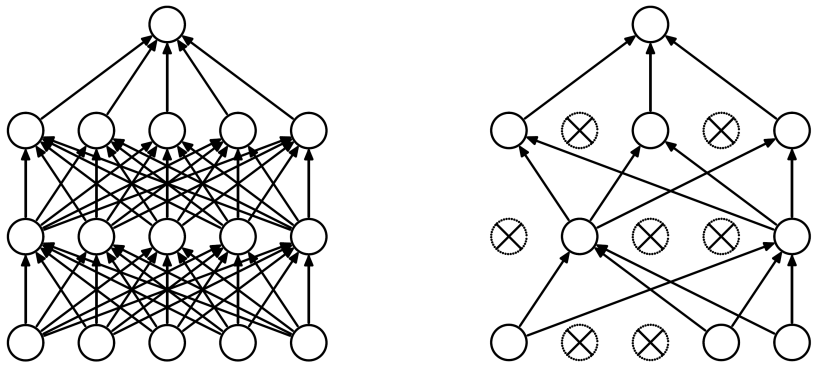
See Fig. 2.5 for a graphical illustration. Dropout in this case is only used during training, but is deactivated during inference. Dropout is also one of the main ways to make a neural network Bayesian, see Section 3.3. We will also later present a Bayesian neural network based on dropout regulation to prevent overfitting (Section 3.5).

Finally, to execute the SGD algorithm, one must compute the gradient of the empirical risk relative to the parameters of the network. Rumelhart et al. (1986) introduced the backpropagation algorithm. The idea is to compute the gradient one layer at a time, starting from the output layer, by using the chain rule.

To summarize, deep feedforward neural networks such as the MLP are capable to capture complex data distributions which are not necessarily linearly separable, by using one or multiple hidden layers, along with non-linear activation functions. The universal approximation theorem even states that a FNN with one finite hidden layer can approximate any continuous function on $\mathbb{R}^n$, given a certain amount of hidden units (Hornik, 1991; Cybenko, 1992). Supposedly, any data can be used to train a FNN. However, depending on which type of data is considered, they may be suboptimal due to inductive bias (or lack thereof).

2.2. Convolutional neural networks

Convolutional neural networks (CNN) (Fukushima, 1980; LeCun et al., 1989) are a type of neural networks that were initially designed to tackle image processing. Images are 3-dimensional data: $X \in \mathbb{R}^{h \times w \times c}$, with $(h, w, c) \in (\mathbb{N}^*)^3$ being the height, width and number of channels (3 channels for RGB images) respectively. By their nature they contain spatial information, both locally and globally. For instance, it is possible to detect



(a) Neural network without dropout

(b) Neural network with dropout

Figure 2.5.: Graphical representation of dropout. On the left, the network using dropout has random neurons deactivated, and the resulting network has fewer connections. Credits: Srivastava et al. (2014).

an edge in an object not only by looking at one pixel in the edge, but also by noticing that some neighboring pixels have a very different value, such as going from black to white when considering a handwritten digit. Spatial information is also important to detect patterns such as textures, color gradients, etc.

The main idea behind CNNs is to take advantage of this kind of spatial information in the data by introducting spatial invariance, capturing local information, and weight sharing. Instead of singular weights, a convolutional layer is comprised of filter matrices (or kernels) $(\mathbf{W}_i)_{i=1}^{C}$, $\forall i$, $\mathbf{W}_i \in \mathcal{M}_m(\mathbb{R})$ which are used to perform a convolution operation over the input, resulting in a new image called a *feature map*. Each filter is applied on every channel of the feature map separately, the idea being that certain filters can be used to learn specific patterns on the image. Activation functions are also applicable in CNNs to introduce non-linearity. The activation function is applied element-wise to each feature map.

Given an input $Y_l \in \mathbb{R}^{h_l, w_l, C_l}$ to a convolutional layer and its corresponding kernels $(\mathbf{W}_{j,i}^{l})_{i,j \in [\![1, C_{l+1}]\!] \times [\![1, C_l]\!]}$ and activation function f_l, the output $Y_{l+1} \in \mathbb{R}^{h_{l+1}, w_{l+1}, C_{l+1}}$ is expressed as

$$\forall j \in [\![1, C_{l+1}]\!],\ Y_{l+1}^{j} = f_l \left(\sum_{i=1}^{C_l} \mathbf{W}_{j,i}^{l} * Y_l^i \right) \tag{2.11}$$

where $*$ denotes the convolution operator.

By sliding the n filters along the image, we get n feature maps that are then fed through a pooling layer, which reduces their dimension. An example is the maxpooling operation: given a pooling size (k, k), the feature map is divided in patches of size (k, k) and subsamples by taking the element with maximal value in each patch. This operation reduces the width and height of the feature map by k and is applied before the activation function. An interesting property of pooling layers is their ability to preserve spatial

information. By lowering the dimension of the feature maps, it also increases the amount of pixels used to compute a given voxel in subsequent convolutional layers. This means that early layers will have a tendency to capture the fine details in the image, while late layers will learn more spatially global, high level information. See Fig. 2.6 for a graphical overview of the process.

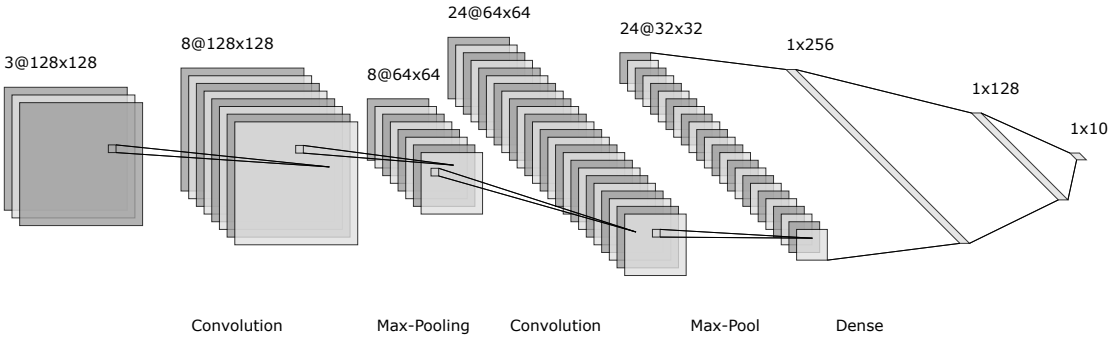


Figure 2.6.: Illustration of a CNN architecture. The image is processed through a series of convolutional and pooling layers, and then the resulting feature maps are flattened into a fully connected network, which predicts the output. This figure was created using the NN-SVG tool (LeNail, 2019).

Another example of pooling layer is a convolution with stride layer. A usual convolutional layer has a stride of 1, which means that the filter window shifts by one unit each time on the feature map. A convolutional layer with stride n shifts the window n units at a time, reducing the overall size of the output feature map, effectively dividing the size of its height and width by n. Another way to look at a convolution with stride layer is to consider it as a pooling layer that is learnable by the CNN, which can be useful in some cases.

We know that each voxel of a feature map has been computed using a subset of pixels in the original input, this subset being dependent on the filter size and pooling operation. This ensemble of pixels represents its *receptive field* (see Fig. 2.7). The size of the receptive field is recursively computed with the following equation (Araujo et al., 2019)

$$r_{l-1} = s_l\, r_l + (k_l - s_l)\,, \tag{2.12}$$

with r_l the size of the receptive field at the l^{th} layer, s_l the stride/pooling size at layer l and k_l the kernel (filter) size at layer l.

By recursively applying this relation, we obtain the size of the overall receptive field of the network r_0 (Araujo et al., 2019)

$$r_0 = \sum_{l=1}^{L} \left((k_l - 1) \prod_{i=1}^{l-1} s_j \right) + 1\,. \tag{2.13}$$

Convolutional layers have proven to be very effective in image processing and computer vision tasks (Krizhevsky et al., 2012), due to their favorable inductive bias. The weight-

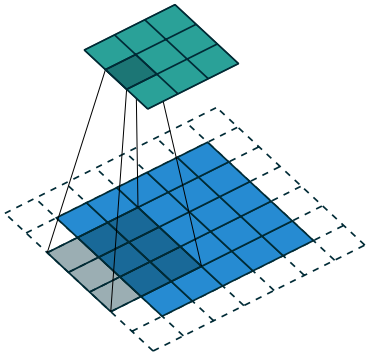


Figure 2.7.: Illustration of the receptive field of a convolutional layer. The receptive field of a voxel (coefficient of a feature map) depends on the size of the convolutional filter used for its computation, as well as the size of the stride/pooling operations. Credits: Dumoulin and Visin (2016).

sharing property of CNNs also help reduce the overall number of parameters, making them very efficient and less prone to overfitting.

However new architectures have recently proven to compete with CNNs, such as attention-based models (Vaswani et al., 2017), which ultimately led to the transformer models, like the Vision Transformer (ViT) for image processing (Dosovitskiy et al., 2021). Transformers use multiple attention layers, which learn the area of the image that is relevant for detecting particular patterns, i.e. it shows the parts of the data where the model needs to pay attention. Transformers are extremely powerful deep learning models, but they require a much higher amount of data and are more costly to train, as they feature much more parameters than a usual CNN. Transformers effectively need to learn the spatial information in the image that is made easier to capture in a CNN due to their image-specific inductive bias. As such, for the remainder of this thesis, we will keep using CNNs as they are easier to manipulate and implement, and are still widely used in the machine learning community.

2.3. Generative models

So far, we have seen examples of neural networks suitable for supervised or unsupervised learning. Another useful type of deep learning models to consider are *generative models*, which will be used in later chapters in the context of counterfactual explanations. A generative model is a function $\mathcal{G}_\theta : \mathbb{R}^k \to \mathbb{R}^n$ that aims to generate samples from the data distribution $p(x)$ with $x \in \mathbb{R}^n$ using an underlying latent space $\mathcal{Z} \subset \mathbb{R}^k$.

The goal of each generative model is to maximize the likelihood $p(x \,|\, \theta)$ with respect to θ. Explicit density methods to solve the maximum likelihood optimization aims at either learning the true distribution (exact density), or learning an approximation of the true distribution (approximate density).

Approximate density approaches to generative models include *variational autoencoders* (VAEs), which uses a variational posterior distribution; and *diffusion models*, which uses a Markov chain. On the other hand, an example of explicit density models are *normalizing flows*, which allows to have an explicit representation of the PDF, meaning that not only it is possible to sample from the true distribution but also compute the PDF for any value.

Finally, on the opposite side of explicit density methods, implicit density techniques do not try to obtain a PDF, but rather directly learns to generates data using a likelihood-free objective function. Such implicit density generative models include *Generative adversarial networks* (GANs). See Fig. 2.8 for a graphical representation of the classification of generative models.

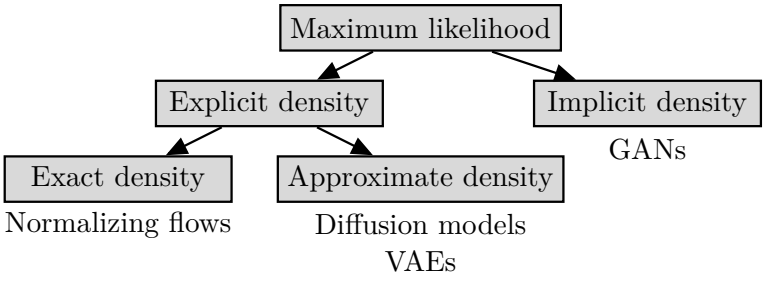


Figure 2.8.: Classification of generative models.

We will know describe each of the different generative models in more detail.

2.3.1. Variational autoencoders

Simple autoencoders and, later on,*variational autoencoders* (Kingma and Welling, 2014), or VAEs, define a type of approximate density generative model that is decomposed in two parts: the encoder and the decoder.

In the case of a simple autoencoder, we have two functions: the encoder (or recognition model) $e_\phi : \mathcal{X} \to \mathcal{Z}$, parametrized by ϕ, which encodes the data into the latent space $\mathcal{Z}$; and the decoder (or generative model) $d_\theta : \mathcal{Z} \to \mathcal{X}$, parametrized by θ, which maps the latent space into the input space (see Fig. 2.9). These two functions are defined such that, optimally, we would have $x \approx d_\theta(e_\phi(x))$: given an input x, the decoder should be able to reproduce it from its latent representation $z = e_\phi(x)$. Since $\mathcal{Z}$ is usually defined as a subset of a vector space of dimension inferior to the ambient vector space of the input space, an auto-encoder can be considered as a compression scheme.

The autoencoder is trained according to the following optimization objective, by minimizing the information lost in the compression-decompression process

$$\mathcal{L}(x; \theta, \phi) = ||x - d_\theta(e_\phi(x))||^2 . \tag{2.14}$$

However, autoencoders have some issues. First of all, auto-encoders tend to overfit the latent space, as there is no regularization made during training to ensure the latent

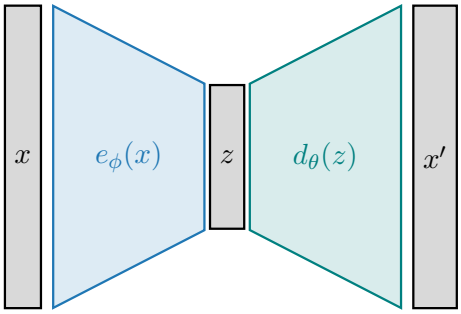


Figure 2.9.: Architecture of a simple autoencoder. The encoder computes a latent representation z from x. Then decoder uses z to generate x' to be as close as possible to x.

space has a coherent structure. More precisely, latent space learned by an auto-encoder tend to lack continuity, meaning that two neighbor latent points z and z' may have widely different decodings: $d_\theta(z) \neq d_\theta(z')$. Secondly, the latent space cannot be used for generation, as random points sampled from the latent space may represent meaningless information once decoded, meaning that they are very unlikely to have been generated by the true data distribution $p(x)$.

Variational autoencoders aim to regularize the latent space to ensure continuity and enable sampling of the latent space by mapping inputs x to a probability distribution $p(z \,|\, x)$ rather than a single point estimate. More precisely, consider a latent variable model with a joint distribution $p(x, z \,|\, \theta)$ and parameters θ. We suppose that we have a prior distribution $p(z)$ on the latent variable, such that $p(x, z \,|\, \theta) = p(x \,|\, z, \theta)p(z)$. The marginal log-likelihood (or *evidence*) of the data is given by

$$\log p(x \,|\, \theta) = \log \mathbb{E}_{p(z)}[p(x \,|\, z, \theta)] \,. \tag{2.15}$$

This is the quantity that the VAE seeks to maximize, as we want the parameters θ that best describe the data distribution. Unfortunately, this expectation over the latent distribution is very often untractable. To get around this issue, and to avoid costly computations from sampling algorithms (such as MCMC sampling), the solution is to use *variational inference* (Jordan et al., 1999). The idea is simple: use a tractable approximate variational distribution $q_\phi(z \,|\, x)$ parameterized by ϕ such that $q_\phi(z \,|\, x\,) \approx p(z \,|\, x, \theta)$. In this case, ϕ represents the parameters of the encoder and θ the parameters of the decoder. See Fig. 2.10 for an illustration of variational inference.

A metric that can be used to optimize the similarity between two probability distributions is the *Kullback-Leibler divergence* (or KL divergence). Given $q_\phi(z \,|\, x)$, the KL divergence between the variational posterior and $p(z \,|\, x, \theta)$ is defined as

$$D_{KL}(q_\phi(z \,|\, x) \,||\, p(z \,|\, x, \theta)) := \mathbb{E}_{q_\phi(z \,|\, x)} \left[\log \frac{q_\phi(z \,|\, x)}{p(z \,|\, x, \theta)} \right] . \tag{2.16}$$

This divergence quantifies the expected difference of bits necessary to encode samples from $p(z \,|\, x, \theta)$ by using $q_\phi(z \,|\, x)$ rather than $p(z \,|\, x, \theta)$ itself. Note that the KL divergence is not a distance as it is not symmetric, i.e. it is possible to find two distributions p, q

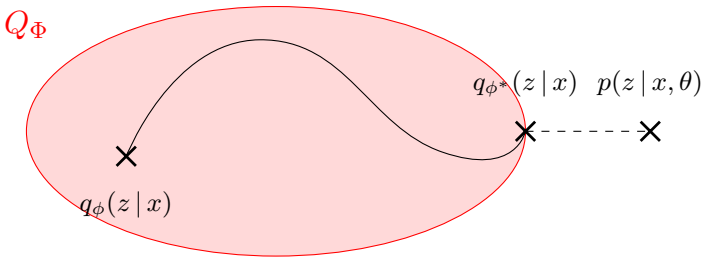


Figure 2.10.: Illustration of variational inference. The initial variational distribution $q_\phi(z \mid x)$ is optimized to the nearest distribution $q_{\phi^*}(z \mid x)$ of the true distribution $p(z \mid x, \theta)$ within the set of variational distributions Q_Φ.

such that $D_{KL}(q \,||\, p) \neq D_{KL}(p \,||\, q)$. Still, it has the property of $D_{KL}(q \,||\, p) = 0$ iff $p = q$ almost surely.

We can further develop the expression of the KL divergence in Equation 2.16 to get

$$\begin{aligned} D_{KL}\left(q_\phi(z \mid x) \,||\, p(z \mid x, \theta)\right) &= \mathbb{E}_{q_\phi(z \mid x)}\left[\log \frac{q_\phi(z \mid x)}{p(z \mid x, \theta)}\right] \\ &= \mathbb{E}_{q_\phi(z \mid x)}\left[\log \frac{q_\phi(z \mid x)\, p(x \mid \theta)}{p(x \mid z, \theta)\, p(z)}\right] \\ &= D_{KL}(q_\phi(z \mid x) \,||\, p(z)) - \mathbb{E}_{q_\phi(z \mid x)}[\log p(x \mid z, \theta)] + \log p(x \mid \theta)\,. \end{aligned} \tag{2.17}$$

By isolating the log-likelihood in Equation 2.17, we get

$$\begin{aligned} \log p(x \mid \theta) &= D_{KL}\left(q_\phi(z \mid x) \,||\, p(z \mid x, \theta)\right) + \mathbb{E}_{q_\phi(z \mid x)}\left[\log p(x \mid z, \theta)\right] - D_{KL}\left(q_\phi(z \mid x) \,||\, p(z)\right) \\ &\geq \mathbb{E}_{q_\phi(z \mid x)}\left[\log p(x \mid z, \theta)\right] - D_{KL}\left(q_\phi(z \mid x) \,||\, p(z)\right)\,. \end{aligned} \tag{2.18}$$

Since $D_{KL}\left(q_\phi(z \mid x) \,||\, p(z \mid x, \theta)\right)$ is untractable, we need to introduce the lower bound in Equation 2.18. This lower bound is called the *evidence lower bound* (ELBO), and is defined according to Equation 2.18 as

$$\mathcal{L}_{ELBO}(\theta, \phi; x) := \mathbb{E}_{q_\phi(z \mid x)}\left[\log p(x \mid z, \theta)\right] - D_{KL}\left(q_\phi(z \mid x) \,||\, p(z)\right)\,. \tag{2.19}$$

Maximizing the ELBO will therefore ensure that the lower bound on the marginal-likelihood will be higher.

As seen in Equation 2.19, the ELBO is decomposed in two terms. The first term of the ELBO, $\mathbb{E}_{q_\phi(z \mid x)}\left[\log p(x \mid z, \theta)\right]$, is the expected value of the log-likelihood on the variational posterior, and it measures how well the approximate variational posterior explains the data distribution. It can be interpreted as a reconstruction term, as maximizing this quantity will increase the likelihood of the decoded data under the variational posterior.

The second term, $-D_{KL}\left(q_\phi(z \mid x) \,||\, p(z)\right)$, is the KL divergence between the variational posterior and the prior $p(z)$. This term will encourage the variational posterior to not

diverge too far from the prior, and it can be considered as a regularization term. For instance, if the prior $p(z)$ follows a normal distribution $p(z) \sim \mathcal{N}(0, 1)$, the KL term will force $q_\phi(z \mid x)$ to be similar to the standard Gaussian distribution, which prevents it to collapse to a Dirac distribution (which is equivalent to what normal auto-encoders do).

This ELBO will be used to optimize the parameters of both the encoder and decoder of the VAE, by using a gradient descent algorithm. However, the gradient of the ELBO is still impractical because it requires to compute the gradient of a probability distribution, which is a stochastic quantity. Several estimators of this gradient were proposed to solve this issue. For instance, consider the gradient of the reconstruction term

$$\begin{aligned}
\nabla_\phi \mathbb{E}_{q_\phi(z \mid x)} \left[\log p(x \mid z, \theta)\right] &= \nabla_\phi \int_{\mathcal{Z}} \log p(x \mid z, \theta)\, q_\phi(z \mid x)\, dz \\
&= \int_{\mathcal{Z}} \log p(x \mid z, \theta)\, \nabla_\phi q_\phi(z \mid x)\, dz \\
&= \int_{\mathcal{Z}} \log p(x \mid z, \theta)\, \frac{\nabla_\phi q_\phi(z \mid x)}{q_\phi(z \mid x)}\, q_\phi(z \mid x)\, dz \\
&= \int_{\mathcal{Z}} \log p(x \mid z, \theta)\, \nabla_\phi \log q_\phi(z \mid x)\, q_\phi(z \mid x)\, dz \\
&= \mathbb{E}_{q_\phi(z \mid x)} \left[\log p(x \mid z, \theta)\, \nabla_\phi \log q_\phi(z \mid x)\right] . \qquad (2.20)
\end{aligned}$$

This estimator of the gradient is called the *score function* estimator or *Reinforce* (Glynn, 1990; Paisley et al., 2012). In practice, we would estimate the expected value by some form of sampling technique, as the full integral over the variational distribution is very often untractable. Usually, the technique would be MC sampling, but in the case of Equation 2.20, estimating this expectation with the MC estimator would be unreliable, as this gradient estimator is known to have high variance.

The solution is to introduce the so-called *reparametrization trick* (Kingma and Welling, 2014). The idea is to write $z \sim q_\phi(z|x)$ by introducing a differentiable transformation $g_\phi(\epsilon, x)$ with $\epsilon \sim p(\epsilon)$ being a noise variable: $z = g_\phi(\epsilon, x)$. Given that the relationship between z and ϵ is deterministic, it follows that $q_\phi(z \mid x)\, dz = p(\epsilon)\, d\epsilon$. Therefore, by applying the variable change $z = g_\phi(\epsilon, x)$ in Equation 2.20, we have

$$\begin{aligned}
\nabla_\phi \mathbb{E}_{q_\phi(z \mid x)} \left[\log p(x|z, \theta)\right] &= \nabla_\phi \int \log p(x \mid z, \theta)\, q_\phi(z \mid x)\, dz \\
&= \nabla_\phi \int \log p(x \mid g_\phi(x, \epsilon), \theta)\, p(\epsilon)\, d\epsilon \\
&= \int \nabla_\phi \log p(x \mid g_\phi(x, \epsilon), \theta)\, p(\epsilon)\, d\epsilon \\
&= \mathbb{E}_{p(\epsilon)} \left[\nabla_\phi \log p(x \mid g_\phi(x, \epsilon), \theta)\right] , \qquad (2.21)
\end{aligned}$$

which results in a gradient that does not need to be sampled from, since the parameter ϕ is deterministic. This estimator of the gradient now has a much lower variance than the previous one. Hence, this expected value is in practice estimated with MC sampling.

Usually, in the case of a VAE, it is assumed that $q_\theta(z \mid x)$ is a multivariate Gaussian distribution with a diagonal covariance matrix, i.e. $z \sim q_\phi(z \mid x) = \mathcal{N}(\mu, \sigma^2 I)$. Then, the reparametrization trick gives, $z = \mu + \sigma \odot \epsilon$ where $\epsilon \sim \mathcal{N}(0, I)$. Plugging this into Equation 2.21 yields a quantity that can be estimated to optimize the VAE with low variance.

Finally, the encoder and decoder are implemented as neural networks (usually CNNs for images) $f_\phi(x)$ and $\mathcal{G}_\theta(z)$ respectively. The parameters of this multivariate Gaussian distribution are predicted by the encoder, i.e. $\mu, \sigma = f_\phi(x)$. Then, we compute $z = \mu + \sigma \odot \epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$ and the decoder attempts to reconstruct the original data x from z: $\mathcal{G}_\theta(z) = x'$.

By maximizing the ELBO, and indirectly the marginal likelihood $p(x \mid \theta)$, the encoder and decoder are brought to estimate respectively the latent distribution $p(z \mid x, \theta)$ and the conditional likelihood $p(x \mid z, \theta)$. The distribution $p(x \mid z, \theta)$ can either be a multivariate Gaussian distribution (in the case of real-valued data), or a Bernoulli distribution for a binary output.

In Fig. 2.11, we show the typical architecture of a VAE that summarizes the whole process of encoding, reparametrization trick, and decoding.

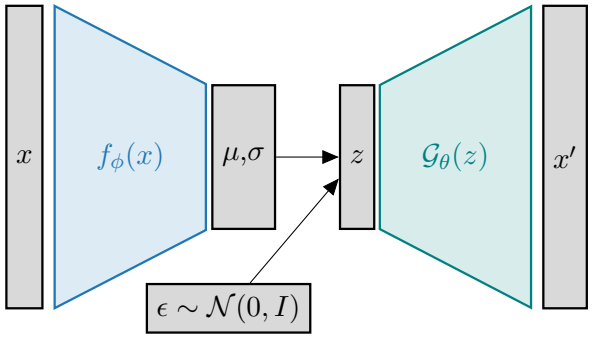


Figure 2.11.: Architecture of a VAE. The encoder computes the parameters of the multivariate Gaussian from the original input x, then z is sampled using the reparametrization trick thanks to the noise variable ϵ. Finally, the decoder uses z to generate x' to be as close as possible to x.

While VAEs are useful due to their ability to encode data points into a latent space, they are not without issues. For instance, in terms of image processing, the generated images by a VAE are notoriously blurry, an artifact due to its regularization term. Some variants of the VAE include the β-VAE (Higgins et al., 2017), which adds a coefficient β in regularization term of the ELBO to put more or less emphasis on it. β-VAEs with lower values of β will produce more accurate images but may overfit. Loaiza-Ganem and Cunningham (2019) proposed to use a continuous Bernoulli distribution for the conditional likelihood $p(x|z)$, which helped to produce sharper images.

Wasserstein auto-encoders (Tolstikhin et al., 2018) use a regularization term between the prior $p(z)$ and the marginalized variational posterior $q_\phi(z) = \mathbb{E}_{p(x)}[q_\phi(z \mid x)]$, which helps in the reconstruction of the latent codes. Wasserstein auto-encoders are especially

efficient when the ground truth distribution $p(z)$ and $q_\theta(z \mid x)$ have disjoint supports.

Finally, Dai and Wipf (2019) use a 2-stage VAE, by consecutively training a usual VAE and a second VAE which maps the latent space $\mathcal{Z}$ to another latent space $\mathcal{U}$ using the same latent dimension. By training the second VAE by sampling latent variables $(z_i)_{i=1}^m$, $m \in \mathbb{N}$, using $q_\phi(z \mid x_i)$, they effectively sample them from $q_\phi(z)$ by marginalization. The results were shown to reduce blurriness in the generated images.

2.3.2. Generative adversarial networks

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) are generative models based on an implicit density approximation of the ground truth. It is defined by two neural networks in competition in a minimax game with each other. The generator $\mathcal{G}_\theta : \mathbb{R}^k \to \mathbb{R}^n$ aims to create samples from the data distribution $p(x)$ using a latent space $\mathcal{Z} \subset \mathbb{R}^k$. On the other hand, the discriminator $\mathcal{D}_\phi : \mathbb{R}^n \to [0, 1]$ is a classifier that, given an input $x \in \mathbb{R}^n$, tries to predict if it was produced by the generator or if it is a real data point sampled from $p(x)$. Formally, the objective of the GAN is described as

$$\min_\theta \max_\phi V(\theta, \phi) = \min_\theta \max_\phi \mathbb{E}_{p(x)}[\log \mathcal{D}_\phi(x)] + \mathbb{E}_{p(z)}[\log(1 - \mathcal{D}_\phi(\mathcal{G}_\theta(z)))] \,. \tag{2.22}$$

Fig. 2.12 shows a graphical representation of the GAN architecture.

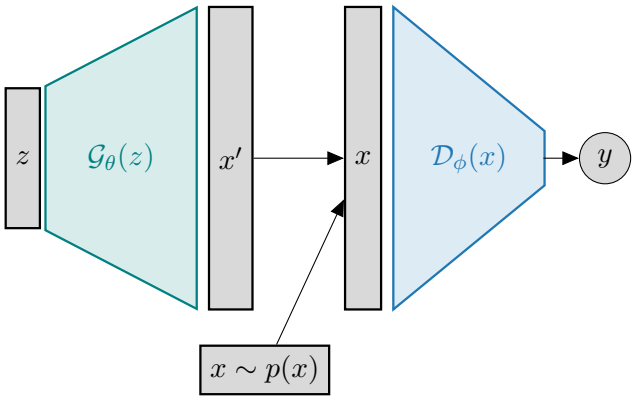


Figure 2.12.: Architecture of a GAN. On the left, the generator takes a latent variable $z \sim p(z)$ and generates a sample x'. On the right, the discriminator takes an input x which can either be sampled from $p(x)$ or sampled from the generator, and outputs y which predicts whether x is real or not.

GANs achieve impressive results in terms of reconstruction, generating images that can look very realistic and are very sharp. Contrary to VAEs however, vanilla GANs are unable to encode an input x into the latent space. However several architectures were proposed in the literature to solve this issue.

Larsen et al. (2016) introduced the VAE-GAN by jointly training a GAN with a VAE, and sharing the parameters between the decoder of the VAE and the generator of the

GAN. The loss is comprised of the regularization term of the VAE, and the GAN objective

$$\mathcal{L} = \mathcal{L}_{prior} + \mathcal{L}_{GAN} + \mathcal{L}^{\mathcal{D}_l}_{llike} \tag{2.23}$$

with

$$\mathcal{L}_{prior} = D_{KL}(q(z \,|\, x) \,||\, p(z)) \tag{2.24}$$
$$\mathcal{L}_{GAN} = \log \mathcal{D}(x) + \log(1 - \mathcal{D}(\mathcal{G}(z)))\,. \tag{2.25}$$

The reconstruction term of the loss of the VAE is replaced by a measure of realism by the discriminator. More precisely, consider the l^{th} layer of the discriminator $\mathcal{D}_l(x)$ for the real data point $x \sim p(x)$. By introducing a Gaussian distribution $p(\mathcal{D}_l(x) \,|\, z) = \mathcal{N}(\mathcal{D}_l(\mathcal{G}_\phi(z)), I)$ with $\tilde{x} = \mathcal{G}_\phi(z)$, $z \sim q(z)$, the reconstruction error of the VAE can be replaced by

$$\mathcal{L}^{\mathcal{D}_l}_{llike}(x) = -\mathbb{E}_{q(z|x)}[\log p(\mathcal{D}_l(x) \,|\, z)]\,. \tag{2.26}$$

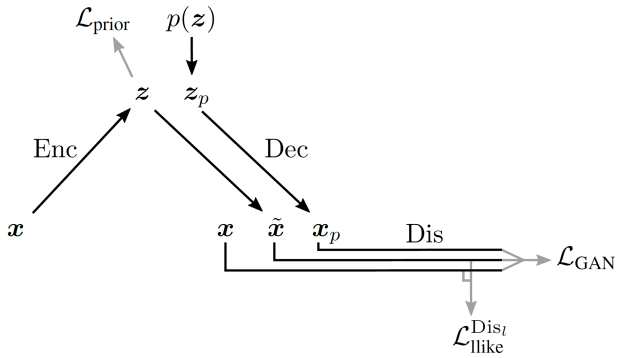


Figure 2.13.: Graphical representation of a VAE-GAN. Credits: Larsen et al. (2016).

The difference between $\mathcal{L}^{\mathcal{D}_l}_{llike}(x)$ with the usual reconstruction term of a VAE is that the VAE-GAN will rather compare the reconstruction of any input with regards to how it is interpreted by the discriminator, which helps train the discriminator too instead of the decoder alone.

Fig. 2.13 summarizes the architecture of the VAE-GAN and how the different parts of the loss are computed.

This architecture achieves results competitive with the usual GAN, while retaining the ability to encode data. More recently, even more advanced GAN architectures have been proposed with the ability to encode instances into the latent space, such as BiGAN (Donahue et al., 2017) and AttGAN (He et al., 2017) .

Another problem with GANs is that they are notoriously difficult to train, as they are prone to mode collapse. Basically, it means that GANs can fail to generalize entire modes of the data distribution, and only generate data from one or a few modes. This issue is still unresolved to this day.

2.3.3. Normalizing flows

Normalizing flows (Rezende and Mohamed, 2015) are a class of generative models that aims to map a simple probability distribution $q_0(z_0)$ to a more complex distribution $p(x)$, using a series of invertible and differentiable functions $(f_k)_{k=1}^K$. The goal is to learn the true distribution, making normalizing flows an exact density generative process. More precisely, the series of transformations, called the flow, is

$$x = z_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(z_0)\,. \tag{2.27}$$

The idea to compute the density $q_K(z_K)$ with respect to the simple density $q_0(z_0)$ is to successively use a variable change from z_K to z_0. Consider for instance, two random variables z and $z' \in \mathbb{R}^n$ such that $z' = f(z)$ with $f : \mathbb{R}^n \to \mathbb{R}^n$ a continuously differentiable (i.e. of class C^1) and invertible function. The distribution $q(z')$ is expressed as

$$q(z') = q(z) \left| \det \frac{\partial f^{-1}}{\partial z'} \right| = q(z) \left| \det \frac{\partial f}{\partial z} \right|^{-1}\,. \tag{2.28}$$

The last equality comes from the inverse function theorem, i.e. the Jacobian of the inverse function f^{-1} is invertible and its inverse is the Jacobian of the original function f. Using this fact, we can apply Equation 2.28 K times to get the relationship between the densities $q_K(z_K)$ and $q_0(z_0)$ as follows

$$q_K(z_K) = q_0(z_0) \prod_{k=1}^{K} \left| \det \frac{\partial f_k}{\partial z_{k-1}} \right|^{-1}\,. \tag{2.29}$$

Finally, it is possible to take the log on both sides of the equation

$$\log q_K(z_K) = \log q_0(z_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial z_{k-1}} \right|\,. \tag{2.30}$$

This is the equation which allows us to compute the exact density $q_K(z_K)$ from the simple density $q_0(z_0)$ and is at the heart of normalizing flows.

An illustration of a normalizing flow is shown in Fig. 2.14, showing the series of invertibles transformations and their respective distributions.

The main problem now is to define which series of invertible and differentiable functions $(f_k)_{k=1}^K$ can be used to learn the normalizing flow. Dinh et al. (2015) proposed to use coupling flows. The input is divided by using an arbitrary partitioning of the input space: $z = (z_A, z_B)$. Then, the family of invertible functions are defined as

$$f(z) = (z_A, z_B + h_\lambda(z_A))\,, \tag{2.31}$$

$$f^{-1}(z') = (z'_A, z'_B - h_\lambda(z'_A))\,. \tag{2.32}$$

h_λ can be any arbitrary function, but in practice it is a neural network parameterized by λ.

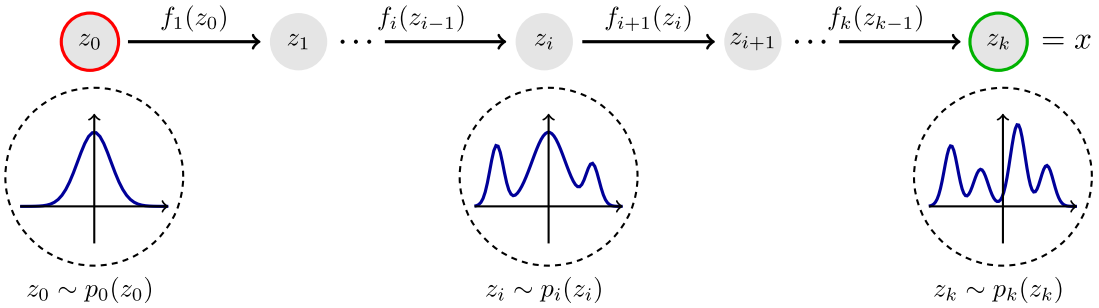


Figure 2.14.: Illustration of a normalizing flow. A simple distribution $p_0(z_0)$ is transformed into a more complex distribution $p_K(z_K)$ through a series of invertible functions. Credits: Janosh Riebesell, Lilian Weng.

The advantage of normalizing flows are their ability to perform variational inference to asymptotically approximate the true posterior distribution, given the flow has enough parameters. However they do need specific architectures to ensure the flow is reversible, which limits the width of possible models that can be used as a normalizing flow.

2.3.4. Diffusion models

Finally, diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021) are a type of approximate density models that learn to generate data from random noise, a process inspired by non-equilibrium thermodynamics. To train a diffusion model, the data (typically images) is slowly corrupted by noise (such as Gaussian noise), creating a series of images that are more and more noisy. Then, a neural network learns to revert the noising process at each step, and the entire pipeline of networks ultimately learns to generate data from pure, random noise.

More precisely, considering a series of latent variables $x_1, \ldots, x_T$, where each variable shares the same dimensionality as the original data $x_0 \sim q(x_0)$. The diffusion process is defined as a Markov chain that progressively adds noise to the data

$$\begin{aligned} q(x_1, \ldots, x_T \,|\, x_0) &:= \prod_{t=1}^{T} q(x_t \,|\, x_{t-1})\,, \\ q(x_t \,|\, x_{t-1}) &:= \mathcal{N}(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t\, I)\,. \end{aligned} \tag{2.33}$$

On the other hand, the reverse process slowly denoises the data

$$\begin{aligned} p(x_0, \ldots, x_T) &:= p(x_T) \prod_{t=1}^{T} p(x_{t-1} \,|\, x_t)\,, \\ p(x_{t-1} \,|\, x_t) &:= \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t\, I)\,. \end{aligned} \tag{2.34}$$

$(\beta_t)_{t=1}^T$ and $(\sigma_t)_{t=1}^T$ are hyperparameters of the diffusion model, while $\mu_\theta(x_t, t)$ is a function using a neural network parameterized by θ that is trained to perform the reverse process. Fig. 2.15 shows both the diffusion and reverse process applied on an image in the form of a graphical model.

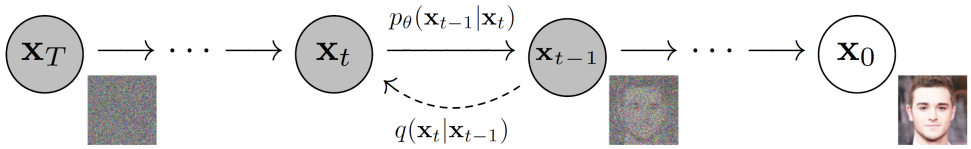


Figure 2.15.: Illustration of a diffusion model in the form of a graphical model. Credits: Ho et al. (2020).

Diffusion models are another example of very powerful generative models, the state-of-the-art being able to compete with other generative models such as GANs in image generation (Dhariwal and Nichol, 2021; Rombach et al., 2022). However, diffusion models can prove to be computationally expensive, as the sampling process rely on a possibly very long Markov chain.

In this chapter, we introduced types of models and architectures that will be used in all of the contributions. In the next chapter, CNNs will be used in the context of Bayesian Deep Learning to estimate different types of uncertainties. Chapter 4 more precisely will use CNNs to estimate galaxy ellipticities with uncertainty estimates. Generative models, and VAEs more precisely, will be more widely used in chapters 5 and 6 for counterfactual explanations.

Chapter 3.

Uncertainty in deep learning

In the following chapter, we will introduce the notions of uncertainty in machine learning, and how they can be used to quantify uncertainty in machine learning. The chapter is presented as follows: first, we will introduce the notions of epistemic and aleatoric uncertainties. This will be followed by a presentation of Bayesian Deep Learning, and the several methods used to approximate Bayesian Neural Networks. Then, we will show how uncertainties can be estimated in the context of Bayesian deep learning. Finally, we will describe our contribution: a Bayesian Neural Network based on Dropout Regulation.

3.1. Epistemic and aleatoric uncertainties

As we have seen in the previous chapter, deep neural networks have proven to be very competitive in many machine learning tasks. However, while they give unprecedented levels of performance, it may not be necessarily enough. In many domains of applications, such as biology, chemistry, medicine, autonomous driving, and in the case of this thesis, astrophysics, reliable uncertainty measurements in addition to the precision given by the models is critical. For instance, in 2016 a Tesla car driving in autonomous mode crashed into a white truck on a sunny day because it mistook the truck for the bright sky, resulting in the first casualty by an autonomous system based on a machine learning model (NHTSA, 2017).

The problem is to define a way to not only know the prediction of a machine learning model, but also how certain it is of its prediction. To solve this, we first need to define two types of uncertainties: *aleatoric uncertainty* and *epistemic uncertainty*.

Aleatoric uncertainty defines the known unknowns. It is the uncertainty related to the data itself due to random influencing but unobserved factors, and in that sense this uncertainty is not reducible. For instance, we can take an image and slowly add Gaussian noise to it, and at a certain step the model will stop reliably predicting the result. It does not matter how complex the model is or how large the dataset is: if the information in the data is insufficient to make a correct prediction, then this aleatoric uncertainty will be high with no way to reduce it unless the data is altered. Noise is only one possible source of aleatoric uncertainty. Another way to look at this is to consider the fact that the relationship between the input space $\mathcal{X}$ and output space $\mathcal{Y}$ is very often non-deterministic, leaving uncertainty in the data distribution $p(x, y)$ even if it has been perfectly learned (Hüllermeier and Waegeman, 2021). For instance, take Fig. 3.1a.

While the model has learned a separation between the two classes, there is a residual aleatoric uncertainty near the border between the two classes as they overlap, which is impossible to resolve even with more knowledge about the data distribution.

On the other hand, epistemic (or model) uncertainty quantifies the unknown unknowns. Instead of uncertainty coming from the data, it represents the lack of knowledge the model has on the data distribution $p(x, y)$. Typically, a machine learning model will exhibit high epistemic uncertainty when dealing with outlier data. A simple example of that is considering a model that has learned to differentiate between images of cats and dogs, and looking at what happens when suddenly the input image is a picture of a rare species of cat, or a stylized picture of a cat like a cartoon ? These are examples of pictures that the model has most likely not seen during training. Therefore, to reduce epistemic uncertainty, one can look for datasets that better cover the distribution $p(x, y)$.

To explain epistemic uncertainty further, consider Equation 2.7. Given our model parameterized by ω, the goal of training our model is to find an optimal parameterization given the space of possible parameters Ω and the empirical risk $\hat{\mathcal{R}}$. We shall call this optimal parameterization ω^*_{emp}. This is different from the parameterization that minimizes the true risk

$$\omega^*_{true} := \underset{\omega\in\Omega}{\operatorname{argmin}}\, \mathcal{R}(\omega) = \underset{\omega\in\Omega}{\operatorname{argmin}} \int_{\mathcal{X}\times\mathcal{Y}} \mathcal{L}(f_\omega(x), y)\, dP(x, y)\,. \tag{3.1}$$

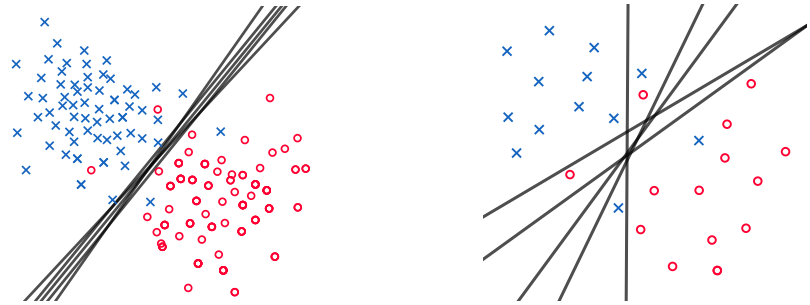
Because $\hat{\mathcal{R}}(\omega)$ is an estimator of $\mathcal{R}(\omega)$, the result of the learning algorithm, ω^*_{emp}, won't necessarily match ω^*_{true}, especially when the learning dataset is too small to be representative of the true distribution. The uncertainty due to this approximation is a source of epistemic uncertainty, called *approximation uncertainty* (Hüllermeier and Waegeman, 2021). This is the main source of epistemic uncertainty that is captured by Bayesian inference (see Section 3.2).

Finally, another source of epistemic uncertainty is *model uncertainty*, which is related to the ability of the parameter space Ω to approximate the true pointwise Bayes predictor

$$f^*(x) := \underset{\hat{y}\in\mathcal{Y}}{\operatorname{argmin}} \int_{\mathcal{Y}} \mathcal{L}(y, \hat{y}) dP(y \mid x)\,. \tag{3.2}$$

$f_{\omega^*_{true}}(x)$ and $f^*(x)$ do not necessarily represent the same function. This uncertainty due to the choice of the type of model is called model uncertainty (Hüllermeier and Waegeman, 2021), and is the generalization error in statistical machine learning. This form of uncertainty is epistemic in the sense that the choice of a class of models is arbitrary and somewhat dependent on our knowledge. However it is not reducible unless considering model selection so that it won't be considerer further.

In Fig. 3.1, we present two cases of epistemic uncertainty: Fig. 3.1a shows a case of low epistemic uncertainty due to a large dataset, whereas Fig. 3.1b highlights an example with fewer data points, and higher epistemic uncertainty.



(a) Low epistemic uncertainty (b) High epistemic uncertainty

Figure 3.1.: Difference between low and high epistemic uncertainty. Left: the amount of data points make the variance in the model's decision boundary low, meaning that the epistemic uncertainty is low. However, residual aleatoric uncertainty makes the prediction around the boundary ambiguous, as the two classes overlap. Right: the low amount of data points induces a high variance in the model's decision boundary, therefore the epistemic uncertainty is high.

3.2. Bayesian deep learning

Bayesian deep learning is motivated by the goal of estimating not only the aleatoric uncertainty of a deep learning model, but also its epistemic uncertainty. Consider a neural network $f_\omega : \mathcal{X} \to \mathcal{Y}$. In a non-Bayesian setup, the model outputs a point-wise predictive posterior distribution $p(y \mid x, \omega)$. As we have seen in Section 3.1, part of the epistemic uncertainty comes from the parameter space Ω. By taking a point-wise estimation in Ω, we exclude the estimation of this source of uncertainty on the predictive distribution.

Bayesian neural networks (MacKay, 1992; Neal, 1995), or BNNs, replace point-wise estimations of the weights ω and instead assume a prior distribution $p(\omega)$ on them (see Fig. 3.2 for a graphical illustration). Typically, the prior distribution is assumed to be Gaussian. Given the dataset $\mathcal{D}$, the goal is to estimate the posterior distribution $p(\omega \mid \mathcal{D})$. This would allow us to estimate the posterior predictive distribution by marginalizing over the posterior

$$p(y \mid x, \mathcal{D}) = \int_\Omega p(y \mid x, \omega)\, p(\omega \mid \mathcal{D})\, d\omega\,. \tag{3.3}$$

Given Bayes' rule, the posterior distribution $p(\omega \mid \mathcal{D})$ is related to the prior $p(\omega)$ by the relation

$$p(\omega \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \omega)\, p(\omega)}{p(\mathcal{D})}\,. \tag{3.4}$$

The problem comes from the fact that Equation 3.4 (and 3.3) is very often analytically impossible to compute. As such, it is necessary to find a way to approximate the posterior. This is usually done by a variational Bayes optimization to derive an approximate posterior $q_\theta(\omega)$, the same method used by variational auto-encoders (see Section 2.3.1).

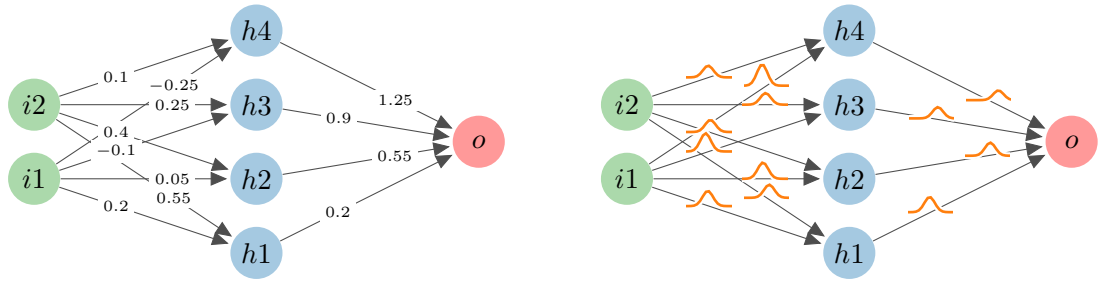


Figure 3.2.: Graphical representation of a Bayesian Neural Network. On the left: non-Bayesian neural network with point-estimate parameters. Right: Bayesian neural network which assumes and learns a probability distribution on the parameters. Credits: Janosh Riebesell.

More precisely, the variational posterior $q_\theta(\omega)$ is sought by minimizing the KL divergence with the posterior distribution

$$
\begin{aligned}
D_{KL}(q_\theta(\omega) \,||\, p(\omega \,|\, \mathcal{D})) &= \int_\Omega q_\theta(\omega) \log \frac{q_\theta(\omega)}{p(\omega \,|\, \mathcal{D})} \, d\omega \\
&= \int_\Omega q_\theta(\omega) \log \frac{q_\theta(\omega)\, p(\mathcal{D})}{p(\mathcal{D} \,|\, \omega)\, p(\omega)} \, d\omega \\
&= D_{KL}(q_\theta(\omega) \,||\, p(\omega)) - \int_\Omega q_\theta(\omega) \log \frac{p(\mathbf{X}, \mathbf{Y} \,|\, \omega)}{p(\mathbf{X}, \mathbf{Y})} \, d\omega \\
&= D_{KL}(q_\theta(\omega) \,||\, p(\omega)) - \int_\Omega q_\theta(\omega) \log p(\mathbf{Y} \,|\, \mathbf{X}, \omega) \, d\omega \\
&\quad + \log p(\mathbf{Y} \,|\, \mathbf{X}) \underbrace{\int_\Omega q_\theta(\omega) \, d\omega}_{1} \\
&\propto D_{KL}(q_\theta(\omega) \,||\, p(\omega)) - \mathbb{E}_{q_\theta(\omega)} \left[\log p(\mathbf{Y} \,|\, \mathbf{X}, \omega) \right] := \mathcal{L}_{VI}(\theta) \,, \text{[1]} \quad (3.5)
\end{aligned}
$$

with $\mathcal{D} = (\mathbf{X}, \mathbf{Y})$, $\mathbf{X} \in \mathcal{P}(\mathcal{X})$ the set of instances in the dataset and $\mathbf{Y} \in \mathcal{P}(\mathcal{Y})$ the set of labels in the dataset. The objective $\mathcal{L}_{VI}(\theta)$ is equivalent to the evidence lower-bound $\mathcal{L}_{ELBO}(\theta) := -D_{KL}(q_\theta(\omega) \,||\, p(\omega)) + \mathbb{E}_{q_\theta(\omega)} \left[\log p(\mathbf{Y} \,|\, \mathbf{X}, \omega) \right] \leq \log p(\mathbf{Y} \,|\, \mathbf{X})$, up to a factor of -1. Minimizing $\mathcal{L}_{VI}(\theta)$ is therefore equivalent to maximizing the ELBO, which bounds the log-evidence of the data.

$\mathcal{L}_{VI}(\theta)$ is decomposed in two terms. The left term, $D_{KL}(q_\theta(\omega) \,||\, p(\omega))$, is a regularization term that prevents the variational posterior to stray too far away from the prior distribution. On the other hand, the right term, $-\mathbb{E}_{q_\theta(\omega)} \left[\log p(\mathbf{Y} \,|\, \mathbf{X}, \omega) \right]$, is the expected log-likelihood of the predictive posterior distribution over the variational posterior. This term will be minimal if the variational posterior can explain the data distribution well

[1] We use $\propto$ here in the sense that the two quantities are equal up to a constant: $x \propto y \Leftrightarrow \exists C \in \mathbb{R}, x = y + C$.

by making the right predictions. Therefore, this term has a similar role than the loss functions such as the L_1 or L_2 loss functions.

In order to minimize $\mathcal{L}_{VI}(\theta)$, we need to compute its gradient $\nabla_\theta \mathcal{L}_{VI}(\theta)$. However, similarly to the gradient of the ELBO in the case of VAEs (see Section 2.3.1), the Reinforce estimator

$$\nabla_\theta \mathbb{E}_{q_\theta(\omega)} \left[\log p(\mathbf{Y} \mid \mathbf{X}, \omega)\right] = \mathbb{E}_{q_\theta(\omega)} \left[\log p(\mathbf{Y} \mid \mathbf{X}, \omega) \nabla_\theta \log q_\theta(\omega)\right] . \tag{3.6}$$

has a variance that is too high to be used reliably. Several techniques to solve this issue will be described in Section 3.3, most of them make use of the reparameterization trick already seen with VAEs (see Section 2.3.1).

Given this variational posterior, the posterior predictive distribution can be estimated by MC sampling on the variational distribution

$$p(y \mid x, \mathcal{D}) \approx \int_\Omega p(y \mid x, \omega)\, q_\theta(\omega)\, d\omega \approx \frac{1}{K_{MC}} \sum_{k=1}^{K_{MC}} p(y \mid x, \hat{\omega}_k) , \tag{3.7}$$

where $(\hat{\omega}_k)_{k=1}^{K_{MC}} \sim q_\theta(\omega)$ refer K_{MC} independent samples from the variational posterior.

3.3. Variational approximations and deep ensemble

Several propositions for an approximate variational distribution $q_\theta(\omega)$ have been made in the literature. These techniques aim to solve the issue of the Reinforce estimator in Equation 3.6 which has a variance that is too high.

3.3.1. MC Dropout

One of the most important techniques to estimate the gradient of the ELBO is *MC Dropout*, proposed by Gal and Ghahramani (2016). In Section 2.1.2, we presented Dropout as a regularization technique, which adds binary noise to the outputs of each layer. In fact, Dropout can be considered as a variational posterior $q_\theta(\omega)$ that applies a Bernoulli distribution to the parameters of the network

$$\begin{aligned} W_i &= M_i \cdot \operatorname{diag}\left([r_j^i]_{j=1}^{n_i - 1}\right) \\ r_j^i &\sim \mathcal{B}(1 - p_i) \quad \forall i = 1, \ldots, L, \forall j = 1, \ldots, n_{i-1} , \end{aligned} \tag{3.8}$$

with $\omega = \{W_i\}_{i=1}^L$ being the random variable on the weights of the network, and $\theta = \{M_i\}_{i=1}^L$ are the variational parameters of the distribution $q_\theta(\omega)$. $p_i \in [0, 1]$ is the dropout rate of the i^{th} layer (see Section 2.1.2).

To solve the issue of the high variance estimator in Equation 3.6, it is necessary to introduce the reparameterization trick, just like it was needed for VAEs. We introduce a deterministic transformation $\omega = g(\theta, \epsilon)$ with $\epsilon \sim p(\epsilon)$. Using a change of variable by

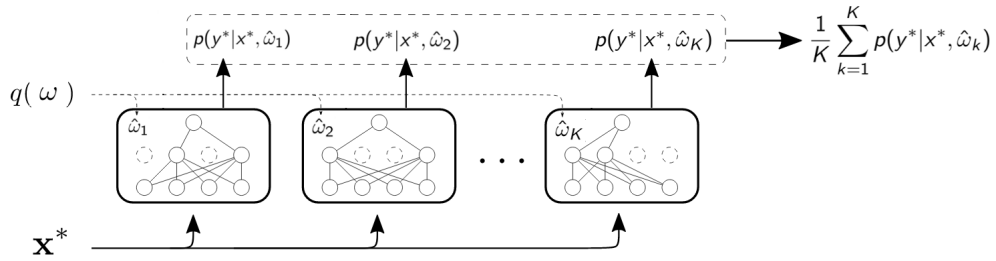


Figure 3.3.: MC Dropout sampling technique. Credits: Xiao and Wang (2019).

knowing that $q_\theta(\omega)\, d\omega = p(\epsilon)\, d\epsilon$, the reparameterization trick leads to a new expression for $\mathcal{L}_{VI}(\theta)$

$$\begin{aligned}\mathcal{L}_{VI}(\theta) &= -\mathbb{E}_{q_\theta(\omega)}\left[\log p(\mathbf{Y} \mid \mathbf{X}, \omega)\right] + D_{KL}(q_\theta(\omega) \,||\, p(\omega)) \\ &= -\int \log p(\mathbf{Y} \mid \mathbf{X}, \omega)\, q_\theta(\omega)\, d\omega + D_{KL}(q_\theta(\omega) \,||\, p(\omega)) \\ &= -\int \log p(\mathbf{Y} \mid \mathbf{X}, g(\theta, \epsilon))\, p(\epsilon)\, d\epsilon + D_{KL}(q_\theta(\omega) \,||\, p(\omega))\,. \end{aligned} \tag{3.9}$$

We can further decompose $(\mathbf{X}, \mathbf{Y})$ in mini batches $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^M$ of size M and approximate $\mathcal{L}_{VI}(\theta)$ as the estimator

$$\hat{\mathcal{L}}_{VI}(\theta) = -\frac{N}{M}\sum_{i=1}^{M} \int \log p(\mathbf{y}_i \mid \mathbf{x}_i, g(\theta, \epsilon))\, p(\epsilon)\, d\epsilon + D_{KL}(q_\theta(\omega) \,||\, p(\omega))\,. \tag{3.10}$$

In MC Dropout, the reparameterization trick is defined as

$$\omega = \{W_1, \ldots, W_l\} = \left\{M_1 \cdot \text{diag}\left(\left[r_j^1\right]_{j=1}^{n_1}\right), \ldots, M_L \cdot \text{diag}\left(\left[r_j^L\right]_{j=1}^{n_L}\right)\right\} =: g(\theta, \mathbf{r})\,. \tag{3.11}$$

With this configuration, the KL term in $\mathcal{L}_{MC}(\theta)$ can further be approximated to give the MC Dropout estimator

$$\hat{\mathcal{L}}_{MCD}(\theta) := -\frac{N}{M}\sum_{i=1}^{M} \log p(\mathbf{y}_i \mid \mathbf{x}_i, g(\theta, \hat{\mathbf{r}}_i)) + \sum_{i=1}^{L} \lambda_i \,||M_i||^2\,, \tag{3.12}$$

which is the usual formulation of a loss function with a L_2 regularization term, with $(\hat{\mathbf{r}}_i)_{i=1}^L$ being L samples from the Bernoulli distribution (see Equation 3.8). This expression of the loss is differentiable, which solves the previous issue of the non-deterministic gradient.

The approximation of the KL term comes from the KL condition (Gal, 2016). If we choose the prior $p(\omega)$ to be a fully factorized Gaussian, i.e. $p(\omega) = \prod_{i=1}^L p(W_i) = \prod_{i=1}^L \mathcal{N}(0, I/l_i^2)$, with all W_i independent and l_i the *prior length scale*

$$l_i^2 := \frac{2N\lambda_i}{1 - p_i} \tag{3.13}$$

then we have

$$\nabla_\theta D_{KL}(q_\theta(\omega) \,|\, p(\omega)) \approx \nabla_\theta N \sum_{i=1}^{L} \lambda_i ||M_i||^2 \tag{3.14}$$

which means that both quantities follow the same optimization path.

MC Dropout is a simple way to make a neural network Bayesian, as most models already use dropout in some way, and it is easy to sample (see Fig. 3.3). MC Dropout has also been successfully applied to CNNs (Gal and Ghahramani, 2015).

3.3.2. Variational Gaussian Dropout

Another example of a BNN variational posterior is *Variational Gaussian Dropout* (Kingma et al., 2015), or more simply *Variational Dropout*. Their idea is to use the local reparameterization trick. Suppose a hidden layer with input neurons $\mathbf{A}$ from a previous layer, weight matrix $\mathbf{W}$, and $\mathbf{B} = \mathbf{A}\mathbf{W}^T$ is the output matrix of the layer before a non-linearity is applied. Suppose that the variational posterior is defined as a fully factorized Gaussian distribution, i.e. $q_\theta(\omega_{j,i}) = \mathcal{N}(\mu_{i,j}, \sigma_{i,j}^2)\, \forall \omega_{i,j} \in \mathbf{W}$ with the variables $\omega_{i,j}$ being independent. A sample for each weight would then be needed to compute the loss, which is computationally inefficient. Instead, the idea would be to sample from the activation matrix coefficients $b_{m,j} \in \mathbf{B}$, since the weights $\omega_{i,j}$ only influence the log-likelihood through $\mathbf{B}$. Since the activation matrix is lower dimensional, this is more practical. The variational distribution on $\mathbf{B}$ is defined as

$$\begin{aligned} \forall \omega_{i,j} \in \mathbf{W},\, q_\theta(\omega_{i,j}) = \mathcal{N}(\mu_{i,j}, \sigma_{i,j}^2) \Rightarrow q_\theta(b_{m,j} \,|\, \mathbf{A}) = \mathcal{N}(\gamma_{m,j}, \delta_{m,j}) \\ \gamma_{m,j} = \sum_i a_{m,i}\, \mu_{i,j} \quad \text{and} \quad \delta_{m,j} = \sum_i a_{m,i}^2\, \sigma_{i,j}^2 \,. \end{aligned} \tag{3.15}$$

This allows for the reparameterization $b_{m,j} = \gamma_{m,j} + \sqrt{\delta_{m,j}}\, \epsilon_{m,j}$, with $\epsilon_{m,j} \sim \mathcal{N}(0,1)$, as opposed to $\omega_{i,j} = \mu_{i,j} + \sigma_{i,j}\, \epsilon_{i,j}$ with $\epsilon_{i,j} \sim \mathcal{N}(0,1)$ in the case of the usual reparameterization trick.

In the case of the Variational Gaussian Dropout, noise variables ξ are drawn from the Gaussian distribution $\mathcal{N}(1, \alpha)$ with $\alpha = \frac{p}{1-p}$, $p \in [0, 1[$ being the dropout rate. Indeed, Srivastava et al. (2014) have shown that a noise variable following a continuous distribution with the same mean and variance as the Bernoulli distribution works just as well to prevent the parameters of the network from overfitting. As such, this distribution is used to perturb the input layer: $\mathbf{B} = (\mathbf{A} \odot \xi)\, \mathbf{M}$, with $\xi_{i,j} \sim \mathcal{N}(1, \alpha)$, and $\mathbf{M} = (m_{i,j})_{i,j}$ a weight matrix. In this case, the distribution over the activations $b_{m,j}$ is

$$q_\phi(b_{m,j} \,|\, A) = \mathcal{N}(\gamma_{m,j}, \delta_{m,j}), \text{ with } \gamma_{m,j} = \sum_i a_{m,i}\, m_{i,j}, \text{ and } \delta_{m,j} = \alpha \sum_i a_{m,i}^2\, m_{i,j}^2 \,. \tag{3.16}$$

The Gaussian dropout noise can also be interpreted as a fully factorized Gaussian variational posterior on the weights, such that $q_\theta(\omega_{i,j}) = \mathcal{N}(m_{i,j}, \alpha\, m_{i,j}^2)$.

While this approach allows for a faster computation with a lower variance, it has some limitations, as the optimization diverges if $\alpha > 0.5$. Besides, Hron et al. (2018) have shown that Variational dropout is improper, meaning that there is no normalizing constant such that $p(\omega)$ represents a valid probability density. This means that Variational Dropout cannot rigorously be considered as Bayesian.

3.3.3. Deep ensembles

Adjacent to the Bayesian approaches, *deep ensemble* (Lakshminarayanan et al., 2017) are a class of deep learning models that use ensemble learning to approximate uncertainties. The idea is to train multiple neural networks initialized at multiple different weights $(\omega_0^1, \ldots, \omega_0^N)$. This technique is easy to implement with no hyperparameter tuning, and can be parallelized, but it does require to train multiple models. Ensemble approaches have been shown to give better uncertainty estimates than variational approaches (Lakshminarayanan et al., 2017).

Fort et al. (2019) studied the behavior of deep ensemble compared to variational BNNs, like MC Dropout. They show that deep ensemble models will converge to different modes of the parameter space Ω while sharing the same loss performance. BNNs, on the other hand, will stay around a single mode, which can be explained by the behavior of Gaussian distributions. So, while deep ensemble capture the variety of the different modes in the parameter space, BNN are limited to learn a narrow range distribution centered on a single mode. The ideal case would be to have a type of model which would both capture the different modes, and the variety in each mode. This might for instance be possible by combining an ensemble of variational Bayesian models.

3.4. Estimating the uncertainties

Let us suppose that we have a trained BNN with a posterior distribution $\omega \sim p(\omega \mid \mathcal{D})$. Equation 3.7 gives us the posterior predictive distribution, which gives us the prediction of the model given the input X and the training dataset $\mathcal{D}$. But how can we get quantitative values of the aleatoric and epistemic uncertainties described in Section 3.1 ?

Consider the case of discrete outputs $Y \in \mathcal{Y}$ (such as a classification task). The aleatoric and epistemic uncertainty can be decomposed by considering the total predictive uncertainty, represented by the entropy of the predictive posterior distribution (Gal et al., 2017b; Depeweg et al., 2018)

$$\mathcal{U}_{pred.}(X) := \mathbb{H}\left[p(Y \mid X, \mathcal{D})\right] = -\sum_{y \in \mathcal{Y}} p(y \mid X) \log p(y \mid X)\,. \tag{3.17}$$

By definition, the entropy measures the uncertainty of a categorical distribution. It is maximum when the distribution is uniform, in which case it is equal to $\log C$ where C is the number of classes; and is equal to 0 (and minimal) when the distribution outputs only one class (deterministic distribution).

Then, considering that the distribution $p(Y \mid \omega, X)$, taken at a weight point estimate $\omega \in \Omega$, does not exhibit epistemic uncertainty, the expected conditional entropy of Y given ω

$$\mathcal{U}_{aleat.}(X) := \mathbb{E}_{p(\omega \mid \mathcal{D})}\mathbb{H}\left[p(Y \mid \omega, X)\right] = -\int_{\Omega} p(\omega \mid \mathcal{D}) \left(\sum_{y \in \mathcal{Y}} p(y \mid X, \omega) \log p(y \mid X, \omega) \right) d\omega \tag{3.18}$$

quantifies the aleatoric uncertainty. By marginalizing the entropy over $p(\omega \mid \mathcal{D})$, the epistemic uncertainty on the model's parameters is effectively removed. In other words, it is the residual uncertainty observed on average when ω is known.

The epistemic uncertainty is then simply obtained by removing the aleatoric uncertainty from the total predictive uncertainty

$$\begin{aligned} \mathcal{U}_{epist.}(X) &:= \mathcal{U}_{pred.}(X) - \mathcal{U}_{aleat.}(X) \\ &= \mathbb{H}\left[p(Y \mid X, \mathcal{D})\right] - \mathbb{E}_{p(\omega \mid \mathcal{D})}\mathbb{H}\left[p(Y \mid \omega, X)\right] \\ &= \mathbb{I}(Y, \omega \mid X, \mathcal{D})\,. \end{aligned} \tag{3.19}$$

The last equality comes from the relationship $\mathbb{I}(Y, \omega \mid X, \mathcal{D}) = \mathbb{H}[Y \mid X] - \mathbb{H}[Y \mid \omega, X, \mathcal{D}]$.

The epistemic uncertainty is quantified by the mutual information between the label y and the weights ω. In other words, the epistemic uncertainty will be higher if a new sample X with label Y carries a lot of information about the model parameters ω. This makes sense, as we expect outlier data to exhibit high epistemic uncertainty, and such an outlier would give a lot of information about ω if it were learned, more than a data point near a mode of the data distribution which is well known by the model.

Given the following property of the mutual information , it is possible to derive another expression for the epistemic uncertainty

$$\begin{aligned} \mathcal{U}_{epist.}(X) = \mathbb{I}(Y, \omega \mid X, \mathcal{D}) &= D_{KL}(p(Y, \omega \mid \mathcal{D}, X) \,||\, p(\omega, \mathcal{D})\, p(Y \mid X, \mathcal{D})) \\ &= \int p(Y \mid \omega, X, \mathcal{D})\, p(\omega \mid \mathcal{D}) \log \frac{p(Y \mid \omega, X, \mathcal{D})\, p(\omega \mid \mathcal{D})}{p(Y \mid X, \mathcal{D})\, p(\omega \mid \mathcal{D})} \\ &= \mathbb{E}_{p(\omega \mid \mathcal{D})}[D_{KL}(p(Y \mid X, \omega, \mathcal{D}) \,||\, p(Y \mid X, \mathcal{D}))]\,, \end{aligned} \tag{3.20}$$

which can be interpreted as the average consensus between each of the point-wise predictive posterior distributions $p(Y \mid X, \omega, \mathcal{D})$ and the averaged posterior predictive distribution $p(Y \mid X, \mathcal{D})$: if the point-wise predictive posteriors are in disagreement with each other and thus with $p(Y \mid X, \mathcal{D})$, then the epistemic uncertainty will be high.

Of course, equations 3.17, 3.18 and 3.19 are all intractable as we do not have access to the posterior distribution $p(\omega \mid \mathcal{D})$. Instead, they are all approximated by the variational posterior $q_\theta(\omega)$ and then estimated by MC sampling.

$$\hat{\mathcal{U}}_{pred.}(X) := -\sum_{y\in\mathcal{Y}} \left(\frac{1}{M}\sum_{i=1}^{M} p(y\,|\,\hat{\omega}_i, X)\right) \log\left(\frac{1}{M}\sum_{i=1}^{M} p(y\,|\,\hat{\omega}_i, X)\right) \tag{3.21}$$

$$\hat{\mathcal{U}}_{aleat.}(X) := -\frac{1}{M}\sum_{i=1}^{M}\sum_{y\in\mathcal{Y}} p(y\,|\,\hat{\omega}_i, X)\log p(y\,|\,\hat{\omega}_i, X) \tag{3.22}$$

$$\hat{\mathcal{U}}_{epist.}(X) := \hat{\mathcal{U}}_{pred}(X) - \hat{\mathcal{U}}_{aleat}(X)\,, \tag{3.23}$$

with $(\hat{\omega}_i)_{i=1}^{M}$ being M samples from the variational posterior $q_\theta(\omega)$. In the case of a deep ensemble, they would be the different weights of each model in the ensemble.

Gal et al. (2017b) used MC estimation of the uncertainties in a BNN with MC Dropout as a variational posterior, while Shaker and Hüllermeier (2020), in the context of a random forest, used those estimations for an ensemble model.

3.5. A Bayesian Neural Network based on Dropout Regulation

3.5.1. Adjusting the dropout rate in MC Dropout

We present here the contribution done in Theobald et al. (2020). We have seen that the variational approaches of BNNs rely on hyperparameters which can influence the quality of the variational posterior. In MC Dropout for instance, the dropout rate p is fixed at a certain value. While it can be tuned by grid search, it is still usually set at a typical value of $p = 0.5$, which is not necessarily an optimal value to maximize the ELBO. Besides, grid search can be computationally expensive as the dropout rate parameters scale with the model architecture.

Still, there were attempts in the literature at considering the dropout rate as an actual parameter to be optimized, meaning that the dropout rate is considered as a variational parameter: $p \in \theta$. This generally means computing the gradient with respect to p: $\nabla_p \mathcal{L}_{VI}(\theta)$, and this is a difficult problem. We will now present a few approaches seen in the literature to tackle this gradient computation.

Concrete Dropout. *Concrete Dropout* (Gal et al., 2017a) set up the following variational parameters: $\theta = \{M_i, p_i\}_{i=1}^{L}$, with M_i and p_i the weights and dropout rates of the i^{th} layer respectively. Then, the KL term in Equation 3.5 can be expressed as

$$D_{KL}(q_\theta(\omega)\,||\,p(\omega)) = \sum_{i=1}^{L} D_{KL}(q_{M_i}(W_i)\,||\,p(W_i))\,. \tag{3.24}$$

(Gal et al., 2017a) chose more precisely a quantised Gaussian prior, as proposed by Gal et al. (2017b). Instead of defining the prior as a fully factorized Gaussian $\mathcal{N}(0, l^{-2}I)$, it is rather defined using a discrete probability mass function $p_l(\mathbf{w}) \propto$

$e^{-\frac{l^2}{2}\mathbf{w}^T\mathbf{w}}$ over a finite space $\mathbf{w} \in \mathcal{W}$. The variational distribution is expressed as $q(\mathbf{w}) = p\,\delta(\mathbf{w}-0) + (1-p)\,\delta(\mathbf{w}-M)$, where $\mathbf{w}$ is a row of the weight matrix W. Then, the KL divergence to the prior is

$$\begin{aligned} D_{KL}(q(\mathbf{w})\,||\,p_l(\mathbf{w})) &= \sum_{\mathbf{w}\in\mathcal{W}} q(\mathbf{w}) \log \frac{q(\mathbf{w})}{p_l(\mathbf{w})} \\ &= \sum_{\mathbf{w}\in\mathcal{W}} q(\mathbf{w}) \log \frac{q(\mathbf{w})}{e^{-\frac{l^2}{2}\mathbf{w}^T\mathbf{w}}} + \log Z_l \\ &= p\log p + (1-p)\log\frac{1-p}{e^{-\frac{l^2}{2}\mathbf{w}^T\mathbf{w}}} + \log Z_l \\ &+ \sum_{\mathbf{w}\in\mathcal{W}\setminus\{0,M\}} 0 \log \frac{0}{e^{-\frac{l^2}{2}\mathbf{w}^T\mathbf{w}}} \\ &\propto \frac{l^2(1-p)}{2}\mathbf{w}^T\mathbf{w} - \mathbb{H}(p) \end{aligned} \tag{3.25}$$

where $Z_l = \sum_{\mathbf{w}\in\mathcal{W}} e^{-\frac{l^2}{2}\mathbf{w}^T\mathbf{w}}$ is the normalizing constant of $p_l(\mathbf{w})$ and $\propto$ is again used in the sense that the two quantities are equal up to a constant. As a reminder, the entropy of a Bernoulli distribution with parameter p is $\mathbb{H}(p) = -p\log p - (1-p)\log(1-p)$.

By summing this expression over the rows $\mathbf{w}$ of the weight matrix W, we get

$$D_{KL}(q_M(W)\,||\,p(W)) \propto \frac{l^2(1-p)}{2}||M||^2 - K\mathbb{H}(p) \tag{3.26}$$

for every weight matrix W in each layer of the neural network, and K being the dimension of the input.

The entropy term will push p towards the maximum entropy, i.e. $p = 0.5$, even more if the model has a lot of parameters. But as the amount of data N increases, the regularization term becomes more and more insignificant (see Equation 3.12), and the dropout rate will go towards 0.

The problem now is that the Bernoulli distribution is non-differentiable with respect to its parameter p. To solve the computation of $\nabla_p \mathcal{L}_{VI}(\theta)$, they introduce the *Concrete distribution* (Maddison et al., 2017), which is a continuous relaxation of the Bernoulli distribution. This relaxation allows to define a new random variable z such that it is possible to write $z = g(p, U)$ where p is a deterministic parameter and U a random variable. The concrete distribution allows to sample values in the interval $[0, 1]$, however most of its mass is concentrated near 0 and 1, which makes it a convenient continuous alternative to the Bernoulli distribution.

In the one-dimensional case we use here, the concrete distribution is parameterized by a temperature $t > 0$, and is simply expressed as

$$z = \sigma\left(\frac{1}{t}\left(\log p - \log(1-p) + \log U - \log(1-U)\right)\right) \tag{3.27}$$

with $U \sim \mathcal{U}(0,1)$ a random variable following the uniform distribution on the interval $[0,1]$, and σ representing the sigmoid function. The new variable z will be used to sample the coefficients of the Dropout mask. Since the relation between z and u is independent of p, this expression is differentiable with respect to p, allowing the optimization of the dropout rate using the reparameterization trick.

Learnable Bernoulli Dropout. Another attempt at optimizing the dropout rate is *Learnable Bernoulli Dropout*, or LBD (Boluki et al., 2020). Instead of considering a continuous relaxation of the Bernoulli distribution, they instead estimate the gradient $\nabla_p \mathcal{L}_{VI}(\theta)$ using the Augment-REINFORCE-Merge (ARM) estimator (Yin and Zhou, 2019).

Let us define $\mathbf{z} = [z_1, \dots, z_K]$ K random variables following K i.i.d. Bernoulli distributions $(\mathcal{B}(1-p_i))_{i=1}^K$, with $\alpha = [\alpha_1, \dots, \alpha_K]$ the logits of the Bernoulli distributions such that $\forall i \in [1,K]$, $p_i = 1 - \sigma(\alpha_i)$ with σ the sigmoid function. $\mathbf{z}$ represents here the dropout mask over the neural network.

We define the objective function as

$$\hat{\mathcal{L}}(\theta) := -\frac{N}{M} \sum_{i=1}^{M} \log p(y_i \,|\, x_i, \hat{\omega}_i) + D_{KL}(q_\theta(\omega) \,||\, p(\omega)) \tag{3.28}$$

with the variational posterior $q_\theta(\omega)$ defined as a Bernoulli distribution (Equation 3.8).

The aim is to optimize the expected value of the loss with respect to the Bernoulli parameters

$$\min_{\theta,\alpha} \mathbb{E}_{\mathbf{z}}[\mathcal{L}(\theta)] \,. \tag{3.29}$$

Then, the ARM estimator of the gradient of $\mathbb{E}_{\mathbf{z}}[\mathcal{L}(\theta)]$ with respect to α is given by

$$\begin{aligned} \nabla_\alpha \mathbb{E}_{\mathbf{z}}[\mathcal{L}(\theta)] = \frac{N}{M} \sum_{i=1}^{M} \mathbb{E}_{\mathbf{u}_i} \left[\left(\mathcal{E}(\theta, \mathbb{1}_{[u_i > \sigma(-\alpha)]}) - \mathcal{E}(\theta, \mathbb{1}_{[u_i < \sigma(\alpha)]}) \right) \left(u_i - \frac{1}{2} \right) \right] \\ + \nabla_\alpha \mathcal{R}(\alpha) \end{aligned} \tag{3.30}$$

with $\mathbf{u}_i \sim \prod_{k=1}^{L} \mathcal{U}_{[0,1]}(u_{i,k})$ a fully factorized uniform distribution on [0,1], $\mathcal{E}$ the empirical loss for a minibatch, and $\mathcal{R}(\alpha)$ representing the regularization term of the loss. $\mathcal{E}(\theta, \mathbb{1}_{[u_i < \sigma(\alpha)]})$ represents the loss that is obtained by setting the dropout mask as $\mathbf{z} = \mathbb{1}_{[\mathbf{u} < \sigma(\alpha)]} := (\mathbb{1}_{[\mathbf{u}_1 < \sigma(\alpha_1)]}, \dots, \mathbb{1}_{[\mathbf{u}_K < \sigma(\alpha_K))]})$, and the same for $\mathcal{L}(\theta, \mathbb{1}_{[u_i > \sigma(-\alpha)]})$.

This estimator of the gradient has been shown to be unbiased with low variance (Yin and Zhou, 2019), which makes it appropriate for the optimization process. The proof behind 3.30 is fully described by Yin and Zhou (2019), but can be understood by first computing the gradient in the univariate case (in which case the gradient of the sigmoid function is known), and then generalizing the univariate case to the multivariate case by using the law of total expectation.

3.5.2. The Dropout Regulation algorithm

Both previous methods offer a solution to the Bernoulli gradient problem, but they are still relatively complex to use. We propose an alternative algorithm which offers a simple way to automatically adjust the dropout rate during training.

This method, called *Dropout Regulation*, is inspired by automation and control loops. The goal is to regulate the dropout rate given the gap of performance between the training data and the validation data, as dropout was originally conceived as a regularization technique to prevent overfitting. This means that this method is different from the optimization based approaches, with the advantage to be very simple to implement and use. The idea is that when the model has similar performances between the training and validation data, dropout is not needed, but when the model starts performing much better on the training data, we need a higher dropout rate for a stronger regularization.

This idea is also linked to epistemic uncertainty, the intuition being that a higher dropout rate would yield a more faithful estimation of the epistemic uncertainty in the case where the model does overfit. Overfitting is more prevalent when the amount of data is low with respect to the information needed to learn the underlying data distribution. Remind from Equation 3.26 that a higher dropout rate is preferable as the amount of data is limited. On the other hand, if the amount of data is large enough to efficiently learn the data distribution, overfitting will be naturally mitigated and a lower dropout rate would be acceptable.

Consider now a learning task with training data $\mathcal{D}^{train}$. Recall from Section 2.1.2 that the empirical risk is the estimator used to optimize the parameters of the neural network

$$\hat{\mathcal{R}}^{train}(\omega) := \frac{1}{N} \sum_{(x,y)\in\mathcal{D}^{train}} \mathcal{L}(f_\omega(x), y)\,. \tag{3.31}$$

Usually, a second dataset $\mathcal{D}^{val}$, called the validation dataset is used to evaluate the generalization capabilities of the model, but this data is not used during training, i.e. $\mathcal{D}^{val} \cap \mathcal{D}^{train} = \emptyset$. Therefore, a second validation empirical risk $\hat{\mathcal{R}}^{val}(\omega)$ can be defined by evaluating the risk on $\mathcal{D}^{val}$.

Generally, the validation empirical risk is greater than the training empirical risk, i.e. $\hat{\mathcal{R}}^{train}(\omega) \leq \hat{\mathcal{R}}^{val}(\omega)$. Ideally, a model that generalizes well would have $\hat{\mathcal{R}}^{train}(\omega) \approx \hat{\mathcal{R}}^{val}(\omega)$. In reality, what we observe is that $\hat{\mathcal{R}}^{train}(\omega)$ decreases to an optimum during training and $\hat{\mathcal{R}}^{val}(\omega)$ first decreases and then increases as the model overfits and loses the ability to generalize over a larger dataset. Therefore, we would want the quantity $\epsilon(\omega) = \hat{\mathcal{R}}^{val}(\omega) - \hat{\mathcal{R}}^{train}(\omega)$ to be as close to zero as possible during training, or at the very least prevent it from increasing. This gap $\epsilon(\omega)$ between the two empirical risks can be considered as a measure of overfitting. Fig. 2.4 illustrates the typical behavior of $\hat{\mathcal{R}}^{val}(\omega)$, $\hat{\mathcal{R}}^{train}(\omega)$ and $\epsilon(\omega)$, and the optimal scenario we would want to have (dotted line).

Dropout helps to mitigate the effects of overfitting by adding random binary noise to the parameters of the network. But having a fixed value of the dropout rate p may not be ideal: it could prevent the model to learn more efficiently if the gap $\epsilon(\omega)$ is low

enough (in which case we can afford a lower dropout rate or remove it entirely). On the other hand, it may not be enough to prevent overfitting if $\epsilon(\omega)$ is already increasing: we would want a higher dropout rate to have a stronger regularization.

Inspired by automation techniques, we subject the dropout rate p to a control loop: the dropout rate will be adjusted dynamically during training with respect to $\epsilon(\omega)$. We assume that the dropout rate is the same in all layers. Algorithm 2 describes the dropout regulation algorithm.

Algorithm 2 Dropout Regulation training

Input: Deep learning model $f_\omega : \mathcal{X} \to \mathcal{Y}$, initial parameters ω_0, training dataset $\mathcal{D}^{train}$, validation dataset $\mathcal{D}^{val}$, dropout controller function $C(\epsilon) : \mathbb{R} \to [0, 1]$, number of epochs T, optimization algorithm Fit.
Output: optimal parameters ω_{t^*}, p_{t^*}
Set $p_0 = 0$
for $t = 1$ *to* T **do**
  $\omega_t = Fit(\omega_t | f_{\omega_{t-1}}, \mathcal{D}^{train})$ with dropout rate p_{t-1}
  Compute $\epsilon(\omega_t) = \hat{\mathcal{R}}^{val}(\omega_t) - \hat{\mathcal{R}}^{train}(\omega_t)$
  Set $p_t = C(\epsilon(w_t))$
end
Return optimal parameters ω_{t^*}, p_{t^*} s.t. $t^* = \text{argmin}_t \hat{\mathcal{R}}^{val}(\omega_t)$

Fig. 3.4 illustrates how the dropout regulation algorithm is reminiscent of a control loop. The target value $\epsilon_0 = 0$ is on the left. The quantity that we want to control and possibly set to 0 is $\delta\epsilon = \epsilon(\omega) - \epsilon_0 = \epsilon(\omega)$ which is represented by the sum-operator on the left of the figure, the symbol of a cross inside a circle with the signs of the inputs outside. Error $\epsilon(\omega)$ then goes through the dropout controller to compute the next dropout rate $p = p(\epsilon(\omega))$ (Dropout Controller block). After that, the model learns through a new epoch with the new dropout rate to compute the updated parameters ω (Learn/Fit Block). At this point, the algorithm can stop at any fixed point to output ω (arrow on the far right). The parameters ω are then used to evaluate the empirical risks of both the training and validation datasets (Evaluate train/val blocks below). These risks are then subtracted to get $\epsilon(\omega)$ (sum-operator below). We then loop back by computing $\delta\epsilon$. Note that in Algorithm 2 we select ω such that the empirical risk on the validation data is minimal.

3.5.3. PI Dropout controller

The representation of the algorithm as a control loop allows us to consider the system as a form of controller, with the hope of adjusting p to set $\epsilon(\omega)$ to the desired value of 0. All that needs to be done now is how to specify $p(\epsilon)$. We will use for this application the PID controller.

The PID (Proportional-Integral-Derivative) controller is defined as follows. Let $y(t)$ be a measured process variable, $r(t)$ a desired setpoint for $y(t)$, $\epsilon(t) = y(t) - r(t)$ the error measured between the process variable and the target. Let $u(t)$ be a control variable that

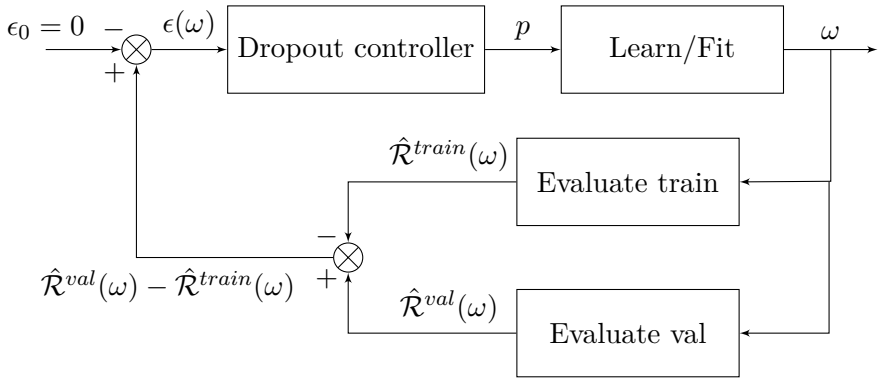


Figure 3.4.: Control loop representation of Algorithm 2

can be adjusted by the controller over time to minimize the error. A PID controller sets $u(t)$ to be

$$u(t) := K_p \left(\epsilon(t) + \frac{1}{T_i} \int_0^t \epsilon(\tau) d\tau + T_d \frac{d\epsilon(t)}{dt} \right) , \tag{3.32}$$

with $K_p, T_d \geq 0, T_i > 0$.

In Dropout Regulation, we compute $u(t)$ using Equation 3.32 by using the error $\epsilon(t)$ between the two risks. However $u(t) \in \mathbb{R}$ and the control variable that the network can use is the dropout rate p which is in $[0, 1]$. Remind that dropout is a form of regulation. When the error is low, or even negative, it means that the network needs to learn from its training data better and therefore dropout is not necessary. If the error is too high it means that the model is overfitting and dropout is needed. As $\epsilon \to +\infty$, $C(\epsilon) \to 1$. Therefore $C(\epsilon(\omega))$ is defined as follows

$$C(\epsilon(\omega)) := (1 - \exp(-u(t))) \mathbb{1}_{\mathbb{R}^+}(u(t)) , \tag{3.33}$$

where $\mathbb{1}_{\mathbb{R}^+}(x) = 1$ if $x \geq 0$, 0 otherwise. Usually a temperature parameter T would be considered, with $C(\epsilon(\omega)) = (1 - \exp(-u(t)/T)) \mathbb{1}_{\mathbb{R}^+}(u(t))$, however the coefficient K_p in $u(t)$ already serves that purpose.

We have now defined every parameter of our controller. We will only consider a PI controller, i.e. $T_d = 0$. Also considering that time is discretized, the integral action is approximated by a sum. Ultimately, our controller is defined as

$$u(t) = K_p \left(\epsilon(\omega_t) + \frac{1}{T_i} \sum_{k=1}^{t} \epsilon(\omega_k) \right) , \tag{3.34}$$

and $p(\epsilon(\omega_t))$ as defined in Equation 3.33. The entire regulation process is described in Algorithm 3.

3.5.4. Experiments

Setup. The PI dropout controller has been implemented in the context of an image classification problem. The dataset used was the Dogs vs. Cats Kaggle dataset (Kaggle,

Algorithm 3 PI dropout controller training

Input: Deep learning model $f_\omega : \mathcal{X} \to \mathcal{Y}$, initial parameters ω_0, training dataset $\mathcal{D}^{train}$, validation dataset $\mathcal{D}^{val}$, K_p, T_i, number of epochs T, optimization algorithm Fit
Output: optimal parameters w_{t^*}, p_{t^*}
Set $p_0 = 0$
for $t = 1$ *to* T **do**
  $\omega_t = Fit(\omega_{t-1}|f_\omega, \mathcal{D}^{train})$ with dropout rate p_{t-1}
  Compute $\epsilon(\omega_t) = \hat{\mathcal{R}}^{val}(\omega_t) - \hat{\mathcal{R}}^{train}(\omega_t)$
  Compute $u(t) = K_p \left(\epsilon(\omega_t) + \frac{1}{T_i}\sum_{k=1}^{t} \epsilon(\omega_k)\right)$
  Set $p_{t+1} = (1 - \exp(-u(t)))\mathbb{1}_{\mathbb{R}^+}(u(t))$
end
Return optimal parameters ω_{t^*}, p_{t^*} s.t. $t^* = \text{argmin}_t\, \hat{\mathcal{R}}^{val}(\omega_t)$

2013). We are using 8000 images for training, 1600 for validation and 8000 for testing. The architecture is based on VGG-16 (Simonyan and Zisserman, 2015) with all weights preloaded and frozen but the last convolution block. Then, the output of the convolution part is flattened, and two dense layers of 256 neurons with dropout are added before the last output layer.

The loss used is the categorical crossentropy, using a SGD optimizer with learning rate 10^{-4}, momentum 0.9 and a batch size of 100, and 60 epochs. We are comparing 4 methods: Regulation Dropout with $K_p = 10$ and $T_i = 10000$, Concrete Dropout, MC Dropout and a deterministic network. For both MC Dropout and the deterministic network, the dropout rate is fixed at $p = 0.5$ as it is the commonly used value. And more specifically, dropout is not used at testing time for the deterministic model and therefore only the aleatoric uncertainty is used for uncertainty measurements. At testing time, we compare the models which had the best validation loss during training.

In order to assess the quality of the uncertainty measured, we are using the PAvPU metric, which combines the probability that a prediction that is certain is accurate $P(accurate|certain)$, and the probability that an inaccurate prediction is uncertain $P(uncertain|inaccurate)$ (Mukhoti and Gal, 2018). More precisely, the idea is that if the considered uncertainty is well calibrated, and if the stochastic nature of the dataset is faint (i.e., low aleatoric uncertainty), then any false prediction should be uncertain, and any correct prediction should be certain (and vice versa). Indeed, a model which outputs wrong predictions with high confidence would be unreliable. The PAvPU metric is therefore calculated as such

$$PAvPU := \frac{n_{iu} + n_{ac}}{n_{iu} + n_{ac} + n_{ic} + n_{au}}, \tag{3.35}$$

where n_{iu} is the number of inaccurate and uncertain samples, n_{ac} the number of accurate and certain samples, n_{ic} the number of inaccurate and certain samples and n_{au} the number of accurate and uncertain samples. These are computed using a validation dataset. In this expression, we compute the proportion of samples that are both inaccurate and uncertain, and both accurate and certain, over all samples. If PAvPU is lower, it means

that more samples are inaccurate and certain, or accurate and uncertain, both cases that are not preferable. As such, higher values of PAvPU are considered to represent uncertainty estimates that are more calibrated with regards to the predictions of the network.

The PAvPU metric is evaluated at different thresholds of uncertainty, ranging from $\mathcal{U}_{min}$ to $\mathcal{U}_{max}$ in a linear progression, where $\mathcal{U}_{min}$ and $\mathcal{U}_{max}$ are respectively the minimum and maximum uncertainty over the test dataset:

$$\mathcal{U}_{thres}(t) := \mathcal{U}_{min} + t(\mathcal{U}_{max} - \mathcal{U}_{min}), t \in [0, 1]\,. \tag{3.36}$$

Results. Table 3.1 shows the accuracy and PAvPU metric for the different methods tested. The PAvPU metric was evaluated at an uncertainty threshold equal to the mean uncertainty over the test dataset. The accuracy of Regulation is slightly below of Concrete but above Deterministic and MC Dropout.

Method	Accuracy	PAvPU
Deterministic	0.932	0.7155
MC Dropout	0.928	0.70734
Concrete	**0.935**	0.7575
Regulation	0.934	**0.7579**

Table 3.1.: Accuracy and PAvPU metrics for the different methods. Bold indicates the best metric obtained by the method used.

We can see that for all methods, the accuracy is very high and so the number of accurate samples is very high when compared to the inaccurate. That means that a higher uncertainty will decrease the number of accurate and certain samples. So, interestingly, the PAvPU value for the deterministic network is greater than that of MC Dropout. This is because MC Dropout represents more uncertainty overall (because of the added epistemic uncertainty). Then some accurate samples are classified as uncertain although they were not in the deterministic method. Because the proportion of inaccurate samples is so low, the added uncertainty is not compensated by the number of inaccurate and uncertain samples. Both Concrete and Regulation show similar levels of PAvPU, so our method is quite competitive regarding this metric and accuracy.

Fig. 3.5 shows the results of the PAvPU metric curve for varying thresholds of uncertainty, ranging from $t = 0$ to $t = 1$. We can observe that Regulation is quite similar to Concrete in terms of PAvPU metric performance, and that they are both above Deterministic and MC Dropout. The PAvPU curve for MC Dropout is below the curve of Deterministic, which relates to the previous explanation.

Influence of K_p. We will now study the influence of the hyperparameter K_p of the controller. This parameter influences the strength of the adjustment of the dropout rate p. The higher K_p is, the higher p will become for a given error $\epsilon(\omega)$. We are going to compare two different values of K_p. First of all, we are keeping the same parameters

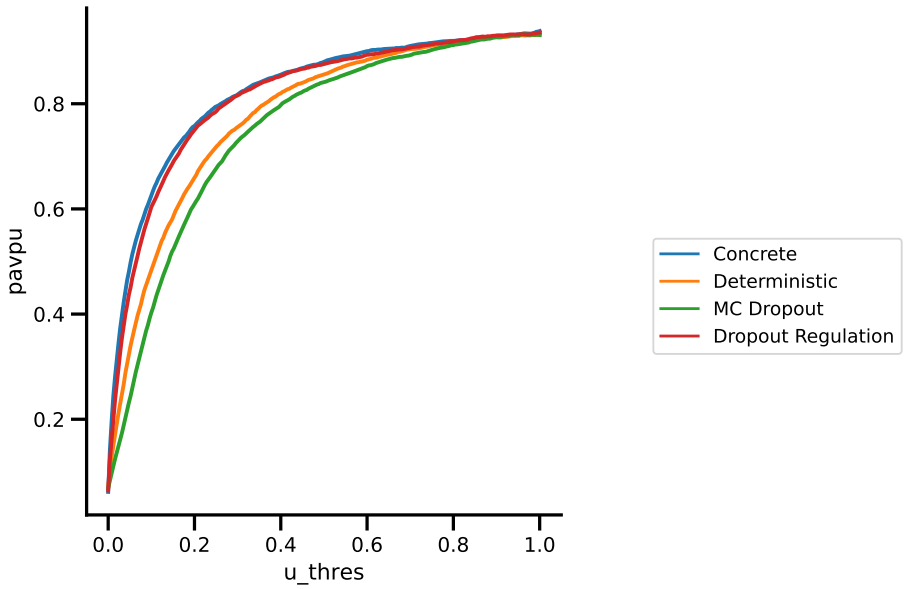
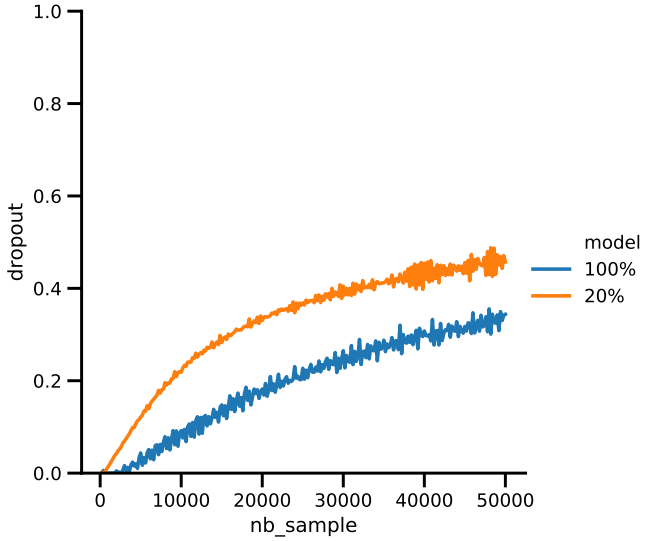


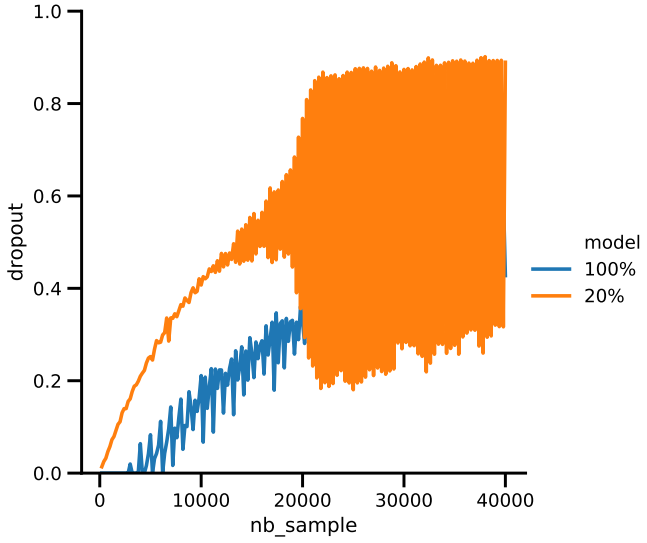
Figure 3.5.: PAvPU metric for 250 different thresholds on uncertainty and different models (higher is better).

as the previous experiment, but this time we have $N_{train} = N_{test} = 1000$ training and test images and $N_{val} = 200$ validation images and a SGD optimizer with learning rate 10^{-4} with no momentum. The two values of K_p that we will compare are $K_p = 0.15$ and $K_p = 0.50$. These values of K_p are lower when compared to the previous experiment because of the absence of momentum. Indeed this parametrization allows us to spotlight an interesting oscillating behavior of the controller. We compare two models: one model trained with only 20% of the data and another trained with 100% of the data.

As one can see, an interesting behavior happens when we increase K_p. In both cases, the dropout starts by slowly increasing, reflecting the increase of $\epsilon(\omega)$ as both performances on the validation and trains datasets diverge. We can also see that the dropout rate is found to be lower for the largest dataset, which is reasonable because the amount of data is in itself a form of regularization. In Fig. 3.6a, the dropout rate is more or less stable throughout. However, in Fig. 3.6b, we observe a divergence of the dropout rate as soon as the value of the dropout rate reaches approximately 0.5, for the model with 20% of the data. Indeed, when the dropout rate starts to reach higher values, the network has less and less active neurons to learn complex patterns which may be necessary to learn from the data well. Therefore, the performance on the train and validation datasets both start to plummet, to the point where the model basically starts to guess randomly. Because of that, the error $\epsilon(\omega)$ indirectly reduces since the performances on both datasets are almost equally worse. Which means that the controller will select next a dropout rate which is way lower, in general $p < 0.5$. Since most of the weights were "frozen" when the dropout rate was high, when the dropout rate becomes lower again, the model performs well again and $\epsilon(\omega)$ is high again, which means that on the next epoch the dropout rate



(a)



(b)

Figure 3.6.: Dropout rate for $K_p = 0.15$ (a) and $K_p = 0.50$ (b). The x-axis represents the amount of images seen during training. It is equivalent to the number of epochs multiplied by the number of images per epoch. In blue the dropout rate for the model with 20% of the data, in orange 100% of the data.

will be very high, etc.

It is actually very interesting to see that such a behavior happens when we apply a controller on a neural network. In most physical systems where a controller can be used, such oscillations can be encountered when the parameters of the controller are too strong. Here also, one must choose the hyperparameters of the controller well to prevent such behavior from happening. Other hyperparameters like the optimizer (momentum) or the size of dataset can also be influential. In any case, since we select in the end the model which has the best performance on the validation data overall, the influence of the oscillations - if they happen - on the final performance are mitigated.

3.5.5. Conclusion

We presented here a method for adjusting the dropout rate during training, called Dropout Regulation. By using control loops from automation, the goal is to empirically adjust the dropout rate in order to have a precise estimation of the uncertainty while being simple to implement. We then applied this method to an image classification problem. Our method got comparable results when compared to other methods with respect to accuracy and uncertainty metrics. Even if our method requires the tuning of the hyperparameters of the controller which can induce some side effects like an oscillating system, Dropout Regulation offers an alternative to the state of the art which makes it interesting.

In the following chapter, we will deepen the notions introduced in this chapter in the context of Bayesian Deep Learning for classification problems, and see how those can be extended to regression problems, with a specific target application in astrophysics.

Chapter 4.

Bayesian deep regression: application to the estimation of galaxy ellipticity

In this chapter, our focus will be on the estimation of galaxy ellipticities, in accordance to the ANR project AstroDeep. Being able to not only have an accurate estimation of those ellipticities, but also reliable uncertainty estimates is of paramount importance, as errors due to uncertainty can propagate through the whole research pipeline. The main contribution in this chapter is the introduction of a Bayesian Neural Network adapted for a multivariate regression problem, as the ellipticity is defined in the complex plane. Then, we will also introduce "Network coaching" as an algorithm that can help the model progressively learn from more and more complex datasets, as the galaxy images can be simulated.

4.1. Introduction to weak lensing

4.1.1. Gravitational weak lensing and cosmic shear estimation

One of the goals of large galaxy surveys such as the *Legacy Survey of Space and Time* (LSST Science Collaboration, 2009) conducted at the Vera C. Rubin Observatory is to study *dark energy*. This component of unknown nature was introduced in the current cosmological standard model to explain the acceleration of the Universe expansion. One way to probe dark energy is to study the mass distribution across the Universe. This distribution mostly follows the dark matter distribution, which does not interact with baryonic matter (i.e. visible matter) except through gravitation, as dark matter represents around 85% of the matter in the Universe.

Consequently, cosmologists need to use indirect measurement techniques such as *cosmic shear*, which measures the coherent distortion of background galaxies images by foreground matter due to *weak gravitational lensing* (Kilbinger, 2015).

To explain this notion, recall that general relativity tells us that gravitation is not a force, but rather an effect of the curvature of space-time. This curvature can be, for instance, due to the presence of a mass, as illustrated in Fig. 4.1.

In astrophysics, gravitational lensing is the distortion of the image of an observed source, induced by the bending of space-time, thus of the light path, generated by the presence of mass along the line of sight. The mass acts like a lens, in partial analogy with optical lenses, as illustrated in Fig. 4.2a. In the *strong* gravitational lensing regime,

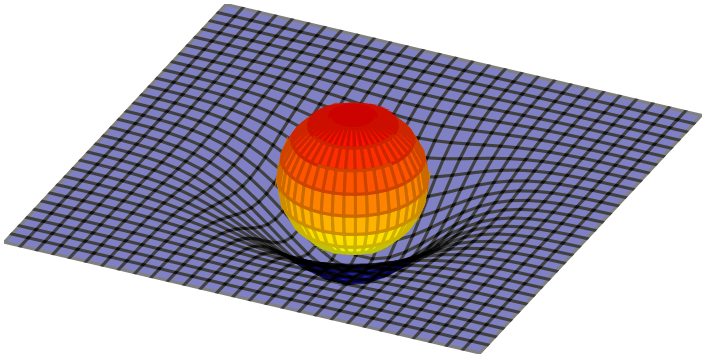
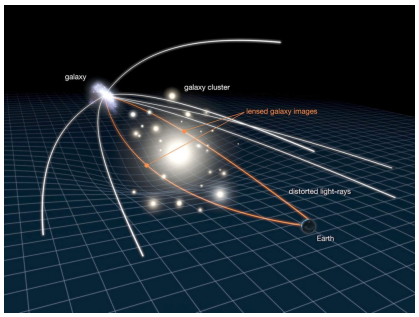


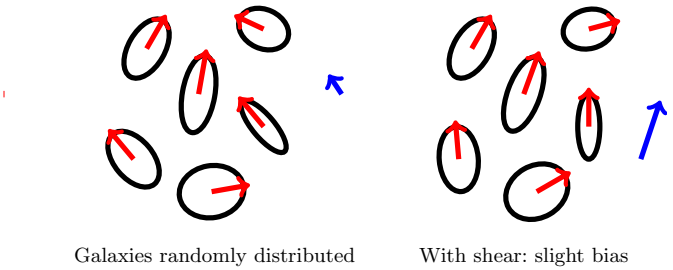
Figure 4.1.: The curvature of space-time is influenced by the presence of an object with a mass. An effect of this curvature is gravity, which affects how matter (and light) travel through space-time.

the mass is large enough for the gravitational lens to be easily detected, as its effects are very noticeable. However, that is not always the case.

The *weak* gravitational lensing effect, on the other hand, is so faint (1% of galaxy shape measurement) that it cannot be detected on an isolated galaxy. Thus, only statistical tools provide a way to detect a local correlation in the observed galaxies orientations. This correlation yields a local value at every point of the observable Universe, defining the *cosmic shear field*. As pictured in Fig. 4.2b, in an isotropic and uniform Universe orientations of galaxies are expected to follow a uniform distribution (left panel). The statistical average of their oriented elongations, hereafter called *complex ellipticities*, is expected to be null. In presence of a lens, a smooth spatial deformation field modifies coherently the complex ellipticities of neighboring galaxies so that their mean is no longer zero (right panel).



(a) Gravitational lensing.



(b) Cosmic shear.

Figure 4.2.: (a) Effect of gravitational lensing: the mass bends the light and deforms the images of the galaxies. (b) Weak lensing: the correlation between orientations and shapes of neighbour galaxies defines the cosmic shear. In blue: average ellipticity. Left: the expected ellipticity distribution. Right: the observed ellipticity distribution. Image: (a) NASA/ESA.

The unbiased measurement of cosmic shear is a major ambition of nowadays cosmology (Mandelbaum, 2018). One avenue to estimate the cosmic shear locally is to combine individual galaxy ellipticity measurements. By looking deeper into the sky, that is to older objects, the next generation of telescopes will allow for the detection of a very large number of galaxies, potentially leading to very precise shear measurement and resulting in tight constraints on dark energy parameters.

Methods already exist to estimate galaxy ellipticities through direct measurement on images recorded by telescope cameras (Kaiser et al. (1995) for example). This is a complex problem as, among other things, the shear signal is carried by faint galaxies which makes it very sensitive to background noise. Another central issue for current and coming surveys in galaxy shape determination, is the treatment of statistically dominant overlapping objects, an effect called *blending*. A current survey projects that 58% of the detected objects will appear blended (Bosch et al., 2017) and this value is expected to reach around 62% for LSST (Sanchez et al., 2021). To overcome this issue, solutions exist such as deblending (Bertin, E. and Arnouts, S., 1996; Melchior et al., 2018a; Arcelin et al., 2020): the separation of overlapping objects. Yet, they are not perfect and rely on an accurate detection of blended scenes which is also a complex problem. As such, in addition to a precise estimation of the complex ellipticities, a reliable measurement of the uncertainties is crucial in order to discard, or at least decrease the impact of, unreliable and inaccurate measurements avoiding as much as possible the introduction of a bias into the shear estimation.

Classical ellipticity measurement methods usually adopt assumptions about the shape of the galaxies (for example via the shape of the window function in Kaiser et al. (1995)) potentially resulting in model (or inductive) bias. In contrast, deep learning models like CNNs make it possible to learn and recognize complex and diverse galaxy shapes directly from data without making any other hypothesis than the representativeness of the training sample. They consequently are appropriate tools to learn the regression of galaxy ellipticities, even in the presence of noise and complex distortions.

Yet standard CNNs are unable to estimate the epistemic uncertainty, as we have explained it in Chapter 3. This type of uncertainty is essential to detect outliers from the training samples, or formulated accordingly to our problem, to distinguish between reliable or unreliable galaxy ellipticity estimation. Therefore, it is needed to build a model that predicts the ellipticities with a calibrated aleatoric uncertainty, and such that it can also minimize the impact of wrongly estimated ellipticity values due to outliers in the computation of the shear by estimating epistemic uncertainty. In the following section, we will show how Bayesian Neural Networks can be used to build and learn such a model.

4.1.2. Galaxy ellipticity and sources of uncertainty

In this section, we will define the galaxy ellipticity, and define which types of noise can corrupt the dataset.

As mentioned previously, it is possible to estimate cosmic shear combining individual measurements of galaxy shape. This shape information is quantified by the *complex*

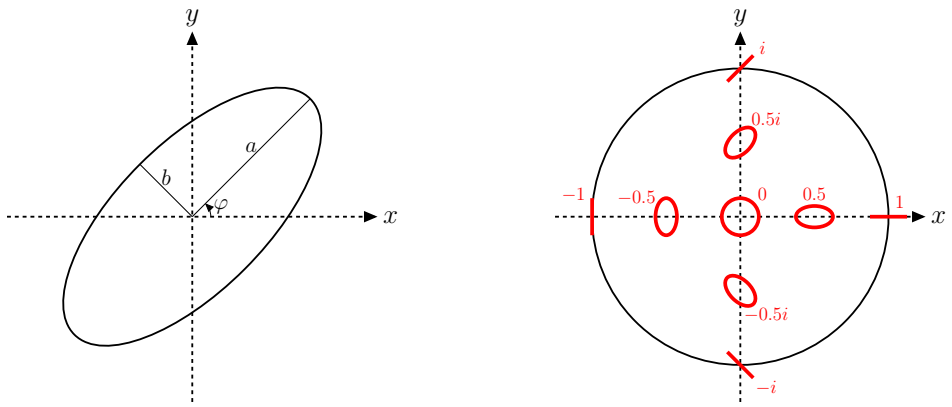
ellipticity, which is defined in cosmology as in Def. 1.

Definition 1. *Let E be an ellipse with major axis a, minor axis b, and with φ as its position angle. The* complex ellipticity[1] *of E is defined as:*

$$\varepsilon = \varepsilon_1 + \varepsilon_2\, i = \frac{1-q^2}{1+q^2}\, e^{2i\varphi}, \tag{4.1}$$

where $q = \frac{b}{a}$ is the axis ratio of the ellipse.

An illustration of the ellipticity parameters is shown in Fig. 4.3a. The complex ellipticity defines a bijection between the orientation and the elongation of the ellipse on one side, and the unit disk on the other side, see Fig. 4.3b.



(a) Ellipticity parameters: major axis a, minor axis b, position angle φ.

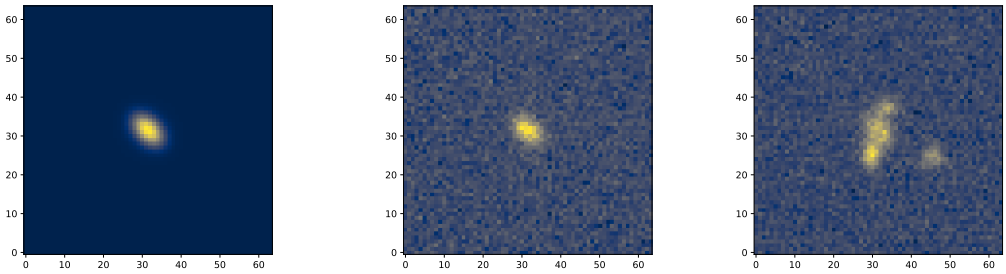
(b) Bijective mapping between ellipse shapes and complex ellipticities.

Figure 4.3.: Geometric representation of the complex ellipticity. (a) The ellipse parameters. (b) The complex ellipticity defines a bijection between ellipse shapes and the unit disk. An ellipticity with low magnitude is close to a circle, while one with a high magnitude is closer to a straight line. The argument defines the orientation of the ellipse.

However, the process to achieve an unbiased measurement of cosmic shear, starting with the estimation of ellipticities, is going to be challenging for several reasons. We test the reliability of our networks prediction on noise and blending, two of the many possible bias sources in the cosmic shear estimation. Both of these issues result from the fact that the shear signal is mostly carried by faint galaxies. By definition, these objects have a low signal-to-noise ratio. The noise corrupts the galaxy images, making the shape estimation much harder (see Fig. 4.4b), and can introduce a bias in shear measurement (Kilbinger, 2015).

Also, a large part of these faint objects will appear blended with foreground galaxies. Even in scenes where objects are only slightly overlapped, the apparent shape of the detected object does not correspond to a single galaxy model and an ellipticity measurement on this image could give a completely wrong result.

[1] This is different from the real valued ellipticity defined as $\varepsilon = 1 - \frac{b}{a}$.



(a) Isolated noiseless galaxy (b) Isolated noisy galaxy (c) Blended noisy galaxies

Figure 4.4.: Three different types of image complexity for the same galaxy: isolated without noise, isolated with noise, blended with noise. Notice how the noise slightly deforms the galaxy (b) and how the blended galaxies makes the ellipticity estimation very difficult (c) when compared to a simple isolated galaxy without noise (a).

Again, this work is the first step of a longer-term goal. Here, we target a reliable estimation of galaxy ellipticity posterior from single band images. This includes obtaining a well calibrated aleatoric uncertainty, tested here with and without the addition of Poisson noise on images, and an epistemic uncertainty allowing for minimization of the impact of untrustworthy measurement due to outliers (here, blended scenes).

We simulate LSST-like images, allowing us to control the parameters of the scene, e.g., the number of galaxies, their location on the image, and the level and type of noise applied. We consider four categories of simulated data: isolated centered galaxies without noise, isolated centered galaxies with noise, and blended scene with and without noise. Images are 64x64 pixels stamps simulated in the brightest of the six bands corresponding to the LSST filters, each of them selecting a different part of the electromagnetic spectrum. These images are simulated placing, in their center, a galaxy whose ellipticity is to be measured.

The image generating process relies on the GalSim library (Rowe et al., 2014) and is based on a catalog of parametric models fitted to real galaxies for the third Gravitational Lensing Accuracy Testing (GREAT3) Challenge (Mandelbaum et al., 2014). It consists in 1) producing an image of a centered noiseless isolated galaxy from a model sampled randomly from the catalog, with its corresponding physical properties (size, shape, orientation, PSF, brightness, redshift, etc) 2) measuring the complex ellipticity of the galaxy with the KSB algorithm (Kaiser et al., 1995) on the image and record it as the image label, 3) possibly adding on random image location other galaxy images (from 0 to 5) to generate blended scenes 4) possibly adding Poisson noise (as in Arcelin et al. (2020)).

In this study and for sake of interpretability, we only provide the reference band (the brightest) which we use to define the target ellipticity, making our images two-dimensional. Once again, while using multiple bands is useful for blended galaxies (Melchior et al., 2018b; Arcelin et al., 2020), here we focus only on predicting the ellipticity of a single centered galaxy with a correctly estimated uncertainty.

4.2. Galaxy ellpiticity regression with a Bayesian CNN

In this section, we will describe the Bayesian Neural Network architecture and BDL framework used to estimate the galaxy ellipticities, as well as the uncertainty estimates for both possible sources: noise and blended scenes.

In Chapter 3, we described the general framework of BDL in the case of a classification task. However, the label space here is not discretized, as the model has to predict two real numbers: the real and imaginary part of the complex ellipticity ε. More precisely, we must be able to predict any complex number in the unit disk, i.e. $\mathcal{Y} = \{\varepsilon \in \mathbb{C} \,|\, |\varepsilon| \leq 1\}$. The model must therefore perform a regression in a 2D space. As such, the estimators of epistemic and aleatoric uncertainties must be redefined.

4.2.1. Uncertainty estimation in a multivariate regression task

Uncertainty estimators. The goal is to reliably estimate the two layers of complexity in the galaxy images: the noise and the blended scenes. Noise is a source of aleatoric uncertainty, since it corrupts the images themselves. Training a CNN to solve a standard regression problem with an L_2 loss does not allow us to estimate the aleatoric uncertainty: the output is not a discrete probability distribution so that its entropy defined by Equation 3.18 is not non-negative anymore. Moreover its computation might become tricky.

In the same vein, epistemic uncertainty would be higher on blended scenes, as the model is trained on images of isolated galaxies. However, in this case the mutual information $\mathbb{I}(\varepsilon, \omega \,|\, X)$ becomes difficult to compute.

In order to define the aleatoric uncertainty, we need to model the complex ellipticity as a random variable following a Multivariate Normal distribution (MVN): $p(\varepsilon \,|\, X, \omega) = \mathcal{N}(\mu_\omega(X), \Sigma_\omega(X))$, with $f_\omega(X) = (\mu_\omega(X), \Sigma_\omega(X))$. The support of this distribution is $\mathbb{R}^2$ as we consider the complex ellipticity as a couple $(\varepsilon_1, \varepsilon_2) \in \mathbb{R}^2$, composed of both the real part and imaginary part of ε. This MVN allows us to model the impact of data-related noise on the prediction of the complex ellipticity. f_ω, the neural network, will output the parameters of the MVN, and will be trained by maximizing the log-likelihood of the MVN, or equivalently minimizing the empirical risk associated to the following loss function

$$\mathcal{L}(\varepsilon, f_\omega(X)) := \det \Sigma_\omega(X) + (\varepsilon - \mu_\omega(X))^T \Sigma_\omega(X)^{-1} (\varepsilon - \mu_\omega(X)) \,. \tag{4.2}$$

Note that the covariance matrix $\Sigma_\omega(X)$, which represents the predictive noise, depends on X. This is because the noise is *heteroscedastic*, in opposition to a *homoscedastic* noise Σ_ω, which would be constant w.r.t. X (Kendall and Gal, 2017).

In fact, this loss function is equivalent to maximizing the log-likelihood of the MVN $\mathcal{N}(\mu_\omega(X), \Sigma_\omega(X))$

$$\begin{aligned} \log p(\varepsilon | X, \omega) &= -2\pi - \frac{1}{2} \det \Sigma_\omega(X) - \frac{1}{2} (\varepsilon - \mu_\omega(X))^T \Sigma_\omega(X)^{-1} (\varepsilon - \mu_\omega(X)) \\ &\propto - \det \Sigma_\omega(X) - (\varepsilon - \mu_\omega(X))^T \Sigma_\omega(X)^{-1} (\varepsilon - \mu_\omega(X)) \end{aligned} \tag{4.3}$$

where $\propto$ is used here as both equal up to a constant (which is -2π) and proportional to a constant (which is $\frac{1}{2}$).

Both Kendall and Gal (2017) and Depeweg et al. (2018) proposed a decomposition of the predictive uncertainty in the case of a regression problem with heteroscedastic aleatoric uncertainty. Given a posterior probability distribution $p(\varepsilon \,|\, X, \mathcal{D})$, the predictive uncertainty is defined as the total predictive variance

$$\Sigma_{pred.}(X, \mathcal{D}) := \mathrm{Cov}(\varepsilon \,|\, X, \mathcal{D}) \,. \tag{4.4}$$

Given the law of total variance, we have that

$$\mathrm{Cov}(\varepsilon \,|\, X, \mathcal{D}) = \mathrm{Cov}_{p(\omega \,|\, \mathcal{D})} \left[\mathbb{E}(\varepsilon \,|\, X, \omega)\right] + \mathbb{E}_{p(\omega \,|\, \mathcal{D})} \left[\mathrm{Cov}(\varepsilon \,|\, X, \omega)\right] \,. \tag{4.5}$$

Where $\mathbb{E}(\varepsilon \,|\, X, \omega)$ and $\mathrm{Cov}(\varepsilon \,|\, X, \omega)$ represent, respectively, the mean and the variance of ε according to $p(\varepsilon \,|\, \omega, X)$, i.e. the posterior predictive distribution. In our case, $\mathbb{E}(\varepsilon \,|\, X, \omega) = \mu_\omega(X)$ and $\mathrm{Cov}(\varepsilon \,|\, X, \omega) = \Sigma_\omega(X)$, by definition.

In Equation 4.5, the left term $\mathbb{E}_{p(\omega \,|\, \mathcal{D})} \left[\mathrm{Cov}(\varepsilon \,|\, X, \omega)\right]$ is the average value of $\mathrm{Cov}(\varepsilon \,|\, X, \omega)$ on the posterior $p(\omega \,|\, \mathcal{D})$. Therefore, this term only estimates the uncertainty of ellipticity epsilon given ω, not the uncertainty on ω itself. As such, it quantifies aleatoric uncertainty

$$\Sigma_{aleat.}(X, \mathcal{D}) := \mathbb{E}_{p(\omega \,|\, \mathcal{D})} \left[\mathrm{Cov}(\varepsilon \,|\, X, \omega)\right] \,. \tag{4.6}$$

On the other hand, the right term $\mathrm{Cov}_{p(\omega \,|\, \mathcal{D})} \left[\mathbb{E}(\varepsilon \,|\, X, \omega)\right]$ quantifies the covariance of the mean of the posterior predictive distribution on the posterior $p(\omega \,|\, \mathcal{D})$. The contribution to the covariance here depends only on ω, and not ε. Therefore, this term represents the epistemic uncertainty

$$\Sigma_{epist.}(X, \mathcal{D}) := \mathrm{Cov}_{p(\omega \,|\, \mathcal{D})} \left[\mathbb{E}(\varepsilon \,|\, X, \omega)\right] \,. \tag{4.7}$$

It follows from these definitions that

$$\Sigma_{pred.}(X, \mathcal{D}) = \Sigma_{aleat.}(X, \mathcal{D}) + \Sigma_{epist.}(X, \mathcal{D}) \,. \tag{4.8}$$

Note these measures are 2×2 matrices, i.e. they are multidimensional measures of uncertainty. However, it would be useful to have a scalar measure of uncertainty to compare them. In a multivariate setup, the covariance matrices are part of $\mathbb{R}^{n \times n}$ with $n > 1$, which is not a totally ordered set. Noticing the fact that the differential entropy of a MVN $\mathcal{N}(\mu, \Sigma)$ is $\frac{1}{2} \ln (2\pi e \det \Sigma)$, it follows that the determinant of Σ is a valid scalar estimation of the uncertainty. Therefore, we define the following scalar quantities

$$\mathcal{U}_{aleat.}(X, \mathcal{D}) := \det \Sigma_{aleat.}(X, \mathcal{D}) \,, \tag{4.9}$$

$$\mathcal{U}_{epist.}(X, \mathcal{D}) := \det \Sigma_{epist.}(X, \mathcal{D}) \,, \tag{4.10}$$

$$\mathcal{U}_{pred.}(X, \mathcal{D}) := \det \Sigma_{pred.}(X, \mathcal{D}) \,. \tag{4.11}$$

Given a BNN with a variational posterior $q_\theta(\omega)$, we can approximate the uncertainties by using MC sampling, which gives the following uncertainty estimators

$$\hat{\Sigma}_{aleat.}(X) := \frac{1}{K}\sum_{k=1}^{K} \Sigma_{\hat{\omega}_k}(X)\,, \tag{4.12}$$

$$\hat{\Sigma}_{epist.}(X) := \frac{1}{K}\sum_{k=1}^{K} \left(\mu_{\hat{\omega}_k}(X) - \hat{\mu}(X)\right)\left(\mu_{\hat{\omega}_k}(X) - \hat{\mu}(X)\right)^T\,, \tag{4.13}$$

$$\hat{\Sigma}_{pred.}(X) := \hat{\Sigma}_{aleat.}(X) + \hat{\Sigma}_{epist.}(X)\,, \tag{4.14}$$

with $\hat{\mu}(X) := \frac{1}{K}\sum_{k=1}^{K} \mu_{\hat{\omega}_k}(X)$ and $(\hat{\omega}_k)_{k=1}^{K} \sim q_\theta(\omega)$.
And finally the scalar values of those estimators

$$\hat{\mathcal{U}}_{aleat.}(X) := \det \hat{\Sigma}_{aleat.}(X)\,, \tag{4.15}$$

$$\hat{\mathcal{U}}_{epist.}(X) := \det \hat{\Sigma}_{epist.}(X)\,, \tag{4.16}$$

$$\hat{\mathcal{U}}_{pred.}(X) := \det \hat{\Sigma}_{pred.}(X)\,. \tag{4.17}$$

In our case, the variational posterior distribution $q_\theta(\omega)$ will be defined as the Dropout variational posterior, see Section 3.3.1.

Confidence ellipses. Another quantitative result we can derive from these uncertainty covariance matrices are their *confidence regions* (or in the case of a MVN, a confidence ellipsoid).

Let $X \sim \mathcal{N}(\mu, \Sigma)$ be a MVN in $\mathbb{R}^n$. Let us define the random variable

$$D := (X-\mu)^T \Sigma^{-1} (X-\mu)\,. \tag{4.18}$$

This new variable represents the square Mahalonobis distance of the random vector X to the mean of the MVN μ. The farther X is from μ, the higher D is and the lower the probability density $p(X)$.

Given a confidence level $1-\alpha$, the $(1-\alpha)$-confidence ellipsoid is then defined relative to D, which follows a chi-square distribution with n degrees of freedom: $D \sim \chi^n_2$. Therefore

$$\mathbb{P}(D \leq \chi^n_2(1-\alpha)) = 1-\alpha = \mathbb{P}(X \in \mathcal{E}^n(\alpha))\,, \tag{4.19}$$

where $\chi^n_2(1-\alpha)$ is the $1-\alpha$ quantile of the distribution χ^n_2, and

$$\mathcal{E}^n(\alpha) := \{X \in \mathbb{R}^n \,|\, D \leq \chi^n_2(1-\alpha)\} \tag{4.20}$$

defines the confidence ellipsoid with confidence level $1-\alpha$.

The length of the ellipsoid semi-axes depend on α[2] and the eigenvalues of Σ, $(\lambda_i)_{i=1}^n$.

Indeed, since Σ is a positive-definite matrix, we can write $\Sigma = P\Lambda P^T$ with $P \in \mathcal{M}_n(\mathbb{R})$ orthogonal[3] and $\Lambda = diag(\lambda_i)$[4]. Then,

[2] More precisely $\chi^n_2(1-\alpha)$.
[3] $P \in \mathcal{M}_n(\mathbb{R})$ is orthogonal iff $P^{-1} = P^T$.
[4] So $\Lambda^{-1} = diag(\lambda_i^{-1})$ since all of the eigenvalues of Λ are strictly positive.

$$D = (X-\mu)^T P^T \Lambda^{-1} P (X-\mu)\,. \tag{4.21}$$

If we define $\Delta := P(X-\mu) =: (\delta_i)_{i=1}^n$, we place ourselves in the orthogonal basis centered on μ such that the axes of this basis coincide with the axes of the ellipsoid. Plugging this into Equation 4.20 gives

$$\begin{aligned} \Delta^T \Lambda^{-1} \Delta &\leq \chi_2^n(1-\alpha)\,, \\ \sum_{i=1}^n \frac{\delta_i^2}{\lambda_i} &\leq \chi_2^n(1-\alpha)\,, \end{aligned} \tag{4.22}$$

which is the equation defining an ellipsoid with semi-axes length $a_i = \sqrt{\lambda_i\, \chi_2^n(1-\alpha)}$, $\forall i \in [\![1, n]\!]$, and centered on μ.

Let us replace ourselves in 2D. The $(1-\alpha)$-confidence ellipse $\mathcal{E}^2(\alpha)$ is centered on $\mu = (\mu_1, \mu_2)^T$ with major semi-axis $a = \sqrt{-2\ln(\alpha)\,\lambda_1}$ and minor semi-axis $b = \sqrt{-2\ln(\alpha)\,\lambda_2}$, given $\lambda_1 \geq \lambda_2 > 0$.

Indeed, given the definition of $\chi_2^2(1-\alpha)$, and that the CDF of χ_2^2 is $F(x) = \int_0^{x/2} e^{-t} dt$, we have

$$\begin{aligned} \int_0^{\chi_2^2(1-\alpha)/2} \exp(-t)dt = [-\exp(-t)]_0^{\chi_2^2(1-\alpha)/2} = 1 - \exp\left(-\frac{\chi_2^2(1-\alpha)}{2}\right) &:= 1-\alpha \\ \iff \chi_2^2(1-\alpha) &= -2\ln\alpha\,. \end{aligned} \tag{4.23}$$

Finally, if we note $\Sigma = \begin{pmatrix} \sigma_{1,1} & \sigma_{1,2} \\ \sigma_{1,2} & \sigma_{2,2} \end{pmatrix}$, then the eigenvalues can be expressed as

$$\lambda_1 = \frac{\sigma_{1,1}+\sigma_{2,2}}{2} + \sqrt{\left(\frac{\sigma_{1,1}-\sigma_{2,2}}{2}\right)^2 + \sigma_{1,2}^2}\,, \tag{4.24}$$

$$\lambda_2 = \frac{\sigma_{1,1}+\sigma_{2,2}}{2} - \sqrt{\left(\frac{\sigma_{1,1}-\sigma_{2,2}}{2}\right)^2 + \sigma_{1,2}^2}\,, \tag{4.25}$$

and the position angle φ is

$$\varphi = \begin{cases} 0 & \text{if } \sigma_{1,2} = 0 \text{ and } \sigma_{1,1} \geq \sigma_{2,2} \\ \frac{\pi}{2} & \text{if } \sigma_{1,2} = 0 \text{ and } \sigma_{1,1} < \sigma_{2,2} \\ \operatorname{atan2}(\lambda_1 - \sigma_{1,1}, \sigma_{1,2}) & \text{else}\,. \end{cases} \tag{4.26}$$

In summary, to compute the confidence ellipsoid given a risk level α, a variance-covariance matrix Σ and a mean vector μ:

- We compute the eigenvalues λ_1, λ_2 and the position angle φ of Σ given Equations 4.24, 4.25 and 4.26.
- We compute the length of the semi-axes a and b given those eigenvalues and risk level α.
- We plot the ellipsoid centered on μ, with semi-axes a and b, and position angle φ.

4.2.2. Architecture of the Bayesian CNN and training protocol

Architecture and data augmentation. Given that we are working with images, the natural choice of a deep learning architecture is a CNN. The architecture of our network was inspired by the work of Dieleman, who proposed a simple model specifically tuned for the Galaxy Zoo challenge, therefore adapted to our data (Dieleman et al., 2015). Each image is augmented in four different parts by cropping thumbnails from high resolution images, centered on spatial modes of light profile (see Fig. 4.5). Then each augmented image is fed to the CNN. The complete architecture is explained in Fig. 4.6. This augmentation is justified by the fact that galaxies images are rotation invariant: seeing the galaxy from multiple points of view helps the model predict the ellipticity of the galaxy. In terms of MC Dropout, we chose the number of MC samples to be $K = 50$, as it is a good compromise between statistical accuracy and computational speed. See Fig. 4.16 for more details on the choice of MC samples.

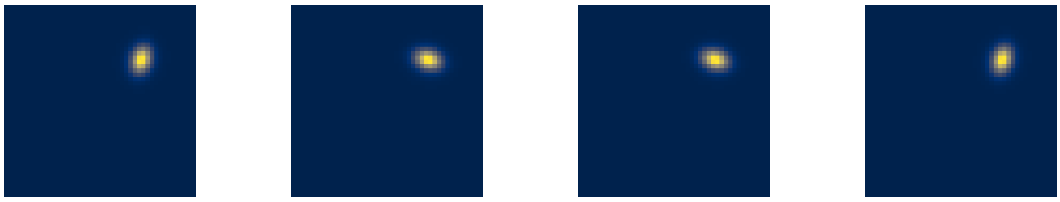
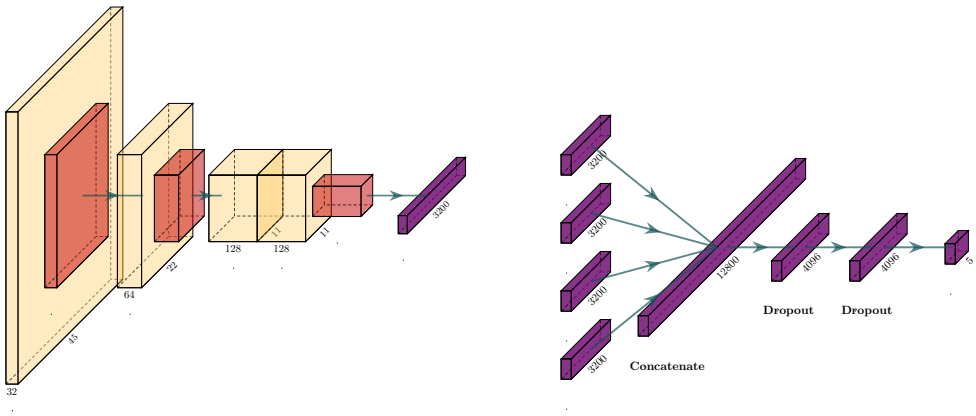


Figure 4.5.: Galaxy data augmentation. The original image, of size 256x256, is cropped at its center in an image of size 64x64. Then, the image is cropped 4 times differently by taking the 45x45 subimage at each corner, and each cropped image is rotated such that each galaxy is in the same quadrant. Each cropped image is used during training (see also Dieleman et al. (2015)).

We conducted each time two experiments, by training one BNN with a dataset of noiseless isolated galaxy images $\mathcal{D}_{noiseless}$ (as in Fig. 4.4a), and another BNN with the same architecture but this time a dataset of noisy isolated galaxy images $\mathcal{D}_{noise}$ (as in Fig. 4.4b).

Training protocol. In order to train a BNN with an MVN output, the model needs to learn both the mean and the covariance matrix. The network's training diverges when trying to learn both at the same time, forcing us to separate the training into two steps. First, we trained a simple neural network without an MVN output: we use only two output neurons representing the mean $\mu(X) = \mu_r(X) + i\mu_i(X)$, using a L_2 loss. Then, we transfered the filters of the convolutional layers into the model with an MVN output, but reinitialized the fully connected layers, and also expanding them by doubling the number of neurons, as the MVN regression is more difficult (see Fig. 4.6). This allows the model to converge smoothly as the mean of the MVN distribution has already been learned, allowing the covariance matrix to be calibrated accordingly.

This protocol works well when training on noiseless images of isolated galaxies but



(a) Convolutional layers (b) Fully connected layers

Figure 4.6.: Convolutional neural network architecture. (a) The input after augmentation has dimensions 45x45x1. Each convolutional block starts with a batch normalization layer (Ioffe and Szegedy, 2015) and has a PReLU activation (He et al., 2015). Batch normalization re-centers and re-scales inputs to make the training process more stable. The first convolutional layer is of dimension 45x45x32 with a 5x5 kernel size (in yellow), followed by a 2x2 Max-Pooling operation (in orange). The second convolutional layer is 22x22x64 with kernel size 3x3, followed by a 2x2 Max-Pooling operation. Then, we add two 11x11x128 convolutional layers with a 3x3 kernel that ends with a final 2x2 Max-Pooling operation, and the resulting feature maps are flattened into a 3200 fully connected layer (in purple). (b) Each augmented image gives a 3200 fully connected layer convolutional output (all augmented images share the same convolutional layers and filters), which are then concatenated into a 12800 fully connected layer. The two final layers have 4096 neurons in the case of an MVN regression, 2048 else; with Maxout activation (Goodfellow et al., 2013) and dropout with a rate of 0.5. The output layer has 5 neurons in the case of an MVN regression.

fails when training on noisy images. Indeed, overfitting occurs during the training of the network without MVN. When transferring the filters to the MVN model, the mean of the MVN is not well calibrated enough and the training of the BNN diverges. To fix this, we adjusted the protocol for the model without MVN, adding noise incrementally during training: we first submitted noiseless images, and modified 5% of the sample, switching from noiseless to noisy images, every 25 epochs for 1000 epochs. This protocol prevents overfitting and allows the MVN model to converge after the transfer. See Fig. 4.7 for an illustration of this training protocol.

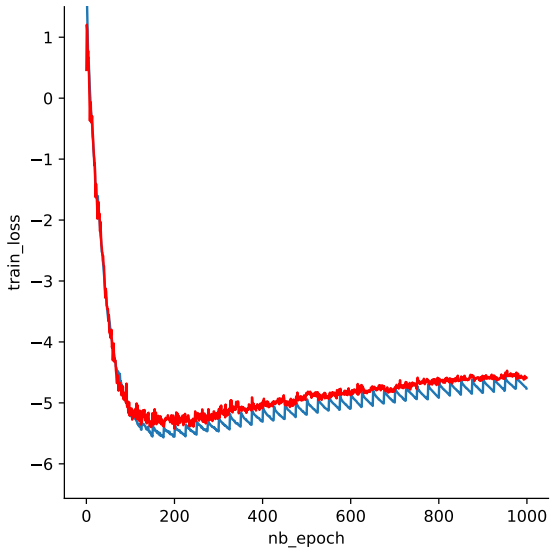


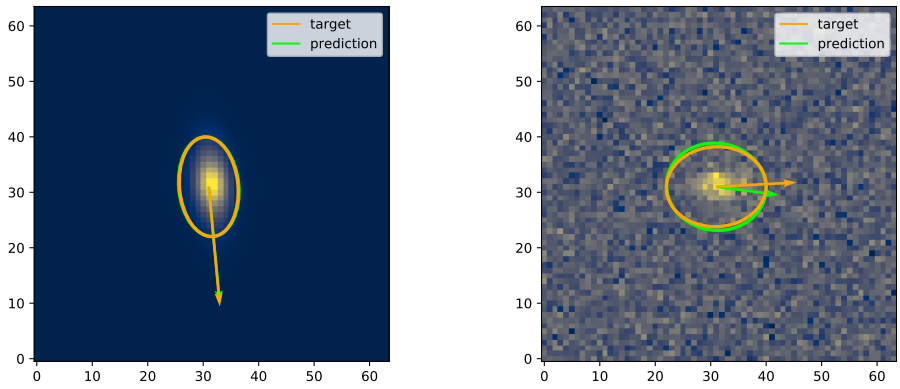
Figure 4.7.: Incremental noise training: loss convergence curve. In red: validation loss. In blue: training loss. Every 25 epochs, the dataset is updated by increasing the noise on the image, hence the sawtooth aspect of the blue curve: the training loss spikes at each update.

4.2.3. Experimental results

Estimation of noise related uncertainty. In this section we show that using an MVN as an output allows for a reliable and well calibrated estimation of the aleatoric uncertainty, i.e. uncertainty related to the noise in the data.

In order to show that estimating the ellipticity of galaxies in the presence of background noise is complex and can induce incorrect predicted ellipticity values, we first train two simple CNNs without an MVN output: one on noiseless images and one on noisy images, accordingly tested on noiseless and noisy images respectively. Fig. 4.8 shows the images of galaxy with their target complex ellipticity superimposed, as well as the predicted one.

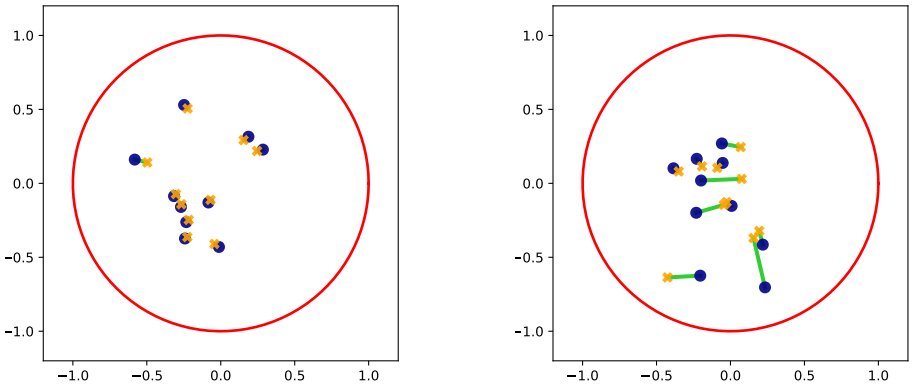
The ellipse represents the estimated shape - with a fixed scale adapted for visualization - and the arrow is the corresponding complex ellipticity - modified with half its argument in order to be aligned wih the main axis of the ellipse. On this example, we can qualitatively



(a) Predicted ellipticity without noise (b) Predicted ellipticity with noise

Figure 4.8.: Galaxy images with the predicted ellipticity superimposed on them. The arrow and the corresponding elliptic shape are rendered in an arbitrary scale for visualization purposes. In orange: the true ellipticity. In green: the predicted ellipticity.

see that the galaxy ellipticity on the noisy image is harder to estimate as the noise deforms the shape of the galaxy. Fig. 4.9 generalizes this observation as it shows a sample of the predicted ellipticities on the complex plane within the unit circle, with the target ellipticity and the difference between predicted and targeted values.



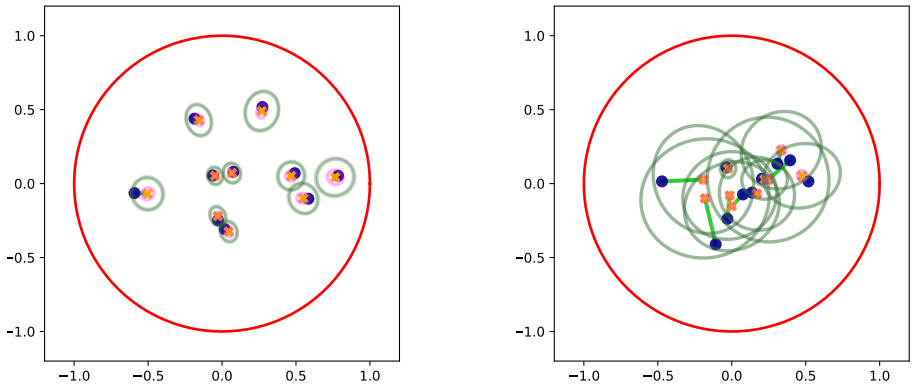
(a) Predicted ellipticities without noise (b) Predicted ellipticities with noise

Figure 4.9.: Predicted ellipticities on the complex plane. In red: unit circle. In yellow: predicted ellipticities. In blue: target ellipticites. In green: difference between true and predicted values.

While the model trained on noiseless data performs really well (Fig. 4.9a), it cannot achieve the same level of performance when trained on noisy data, losing part of its reliability (Fig. 4.9b). As such, using a simple CNN without any estimation of aleatoric uncertainty is not satisfying for our application.

We trained two Bayesian Convolutional Neural Networks with an MVN distribution,

on noise-free and noisy data respectively. These BCNNs estimate both epistemic and aleatoric uncertainties, as seen in Section 4.2.1. Like the simple CNN models, we show in Fig. 4.10, the ellipticities estimated from the BNNs on the complex plane. We also add the 90% confidence ellipses $\mathcal{E}^2_{epist.}(0.1)$, $\mathcal{E}^2_{aleat.}(0.1)$ and $\mathcal{E}^2_{pred.}(0.1)$ of the epistemic, aleatoric and predictive uncertainties respectively.



(a) Predicted ellipticities without noise (b) Predicted ellipticities with noise

Figure 4.10.: Predicted ellipticities on the complex plane. In red: unit circle. In yellow: predicted ellipticities. In blue: target ellipticites. In light green: difference between true and predicted values. In pink: 90% epistemic confidence ellipse. In dark green: 90% aleatoric confidence ellipse. In grey: 90% predictive confidence ellipse.

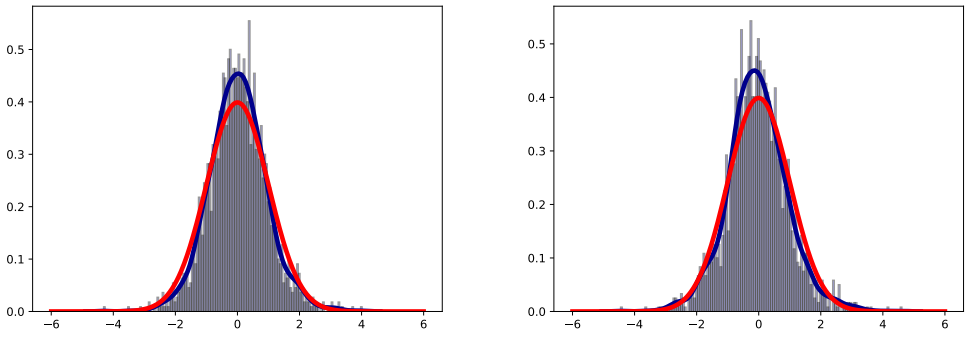
We observe that in both cases, the epistemic uncertainty is low if not negligible, meaning that the model is confident in its predictions. Put another way, all K pairs of outputs $\mu_{\hat{\omega_k}}(X)$ and $\Sigma_{\hat{\omega_k}}(X)$ are roughly equal to their mean, respectively $\hat{\mu}(X)$ and $\hat{\Sigma}_{aleat.}(X)$, so that, according to Eq. 4.13 and Eq. 4.8 , $\hat{\Sigma}_{epist.}(X) \approx 0$ and $\hat{\Sigma}_{pred.}(X) \approx \hat{\Sigma}_{aleat.}(X)$. The aleatoric uncertainty is low for noiseless images but higher for noisy ones, confirming that the noise corrupting galaxy images makes it more difficult for the model to consistently give an accurate ellipticity estimation.

Finally, in order to see if the MVN distribution is well calibrated, we standardize the output and check if the resulting distribution follows the standard distribution. More precisely, if we define

$$Z(X) := \hat{\Sigma}_{pred.}(X)^{-\frac{1}{2}}(\varepsilon - \hat{\mu}(X))\,, \tag{4.27}$$

then the distributions of its two independent components $z_1 \sim Z(X)_1$ and $z_2 \sim Z(X)_2$ should be equivalent to the standard distribution $\mathcal{N}(0,1)$. Note that this is true only because all K output MVNs coincide with each other. Fig. 4.11 shows that the standardized distributions for the model trained on noisy images are indeed well calibrated and therefore the model is neither overestimating nor underestimating the predictive uncertainty.

Estimation of blend related uncertainty. In the previous part we showed that our BNNs are well calibrated to estimate aleatoric uncertainty. Here we submit outliers



(a) Distribution of z_1 (b) Distribution of z_2

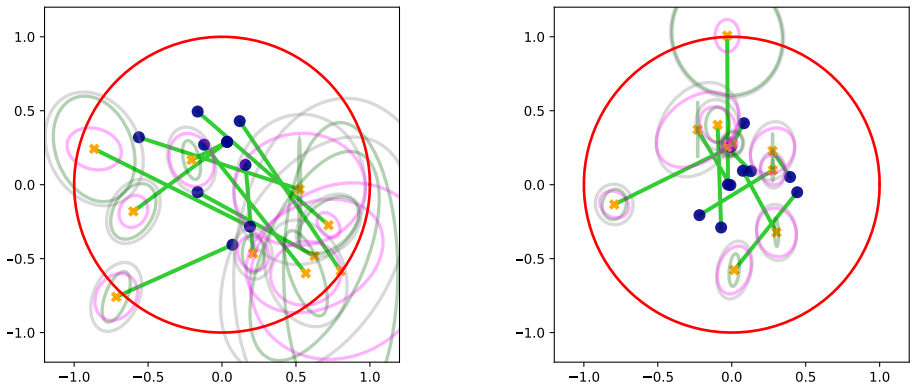
Figure 4.11.: Histogram of the standardized distributions on the model trained with noisy images. In red: standard bell curve. In blue: histogram of the standardized distribution with the smoothed curve.

to the networks in order to study the impact on epistemic uncertainty and whether it can be used to detect them. Our models have only been trained on images of isolated galaxies, but astrophysical images can contain multiple overlapped galaxies. In that case, asking the model to measure a single ellipticity does not make sense. If the epistemic uncertainty behaves as expected, then its measurement would allow us to detect when a predicted ellipticity is incorrect due to the presence of multiple galaxies in the image. We fed images of blended scenes to the two models trained on isolated galaxies (with or without noise), adding noise to the blended scenes only for the model trained on noisy images.

Results shown in Fig. 4.12 demonstrate that in both cases the predictions are particularly inexact when compared to the target ellipticity of the central galaxy. Also, and as expected, the epistemic uncertainty is much higher for these blended scenes than for isolated galaxy images. However, the aleatoric uncertainty gives incoherent values as the model has not been trained to evaluate it on blended images: notice how the aleatoric ellipses are more flattened with a lower area. Fig. 4.13 permits to visualise the behavior of the epistemic uncertainty. It shows how the ellipticities sampled with dropout slightly diverge compared to the mean prediction. Here the model cannot give a consistent answer and therefore its prediction should be deemed untrustworthy.

To quantify the quality of the epistemic uncertainty when it comes to detecting incoherent predictions due to outliers, we computed the ROC curves for each uncertainty type. More precisely, we reduce each covariance matrix (aleatoric, epistemic and predictive) to a scalar by computing its determinant. We interpret these estimates as a scoring function to assess whether an image is an outlier, i.e. a blended image: the higher the score, the more likely the image contains a blend. Finally, we compute for each of these scoring functions its ROC curve. We repeat that process for both networks trained with noisy and noiseless data. The results are shown in Fig. 4.14.

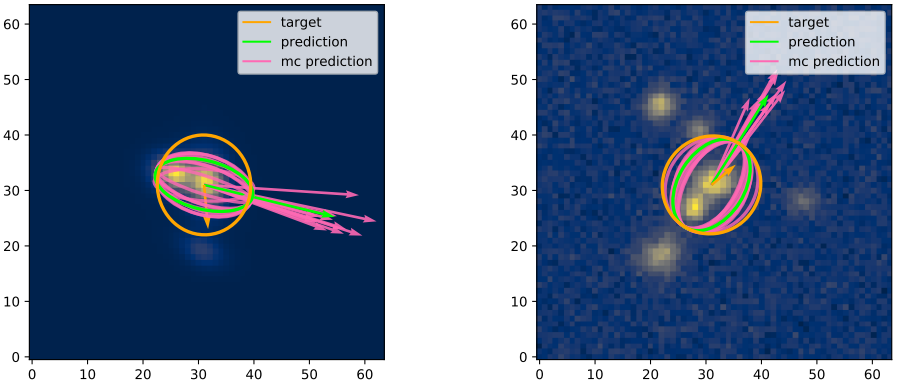
These ROC curves are also summarized by their associated Area Under Curve (AUC) on Table 4.1. The epistemic uncertainty clearly appears as the most consistent “metric”



(a) Predicted ellipticities, without noise

(b) Predicted ellipticities, with noise

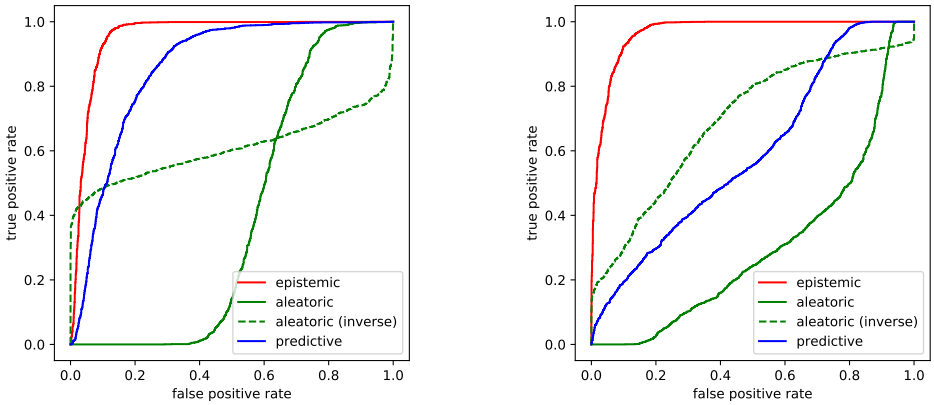
Figure 4.12.: Predicted ellipticities on the complex plane for blended galaxies images. In red: unit circle. In yellow: predicted ellipticities. In blue: target ellipticites (label of the centered galaxy). In light green: difference between true and predicted values. In pink: 90% epistemic confidence ellipse. In dark green: 90% aleatoric confidence ellipse. In grey: 90% predictive confidence ellipse.



(a) Blended galaxies without noise

(b) Blended galaxies with noise

Figure 4.13.: Blended galaxies images with the predicted ellipticity superimposed on them. The arrow and the corresponding elliptic shape are rendered in an arbitrary scale for visualization purposes. In orange: the true ellipticity (label for the galaxy in the center). In green: the predicted ellipticity. In pink: the individual MC dropout predicted ellipses. The green ellipticity is therefore the mean of the pink ones. On both images the prediction is uncertain as the individual MC samples slightly diverge from the mean.



(a) ROC curve, model without noise

(b) ROC curve, model with noise

Figure 4.14.: ROC curves for detecting outliers for aleatoric, epistemic and predictive uncertainty. (a) ROC curve, model without noise. (b) ROC curve model with noise. Since the aleatoric ROC curve gives incoherent answers on outliers (see Fig. 4.12), we also plot the complementary classifier as a dashed line.

Uncertainty	AUC noiseless	AUC noise
Epistemic	**0.956**	**0.969**
Aleatoric	0.394	0.306
Aleatoric (inverse)	0.606	0.694
Predictive	0.856	0.594

Table 4.1.: AUC values for all uncertainties, for the model with and without noise. Here, the epistemic uncertainty is clearly the best to detect outliers, as its AUC value is close to 1 in both noisy and noiseless datasets.

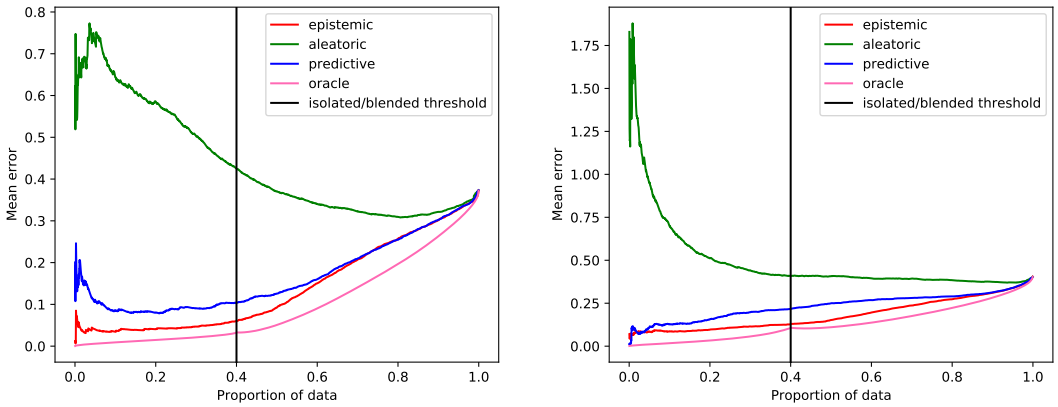
to detect outliers and therefore to give useful information about the confidence in the model predictions. Even the predictive uncertainty performs worse than the epistemic one. This is especially true in the presence of noise since the aleatoric uncertainty then occupies a more important part of the predictive one compared to the noiseless case. Notice that the aleatoric ROC curve is mostly below the diagonal with an AUC below 0.5, meaning it performs worse than a random classifier. This is due to the fact that the model has not been trained to evaluate aleatoric uncertainty on blended scenes. As seen in Fig. 4.12, the aleatoric ellipses are more flattened in the blended cases, meaning their determinant is lower. Thus the aleatoric uncertainty is on average lower on blended scenes when compared to isolated ones.

To compensate, results of the complementary classifier for the aleatoric uncertainty are shown. It is still not as satisfying as the epistemic uncertainty. While using epistemic uncertainty to identify inconsistent predictions due to a lack of knowledge is highly effective, we note that few blended images still have low epistemic uncertainty due, for instance, to a large galaxy that obstructs all of the other ones, making the image actually closer to an isolated galaxy image. However in the context of shear estimation, this is not an issue as the occulted galaxy is simply ignored.

Finally, we evaluate how each type of uncertainty is a reliable representation of the risk of error in ellipticity prediction. Unfortunately, in the presence of blended images, the predictive distribution is no longer a simple MVN but a mixture of K well separated Gaussian distributions. The normalization process that allowed us to obtain the results presented in Fig. 4.11 is no longer applicable here. It is still possible to study the relationship between the uncertainty and the ellipticity prediction error testing a trivial rule: the higher the uncertainty, the more important we expect the error to be. To do so, we do three sorting of the images according to each uncertainty type, from the lowest uncertainty to the highest, on a scale from 0 to 0.4 for isolated objects, and from 0.4 to 1 for blended scenes. This choice of proportion is consistent with the actual proportion of blended objects expected to be seen in LSST (see Section 4.1.1. We then compute the mean ellipticity error considering the proportion of the sorted data from 0 to 1. For blended scenes the ellipticity prediction error is computed w.r.t. the ellipticity of the centered galaxy. We repeat this experiment twice, for networks trained on noiseless and noisy data. Finally, we add an "oracle" curve where the data is sorted directly according to the ellipticity prediction error which represents a perfect sorting. Results are shown in Fig. 4.15.

Once again, epistemic uncertainty proves to be best suited to anticipate ellipticity predictive error. The samples with the lowest epistemic uncertainty have the lowest mean ellipticity error and conversely, while samples with low aleatoric uncertainty already have high mean error. Consequently, on real astrophysical data, when the predictive ellipticity error is obviously unknown, relying on the epistemic uncertainty to reject, or minimize the impact of a sample because of its probable predictive error is the best way to go.

Validation of the number of dropout samples K. In order to validate the estimation of the uncertainties, we also needed to verify if the number of MC Dropout



(a) Mean error curve, model without noise

(b) Mean error curve, model with noise

Figure 4.15.: Mean error curves w.r.t. data proportion for aleatoric, epistemic and predictive uncertainty. (a) Mean error curve without noise. (b) Mean error curve with noise. In black the threshold between the proportion of isolated galaxies: $[0, 0.4]$ and blended galaxies: $[0.4, 1]$. In pink the oracle curve, where the data is sorted by the predictive error. The closest a curve is to the oracle the better.

samples K allows us to have a reliable estimation of the uncertainties. Fig. 4.16 shows the dependence of the uncertainty estimators $\hat{\mathcal{U}}$ for the aleatoric, epistemic and predictive uncertainties with respect to K in the range between 1 and 100. We can clearly see that the estimator converges around a certain value as soon as $K \geq 30$, which validates our choice of $K = 50$.

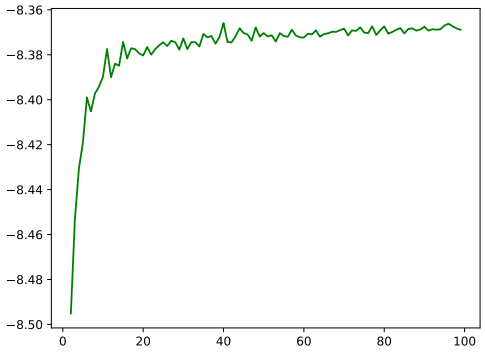
4.2.4. Conclusion

In this section, we presented a Bayesian approach to estimate the posterior distribution of galaxy shape parameters using convolutional neural networks and MC-Dropout. In addition to a precise measurement of the ellipticities, this approach provides a calibrated estimation of the aleatoric uncertainty as well as an estimation of the epistemic uncertainty. We showed that the latter is behaving according to expectations when applied to different kind of galaxy images, and is well-suited to identify outliers and to anticipate high predictive ellipticity error. These results confirm the suitability of Bayesian neural networks for galaxy shape estimation and incite us to continue exploring their use to go from ellipticity posterior distributions, estimated from multi-band galaxy images, to cosmic shear estimation.

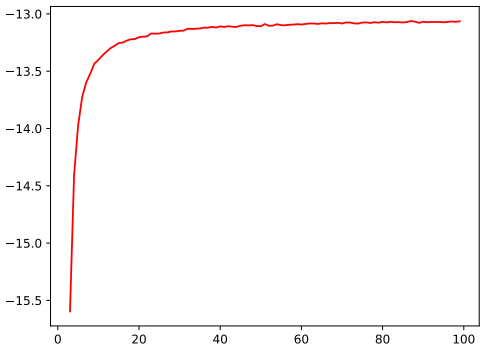
4.3. Galaxy mapping with a Bayesian CNN

4.3.1. Uncertainty estimation and galaxy detection

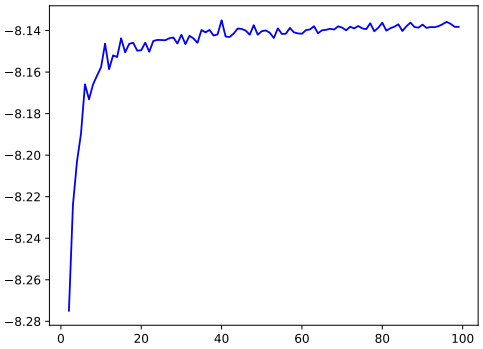
In this part, we will consider another experiment to validate the uncertainty estimates of the Bayesian Convolutional Neural Network for galaxy regression. As stated before,



(a) Aleatoric uncertainty estimation.



(b) Epistemic uncertainty estimation.



(c) Predictive uncertainty estimation.

Figure 4.16.: Validation of the number of MC Dropout samples: uncertainty $\hat{\mathcal{U}}$ with respect to the number of MC samples K.

ensuring that BNNs behave in a manner to reliably detect unknown data and give calibrated uncertainties is crucial in the application for cosmic shear estimation. In the previous experiments, the model was only trained with images of isolated and centered galaxies. While we gave the model images of blended galaxies during test time, even then there was at least one galaxy at the center of the image. An interesting interrogation to ask is: what happens if we give the model an image of an isolated, uncentered galaxy ?

If we expect the epistemic uncertainty to be higher for uncentered galaxies, we can actually use it to detect objects at the same time. A window of a map centered on coordinates (x,y) has a galaxy at its center if and only if the epistemic uncertainty associated to the model's prediction is low.

Following this train of though, let us consider a map $\mathcal{M} = (m_{i,j}) \in \mathcal{M}_m(\mathbb{R})$ comprised of N galaxies, such that $n < m$ where (n, n) is the input size of image fed to the CNN, see Fig. 4.17 as an example.

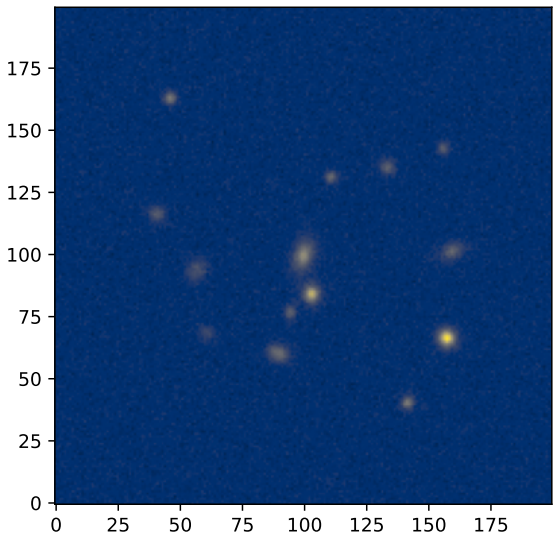


Figure 4.17.: Example of a map of galaxies. Here the map is 200×200 pixels.

We can define a sliding window of size (n, n) in $\mathcal{M}$

$$\forall (i,j) \in [\![1, m-n+1]\!]^2, X_{i,j} := \mathcal{M}_{[\![i,i+n-1]\!],[\![j,j+n-1]\!]}\,, \tag{4.28}$$

where $\mathcal{M}_{[\![i,i+n-1]\!],[\![j,j+n-1]\!]}$ is the submatrix of size (n, n) comprised of the coefficients of $\mathcal{M}$ in rows and columns in $[\![i, i+n-1]\!], [\![j, j+n-1]\!]$ respectively.

Then, we want to define a new map, $\boldsymbol{\mathcal{U}}^{epist.} \in \mathcal{M}_{m-n+1}(\mathbb{R})$, such that

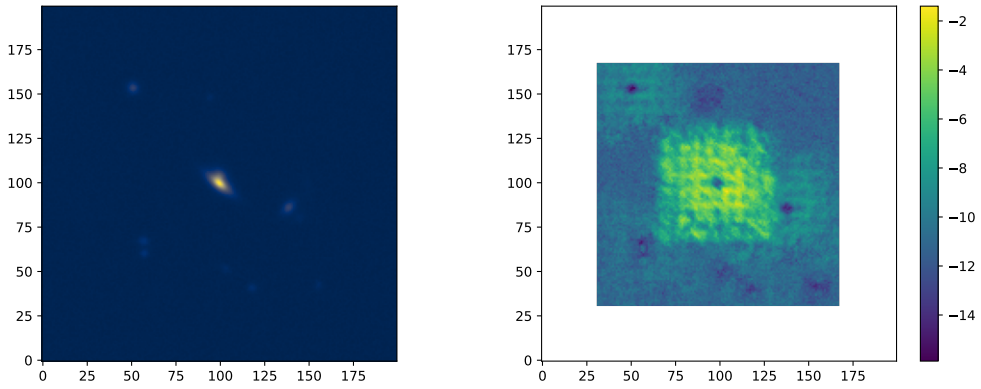
$$\forall (i,j) \in [\![1, m-n+1]\!]^2, \boldsymbol{\mathcal{U}}^{epist.}_{i,j} := \log\left(\hat{\mathcal{U}}_{epist.}(X_{i,j})\right)\,. \tag{4.29}$$

This map can similarly be defined for aleatoric and predictive uncertainties: $\boldsymbol{\mathcal{U}}^{aleat.}$ and $\boldsymbol{\mathcal{U}}^{pred.}$ respectively. Specifically, these maps represent the level of uncertainty the model has at a certain point of $\mathcal{M}$. What we expect to see is minima of epistemic uncertainty

at the coordinates where the galaxies are. In a sense, if the epistemic uncertainty is calibrated correctly, $\boldsymbol{\mathcal{U}}^{epist.}$ should map the position of the galaxies in the image. For better visualization, we take the log-uncertainty.

4.3.2. The checkerboard pattern problem

Given these definitions, let us try a first experiment by simply plotting $\boldsymbol{\mathcal{U}}^{epist.}$ given a certain map $\mathcal{M}$. We retrain our model without data augmentation. If we do this, we get the result shown in Fig. 4.18.



(a) Galaxy map $\mathcal{M}$

(b) Epistemic uncertainty map $\boldsymbol{\mathcal{U}}^{epist.}$

Figure 4.18.: Galaxy map (left) and epistemic uncertainty map (right), first experiment.

As expected, we can see minima of the log-uncertainty at the position of the different galaxies, which confirms our intuition. However, and this is especially noticeable around the galaxy at the center of the map (which is the brightest), we can see a sort of artifact, as the uncertainty varies from high to low in a very regular pattern, albeit blurred.

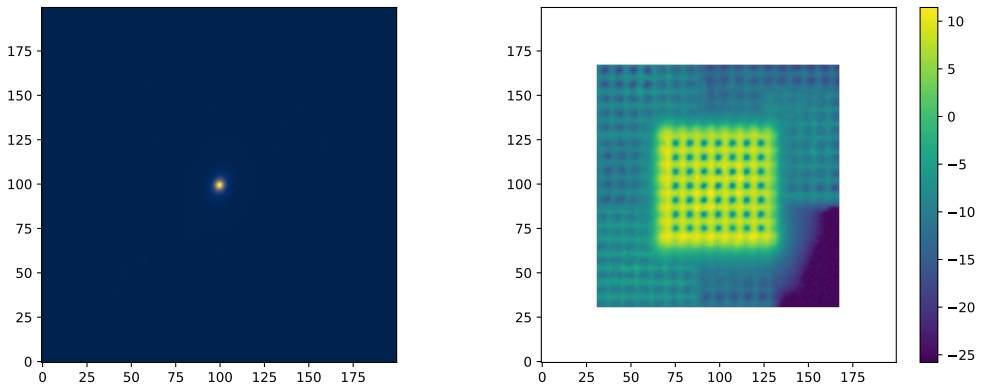
Our first intuition was with the model itself rather than the data, as regular centered images make the model, and its uncertainties, behave correctly (cf. Section 4.2). To explain the experiments more clearly, we will describe in more detail the network architecture.

There are four convolutional layers, three of which end with a pooling operation that reduces the height and width of the feature map by two. This pooling operation is either a maxpooling, or a convolution with stride 2 (meaning that the convolution operation has a step of 2 in both horizontal and vertical directions). In total, the image is reduced by a factor 8 in each dimension, except the number of channels. This means that if the original image is 64×64 pixels, then the final feature maps will have 8×8 voxels.

Then, the final feature map is flattened and sent through two fully connected layers with a ReLU activation, which are ultimately used to predict the ellipticity.

Now, let us repeat the experiment, but this time removing the ReLU activation in the fully-connected layers. If the pattern is blurred, removing non-linear activations could help to see it more clearly.

The results in Fig. 4.19 tells us that the pattern is indeed more visible without the ReLU activation. This pattern is extremely regular, looking like a checkerboard that forms a grid of 8 by 8 around a galaxy, and these patterns are overlapping between each galaxy in the map. Obviously on this map, the galaxy at the center is much brighter than the others which are less visible, but the model detects them nonetheless.



(a) Galaxy map $\mathcal{M}$

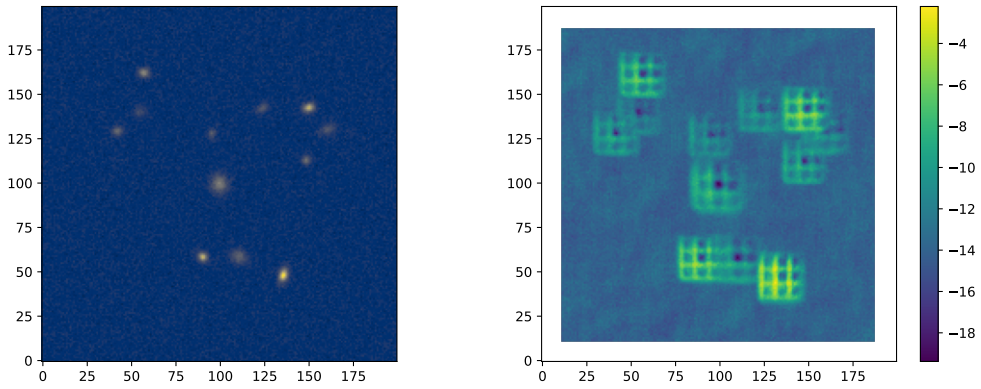
(b) Epistemic uncertainty map $\boldsymbol{\mathcal{U}}^{epist.}$

Figure 4.19.: Galaxy map (left) and epistemic uncertainty map (right), experiment without ReLU activation.

This pattern is problematic in a few ways. First, it breaks our assumption that the uncertainty will be minimal if and only if the window $X_{i,j}$ is centered at a certain galaxy, which defeats the purpose of using the uncertainty to create a map localizing the different galaxies. Secondly, without any further explanation, how can we explain that the model has low epistemic uncertainty on images of shifted galaxies, which it has never seen during training ? Finally, the apparent unpredictable behavior (which was unexpected here) is problematic in applications where we expect the uncertainty to have some form of meaning for the end user. Trying to find an explanation to pathological uncertainty behavior is part of the work to deconstruct the so-called "black box" that are neural networks.

Although non-linear operations, such as activation functions, do reduce the effect of the pattern, it feels more like a "band-aid" which does not explain the underlying reason behind it. To understand where this pattern comes from, notice that its 8 by 8 dimension actually coincides with the size of the final feature maps, before the fully connected layers. Indeed, the original image is 64 by 64, and the convolutional part of the network has three 2 by 2 pooling operations which reduce the dimension of the image, effectively dividing the dimensions by 8 at the end of the process.

To confirm this hypothesis, let us simply reduce the size of the window (and input of the network), to see if the size of the pattern also decreases. This time, the window size will be $n = 24$, and the final feature map will be 3 by 3. Fig. 4.20 shows that the checkerboard pattern is effectively linked to the size of the feature map, and the overall convolutional architecture.



(a) Galaxy map $\mathcal{M}$

(b) Epistemic uncertainty map $\boldsymbol{\mathcal{U}}^{epist.}$

Figure 4.20.: Galaxy map (left) and epistemic uncertainty map (right), experiment without ReLU activation, with a smaller window size.

Consider then the feature map of 3 by 3 voxels, and each of their receptive fields. By applying Equation 2.13, we find a receptive field of $r_0 = 25$ pixels for the CNN (see Fig. 4.21).

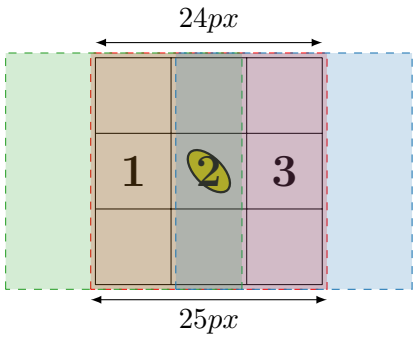
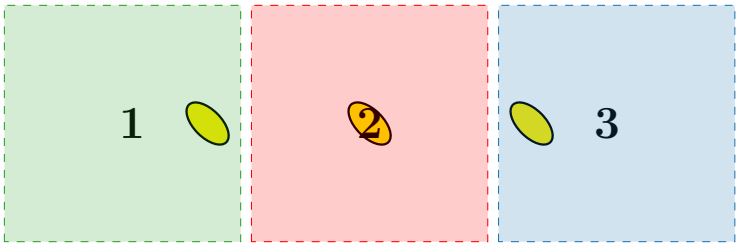


Figure 4.21.: Last feature map in the CNN (3 by 3 grid in black) in the original image (24 by 24 pixels), with the receptive fields of the voxels in the center row (in dashed line and color), with the object (galaxy) at the center of the image.

For the sake of simplicity, we will only illustrate the receptive fields of each of the voxels in the center row of the last feature map. In Fig. 4.22, we show the point of view of each receptive field in that center row, and the position of the galaxy in each of the feature maps. We can see that, when considering a dataset with only images of centered galaxies, the receptive fields of the last feature map sees the galaxy in multiple fixed positions every time. Then, these voxels are flattened into the fully connected part of the network to make its prediction. The classifier uses the redundant information contained in these voxels to predict the ellipticity. This translates to a form of overfitting as the galaxy will predict the correct ellipticity with low uncertainty only if the patterns seen in the receptive fields correspond to the ones seen during training; for instance, the patterns

seen in RF1, RF2 and RF3 in Fig. 4.22

Therefore, if we slide the window by 8 pixels to the left, the filters that detect when the galaxy is on the left, those corresponding to the third voxel (RF3), activate and fire a signal to the fully connected layers that allow for the ellipticity estimation with low epistemic uncertainty.



(a) RF 1 (b) RF 2 (c) RF 3

Figure 4.22.: Receptive fields for each of the center voxels of the feature map, and the respective galaxy position for each of them.

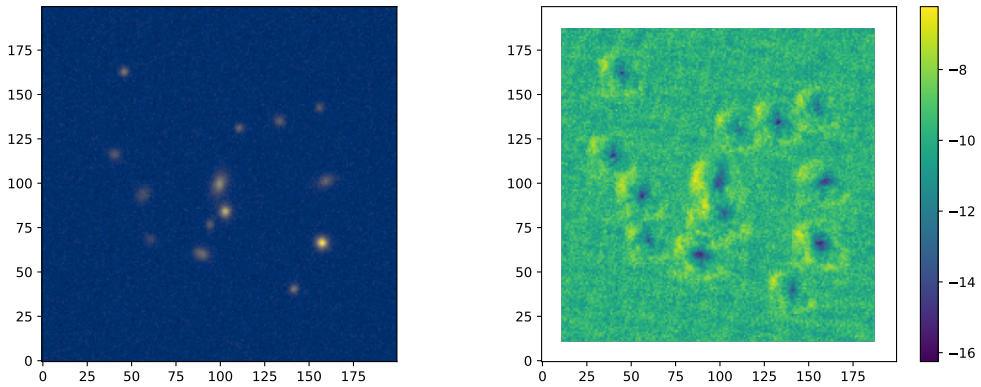
On the contrary, if we slide the window by 4 pixels to the left so that the galaxy will be exactly between two voxels in the image, the position of the galaxy on all of the receptive fields will shift in such a way that no receptive field will correspond to the centered case, leading to a high epistemic uncertainty.

Combining these two explanations, we effectively get that checkerboard pattern with very regular variations of epistemic uncertainty in the vicinity of a galaxy. It can also explain why the model can output low epistemic uncertainty predictions even on unseen data: the structure of convolutional layers allow for generalization of visual patterns, and that includes symmetry, as centering the galaxy gives fixed positional information on the object to analyze. When the galaxy is positioned in such a way that the feature map recovers the symmetry learned in the case of a centered galaxy, the epistemic uncertainty will drop.

In order to have coherent epistemic uncertainty estimation given our dataset, and avoid the appearance of such pattern, the solution is simple: break the symmetry by perturbing the positions of the galaxies in the training dataset. We retrain our model by randomly slightly shifting the galaxies by sampling a shift (s_x, s_y) from two independent uniform distributions defined in $[\![-10, 10]\!]$ such that each galaxy will be centered on (s_x, s_y), with a new shift sample for each image. Then, we fine tune the model on centered galaxies. This training forces the model to regulate its kernels to not learn to recognize specific galaxy positions, but only the centered case.

By repeating the experiment in the case of a 3x3 final feature map, we get the result shown in Fig. 4.23, in which we can see that the checkerboard pattern is effectively removed. We can still see a “halo” of higher epistemic uncertainty around the image, roughly corresponding to the size of the window. We know that we can expect the epistemic uncertainty to be on average constant when the window does not include any galaxy: the image is only comprised of Poisson noise, which has the same pattern

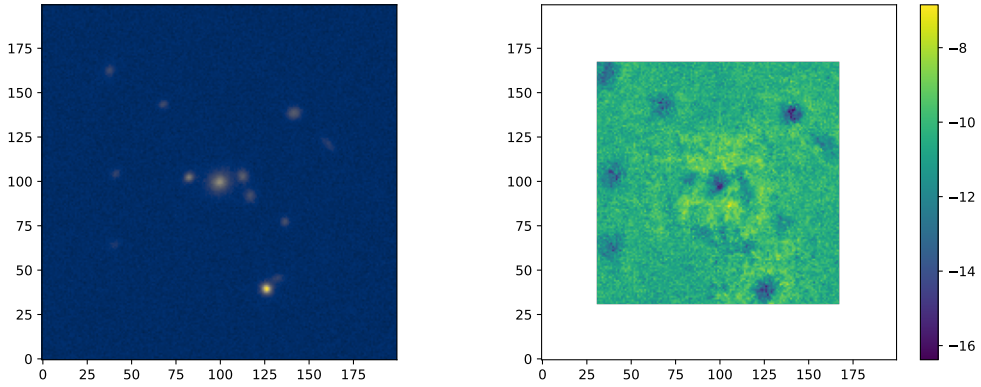
everywhere. On the other hand, when the window slides away from one galaxy in such a way that the galaxy is in the border of the window, the presence of the galaxy will activate the feature maps in a way that induces high epistemic uncertainty, as the model does not have knowledge about such data.



(a) Galaxy map $\mathcal{M}$ (b) Epistemic uncertainty map $\boldsymbol{\mathcal{U}}^{epist.}$

Figure 4.23.: Galaxy map (left) and epistemic uncertainty map (right), experiment with shift dataset training and centered dataset fine tuning.

For the sake of showing that shift training solves the checkerboard pattern with any CNN architecture, we also show the results in the case where we have an 8 by 8 final feature map in Fig. 4.24.



(a) Galaxy map $\mathcal{M}$ (b) Epistemic uncertainty map $\boldsymbol{\mathcal{U}}^{epist.}$

Figure 4.24.: Galaxy map (left) and epistemic uncertainty map (right), experiment with shift dataset training and centered dataset fine tuning, and an 8 by 8 final feature map.

4.3.3. Conclusion

At the end of these experiments, we were able to produce uncertainty map which are coherent with galaxy positions. Although these maps are not able to reliably identify the

position of galaxies when those are blended together, we still see a very strong correlation between low epistemic uncertainty and galaxy position when the galaxies are separated.

However, studying the patterns in uncertainty maps led us to the discovery of a somewhat unexpected behavior of that uncertainty, which was linked to the architecture of the model used, as well as the dataset. Trying to understand where this behavior came from and how to solve it helped us to shed some light on Bayesian Neural Networks and how uncertainty estimation can relate to the choice of model and dataset, an important point to deconstruct the "black box" of such models, especially here in an applied machine learning context.

4.4. Network coaching

In the following section, we will present some preliminary work done on the technique called *network coaching*. Network coaching is a method to improve the performance of a neural network on a complex dataset by starting the training of the network on "simple" data and then increasingly augmenting the complexity of the data during training.

In terms of the problem at hand, we would like to train our neural network to process images of blended scenes to predict the ellipticity of the galaxy at the center. More precisely, we define the dataset with two galaxies at most, one of which is at the center of the picture, and the goal is to predict the complex ellipticity of the galaxy at the center of the image.

The second uncentered galaxy, which we will call the "companion galaxy", is placed at a random point of the image, by sampling its distance from the center r and angle θ using a uniform distribution. θ is randomly sampled in the interval $[0, 2\pi]$ and r in $[0, D/2]$ where D is the width/length of the image. So, this companion galaxy can be at any place in the image, which means that the two galaxies can be separated or blended.

As a reminder, the proportion of blended images is expected to be around 60% (Sanchez et al., 2021). Consequently, we simulate our dataset to have 60% of the images to be blended scenes and the remaining 40% to be isolated scenes.

Training a neural network on such a diverse and complex dataset is hard, and the question is whether introducing complexity to the dataset gradually would help the network learn more effectively. We will present ideas and experiments showing preliminary results about this problem.

4.4.1. Introducing complexity to the dataset

The idea behind network coaching is introducing prior knowledge about the dataset and the regression task to the learning process.

Now, there are two main schools of idea about helping the network learn more efficiently. The unsupervised way notably englobes the field of *active learning*, where the network chooses itself the new points to be labeled and learn from them. Houlsby et al. (2011) for instance proposed a Bayesian approach to active learning by selecting the most informative data points to be labeled, i.e. the data points $x \in \mathcal{X}$ with the highest conditional mutual

information $\mathbb{I}[\omega, y \mid x, \mathcal{D}]$. In astrophysics application, and more precisely galaxy data, Walmsley et al. (2019) used a Bayesian CNN for classifying galaxy morphologies on the Galaxy Zoo dataset (Lintott et al., 2008). Then, they used Bayesian active learning to select new data to be labeled for training the network, by selecting the most informative new images to be labeled.

Our work is more on the supervised spectrum of efficient network learning. Indeed, the whole data selection process is entirely supervised, with no decision being made by the network itself. The situation is however different from Walmsley et al. (2019): since our data is simulated, we can generate as much data (and labels) as we want, and we can use that ability to craft a dataset $\mathcal{D}$ that is tailored to solve the regression task at hand.

Let us return now to the definition of the algorithm of network coaching. The main observation we can make is that the easiest case to learn the ellpiticity is when the two galaxies are completely separated, and when they are blended, it depends on how much they are superimposed. Given that fact, it follows that we can expect a more efficient learning process by slowly increasing the blending rate of the dataset, i.e. start with a dataset with only isolated pictures and slowly converging to a dataset of 60% blended images and 40% isolated ones.

Instead of slowly replacing 60% of the dataset by fully blended and separated scenes, we create new pictures by sampling the distance of the companion galaxy from the center galaxy r in $[\alpha, D/2]$, where α is initialized at $D/2$ and then slowly decreased in a linear manner until it reaches 0. That way, the 60% part of blended scenes start as fully isolated scenes and then becomes more and more complex as more variety and complexity (quantified as the distance r which is correlated to the blend rate, i.e. how much the two galaxies are superimposed) is introduced.

The full procedure is described in Algorithm 4. The function "Generate" takes α as an input and outputs a dataset $\mathcal{D}$ generated as described above. The function "Train" simply takes the dataset $\mathcal{D}$ and model f_ω and trains it for N epochs.

Algorithm 4 Galaxy network coaching algorithm

Input: Number of epochs per dataset update N, number of dataset updates M, CNN f_ω, image size D, image generator Generate.
Output: Trained network f_ω

$\alpha \leftarrow D/2$
for $i = 1$ *to* M **do**
  $\mathcal{D} \leftarrow \text{Generate}(\alpha)$
  $f_\omega \leftarrow \text{Train}(f_\omega, \mathcal{D}, N)$
  $\alpha \leftarrow \frac{D}{2}(1 - \frac{i}{M})$
end
return *Trained neural network* f_ω

4.4.2. Experiments

Network coaching experiments. Several experiments were made to analyze the efficiency of the network coaching approach. All of them revolved around analyzing the performance of the network on diverse metrics, notably the empirical risk on both the validation and training dataset.

First of all, we tried three different training processes. The "classical" learning procedure by simply generating the dataset $\mathcal{D}$ in its final state and training the model on that dataset directly, then the network coaching procedure (as in Algorithm 4), and a Bayesian active learning approach by selecting at each dataset update the most informative data (Houlsby et al., 2011), more precisely we generate 20000 images, we select the 5000 most informative images and then complete the dataset by choosing 5000 random images among the remaining 15000 images.

Then, after the training process, we show the empirical risk histograms on the three models, in Fig. 4.25. While we did not achieve significantly better results than existing methods, a more detailed study of the different parameters would be warranted and might lead to further improvements.

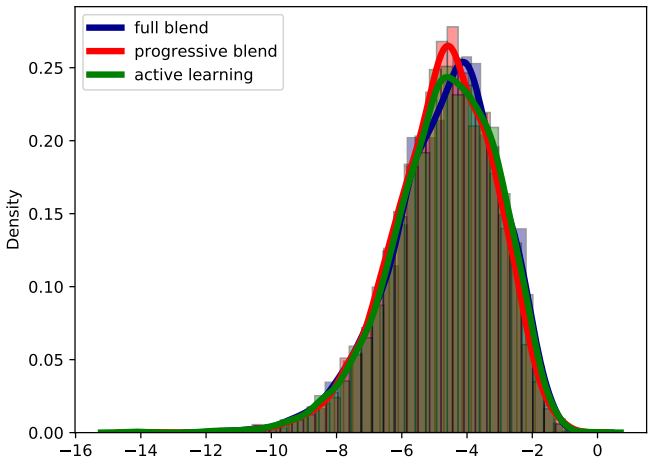


Figure 4.25.: Empirical validation risk distribution on different models. x-axis: validation risk. y-axis: Density of elements in the test set. In blue: model trained without network coaching or active learning. In red: network coaching. In green: active learning. Lower (left) is better.

Another experiment that follows is to partition the dataset by their relative loss value from the model. More precisely, given the dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, we define the ensemble of epistemic uncertainty values $\mathbf{U} := (\hat{\mathcal{U}}_{epist.}(x_i))_{i=1}^N$ and sort it by increasing order. We separate our dataset in four parts, by sorting each element by its epistemic uncertainty. Once the dataset is sorted that way, it is partitioned by the 50% with lowest epistemic uncertainty, then the 30% next, 15% and finally 5% where the model is the

most uncertain and lacks the most knowledge. For each dataset update (from Algorithm 4), we reevaluate the epistemic uncertainty at these four (redefined) datasets. In a sense, this partition represents different parts of the dataset with increasing difficulty for the model to learn.

In Fig. 4.26, we see the performance of the model with or without coaching, on these four different datasets. Each column represents the empirical risk on a different dataset, from left (most certain) to right (most uncertain). The network coaching model can achieve better results on data that is more uncertain compared to a "vanilla" approach, at the cost of performance on data with less epistemic uncertainty. This means that our approach does not perform better on average but only in some cases where the uncertainty is higher.

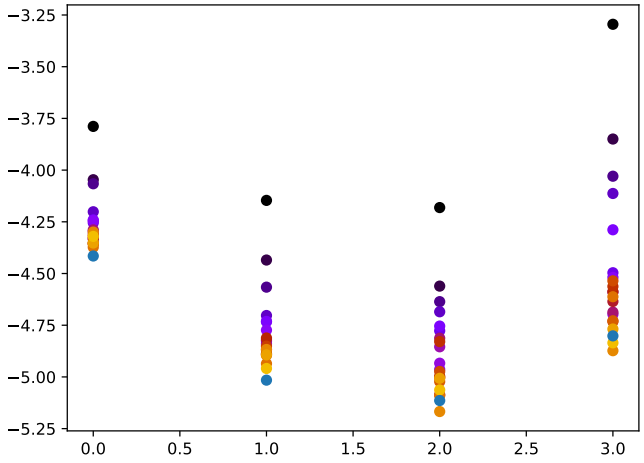


Figure 4.26.: Empirical risk on different degrees of uncertainty. In blue: model trained without network coaching or active learning. Gradient from black to red: network coaching. Black is first dataset update, yellow is last. Each column from left to right represents a different subdataset. Left is most certain and right is the most uncertain.

Active learning experiments. A few other experiments were made to study the effectiveness of the active learning approach, and how to improve it. So far, active learning consisted of selecting the new data that was the most informative for the model, i.e. related to epistemic uncertainty.

Another way to look at the problem is to take interest in the value of the loss function, more precisely the difference between the empirical mean loss and the empirical variance of the loss on a single point

$$\hat{\mathcal{L}}(X,y) := \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}(f_{\hat{\omega}_k}(X),y)\,, \tag{4.30}$$

$$\hat{\sigma}^2_{\mathcal{L}}(X,y) := \frac{1}{K}\sum_{k=1}^{K}(\mathcal{L}(f_{\hat{\omega}_k}(X),y) - \hat{\mathcal{L}}(X,y))^2\,, \tag{4.31}$$

with $\hat{\omega}_k \sim q_\theta(\omega)$.

The idea would be to select the data with the highest potential of reducing the overall empirical risk. Points with high variance and error should then be preferred. In Fig. 4.27, we show the plot of the variance $\hat{\sigma}^2_{\mathcal{L}}(X,y)$ vs. the mean $\hat{\mathcal{L}}(X,y)$ for points in the validation dataset, sorted in increasing order in the x-axis. We also show the 20% points with the highest epistemic uncertainty in red, which shows a correlation between epistemic uncertainty and high variance and mean of the loss function (most red points are in the upper part of the curve).

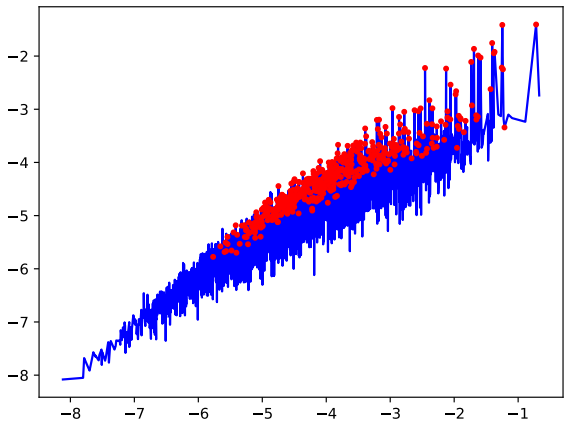
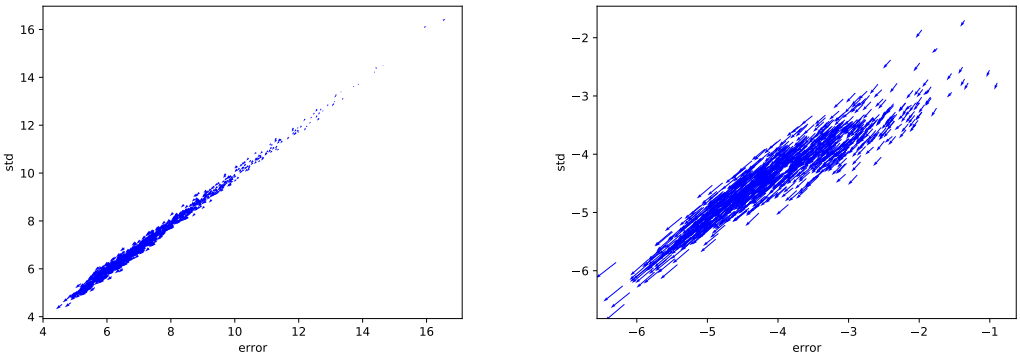


Figure 4.27.: Variance of the loss (y-axis) w.r.t. the mean of the loss (x-axis). In red are the top 20% points with the highest epistemic uncertainty.

This experiment comforts us in the idea that selecting the points that have the most mean value **and** variance for training is a good strategy to learn a Bayesian neural network efficiently. Another experiment to see how effective active learning is, is to study the evolution of the data points in the variance/mean error space. In Fig. 4.28, we plot for each data point a vector representing the evolution before and after a dataset update during active learning, which happens every 50 epochs. During the first training step, a lot of data points are in the "tail", with high variance/mean loss. By the end of the last dataset update, the overall variance/mean loss of the ensemble of the dataset has been very much reduced[5], and the vector field is now more compressed in scale.

[5]Notice how the values of the axes have reduced.



(a) First step (b) Final step

Figure 4.28.: Evolution of the variance/mean loss during training. Each vector represents the evolution of the variance/mean loss of a single image before and after a dataset update for active learning. The axes are in log scale, and the arrows are scaled down for better visualization. The length of the arrows are proportional to the change of variance/mean loss of a data point. (a) First update during training. (b) Last update during training (20th update).

4.4.3. Conclusion

This experiment concludes the exploration of network coaching and the unsupervised active learning counterpart. We unfortunately did not have time to study more in depth this subject during this thesis. However we found these first insights very intriguing and interesting, and it would be a worthy problem for further research.

In the next chapter, with the aim to have even more transparency with regards to deep learning models, we will present Explainable AI and the main concepts used to make deep learning models more interpretable. Just like uncertainties gives more insights on the prediction of a neural network, explanations can also give another point of view to ensure these models are more reliable. More specifically, we will focus on counterfactual visual explanations, summarizing the state-of-the-art and also showing the limits of the current approaches.

Chapter 5.

Interpretability and explainability in deep learning

5.1. Introduction to explainable AI

While complex artificial intelligence models like deep neural networks are becoming more and more powerful for a wide range of tasks, their ability to explain their decisions are inversely becoming more and more difficult to obtain. *Explainable AI* (or XAI) aims at deconstructing such black-box models, by trying to generate an explanation as to *why* a machine learning model made a certain decision.

Even though accuracy and performance is still a characteristic that is sought in many applied fields where precision is very important, it is becoming clear that there is a need for AI systems to be able to explain their predictions, especially in applications where livelihoods are in the balance. XAI enables a form of trust and transparency between the system and the end user, and it allows to identify biases in the model that can hurt principles such as fairness and accountability. Such needs are already being codified in law, such as the European's Union General Data Protection Regulation (GDPR)[1], which states a right to obtain an explanation by the form of information intelligible by the end user.

In the case of a transverse research application such as in astrophysics, being able to retrieve meaningful explanations from a model is also very important to ensure that the measurements predicted by the deep neural network are made according to the right factors, without any unwanted bias.

More generally, *explainable* AI defines an AI system that produces *explanations* which can be interpreted by an end user. Such explanation can come in many forms, but in general it comes down to being able to present an intelligible form of the relation between input and output, like a causal relationship. *Interpretable* AI is essentially used in an interchangeable manner with explainable AI in the literature, and as such we will refer to such systems as explainable AI for the rest of this thesis.

In the following section, we will describe the different forms of explanation methods in XAI, as described in the survey from Islam et al. (2021). Explainability methods can be separated in three types:

- *Model-intrinsic* methods are based on machine learning models that are explainable

[1] https://gdpr.eu/

by design. Oftentimes, it comes down to models with fewer parameters or tractable computations. There is however a trade-off, as intrinsic interpretability can come with a cost in overall performance.

- *Model-agnostic* methods aim to generate explanations from any black-box model, using an explanation generation algorithm that is independent from the model itself. This allows model-agnostic methods to be very flexible and easy to implement. They also come in varied forms which can allow for a wide variety of explanations and visualizations on a single model, but they may lack certain details of more complex models.
- *Example-based* methods are a special case of explainability methods which can either be model-intrinsic or model-agnostic. They aim to generate explanations in the form of an analogy. For instance, an explanation of the form "$Y \Rightarrow Z$ and X is similar to Y so the model will predict that $X \Rightarrow Z$" is an example-based explanation.

5.1.1. Model-intrinsic methods

Intrinsically explainable methods rely on machine learning models that can be fully explained by their design. Most often, it means that a tractable form of the relationship between input and output can be retrieved, and that such relationship can be easily interpreted by an end user. Model-intrinsic methods include Generalized Linear Models (GLMs) and Rule based models.

Generalized Linear Models. Generalized Linear Models (Nelder and Wedderburn, 1972) are a family of models which generalizes linear regression models. GLMs, in essence, model the relationship between the input $X \in \mathbb{R}^n$ and the output $Y \in \mathbb{R}^m$ by parameters $\beta \in \mathbb{R}^n$ and an invertible link function g, such that

$$\mathbb{E}(Y \mid X) = \mu = g^{-1}(X\beta) \tag{5.1}$$

where μ is the mean of the distribution of Y conditioned on X. Depending on the definition of the link function and support of the distribution, GLMs can be used to model multiple types of relationships.

A simple GLM is a linear regression model. In this case, the link function $g : \mathbb{R}^n \to \mathbb{R}$ is simply the identity, such that

$$\mathbb{E}(Y \mid X) = \mu = X\beta\,. \tag{5.2}$$

In the case of a linear regression model, each coefficient of β quantifies the contribution of the corresponding coefficient in X to the mean μ. By analyzing the weights of the linear combination, if the model has learned the dataset well, it is easy to check which input features contribute the most to a certain prediction, which can form an explanation. However linear regression models are quickly limited as soon as the relationship between μ and X becomes non-linear.

Logistic regression is another type of GLM where the support of the distribution is discrete, i.e. the logistic regressor is a classifier. In the example of a binary classification problem, the logistic regression model is defined by a Bernoulli distribution and the link function

$$g(\mu) := \ln\left(\frac{\mu}{1-\mu}\right) \tag{5.3}$$

such that

$$\mathbb{E}(Y \mid X) = \mu = \sigma\left(X\beta\right) \tag{5.4}$$

with $\sigma(x) = \frac{1}{1+e^{-x}}$ the sigmoid function (hence the name since it is the CDF of a logistic distribution). Note that since the considered distribution is the Bernoulli distribution, we have $\mathbb{E}(Y \mid X) = \mathbb{P}\left(Y = 1 \mid X, \beta\right)$.

The goal of the logistic regression is to have $\mathbb{P}\left(Y = y \mid X, \beta\right) > 0.5$, $\forall (X, y) \in \mathcal{D}$. This model can also be generalized to a multiclass classification task. Like the linear regression model, the explanations of the model are retrieved by analyzing the coefficients of the model, meaning that any coefficient in β defines the importance of the corresponding input feature in the decision process. Logistic regression models are, in a sense, similar to the perceptron in terms of a classification model (see Section 2.1.1). However, these models are still limited once the dataset is no longer linearly separable.

Rule-based models. Rule-based models are a type of intrinsically explainable models that aim to separate the instances in the dataset by a set of rules which depends on the input features. Decision trees are one example of such rule based models which are widely used. The idea behind a decision tree is to take each feature of the input $X = (x_i)_{i=1}^n$ and learn scalar boundaries $(h_i)_{i=1}^n$ for each of them in a hierarchical manner, depending on the corresponding label $y \in \{y_j\}_{j=1}^C$. The first feature to be considered is the root node, x_i where $1 \leq i \leq n$. It is compared to some threshold h_i and then depending on the result, another feature x_j with $j \neq i$ whose parent is x_i in the tree is compared to h_j, and so on. In the end, the decision tree creates a path from the root node to one of the leaf nodes, which defines a specific label y_k with $1 \leq k \leq C$ and $C \geq 0$ is the number of classes. This path is exactly the explanation of the decision made by the tree. See Fig. 5.1 for an example of a decision tree.

Still, decision trees do have some disadvantages. Their non-linearity makes them very sensible to slight changes in the input features. As such they lack the ability to give explanations in a more gradual manner, like linear regression models. Also, deep decision trees with lots of nodes can easily overfit and output explanations that cannot be generalized and that can quickly become abstruse.

More recently however, several rule-based classifiers were developed (Dash and Goncalves, 2021; Yu et al., 2021; Ghosh et al., 2022), which generates rules on complex classification problems while avoiding rules that are deemed too complex to be interpretable.

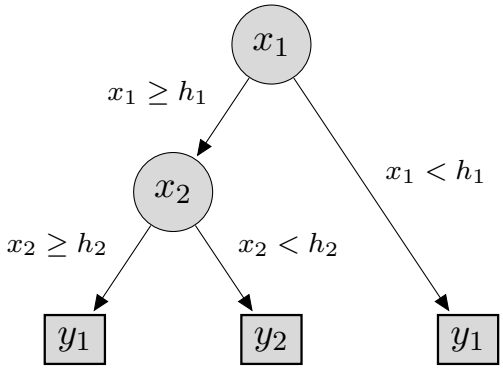


Figure 5.1.: Graphical representation of a decision tree with two features and two classes. An explanation for the class y_2 would be ($x_1 \geq h_1$ and $x_2 < h_2$), while an explanation for the class y_1 would be (($x_1 \geq h_1$ and $x_2 \geq h_2$) or ($x_1 < h_1$)).

5.1.2. Model-agnostic methods

We have seen the main models that can generate intrinsic explanations. However, more complex datasets and problems require more powerful machine learning models, such as deep neural networks. And with this added complexity we loose the ability to generate meaningful intrinsic explanations.

Model-agnostic methods came from the need to be able to explain any machine learning model (hence another name for model-agnostic methods are "black-box methods", since they do not assume any prior information on the current model being explained). Model-agnostic methods are usually based on two types of techniques. Those who are based on surrogate models try to approximate the black-box model by another one, either globally or locally. Another approach to model-agnostic methods is feature importance, by trying to measure how each input feature x_i influences the output, again either locally or globally. We will now describe some of the main techniques for each approach.

Feature importance methods. Among the feature importance methods for model-agnostic explanation are Partial Dependence Plot (PDP), Individual Conditional Expectation (ICE), Accumulated Local Effects (ALE) and SHAP.

PDP (Friedman, 2001) is a global method which shows the marginal effect of one or two features in the prediction of the model. For instance, in a classification model, the PDP is the probability of a certain class with respect to one or two input features, and averaging the remaining features over the dataset. This method however becomes less and less effective to obtain meaningful explanations if the number of features becomes very high (as showing thousands of plots or more becomes inefficient), or if they are heavily correlated.

ICE (Goldstein et al., 2015) gives an alternative to PDP by giving the marginal effects of each feature on the prediction on a single instance of the dataset, instead of averaging over all of them in the case of PDP. A corollary of this definition is that the average of the ICE regressions is PDP. However, by its design, ICE suffers from similar issues

encountered in PDP: it cannot give meaningful explanations if the input features are correlated, and in addition, it also becomes inefficient if the dataset is large.

ALE plots (Apley and Zhu, 2016) also aim to analyze the effects of one (or two) particular feature. However, instead of averaging their predictions (as PDP does), ALE averages their *change* in prediction by averaging local behaviors. More precisely, in the case of one feature, let a model f with input $x = (x_s, x_c)$, x_s being the analyzed input feature and x_c the remaining features. Finally, $\mathcal{X} = \{x^{(i)}\}_{i=1}^{n}$ is the set of instances in the dataset. ALE partitions the range of possible x_s values in K_s intervals: $\forall k \in [\![1, K]\!]$, $N_s(k) = [z_{k-1}, z_k]$. Then, we note the number of instances in each interval $(n_s(k))_{k=1}^{N}$. The ALE function that defines the plot is defined as

$$\hat{f}_{s,ALE}(x) := \hat{g}_{s,ALE}(x) - \frac{1}{n}\sum_{i=1}^{n} \hat{g}_{s,ALE}(x_s^{(i)})$$
$$\text{with } \hat{g}_{s,ALE}(x) := \sum_{k=1}^{K_s} \frac{1}{n_s(k)} \sum_{i, x_s^{(i)} \in N_s(k)} \left[f(z_{k,s}, x_{i,c}) - f(z_{k-1,s}, x_c^{(i)}) \right] . \tag{5.5}$$

Basically, ALE averages on each interval the average change in prediction with respect to x_s, and then the function $\hat{g}_{s,ALE}(x)$ is centered to result in $\hat{f}_{s,ALE}(x)$, therefore it is unbiased. ALE plots can be extended in 2D by partitioning the 2D space in many rectangles.

Because they are unbiased, ALE plots are still valid even when features are correlated. The plots generated are also clearly interpretable, giving meaningful information on the effect of a feature conditioned on others. However they are complex to implement, and even though they are unbiased, they come to a limitation as soon as the features become strongly correlated.

Finally, SHAP (Lundberg and Lee, 2017) is a local feature-importance model-agnostic explainable algorithm based on Shapley values (Shapley, 1953). Shapley values are a solution to a game theory problem such that each value represents the average contribution of a player to a given score function of all players. As an explanation algorithm, SHAP approximates the model prediction $f(x)$ of some instance $x = (x_i)_{i=1}^{n}$ by a linear regression model with coefficients $(\phi_i)_{i=0}^{M}$. Each coefficient represents the Shapley value of some feature of x we want to see the contribution of, and ϕ_0 is set to the mean value of the linear regression model. Note that $M \leq n$, such that the Shapley values are defined only for a subset of features of x. The missing features of the explanation are averaged over the dataset, assuming all features are independent. Note that the explanations are contrastive as they are relative to the mean over the dataset, and cannot be considered as absolute as in a linear regression model.

Surrogate models. Surrogate models aim to explain the model $f(x)$ by approximating it by another, simpler model $g(x)$. Trepan (Craven and Shavlik, 1995) is one example of a global surrogate model, which approximates the model by a decision tree using queries to sample relationships between the inputs x and their predictions y to learn the

structure of the tree. However this method can be difficult to scale as neural networks have become very complex over the years.

One of the most well known surrogate model approach however is LIME (Local Interpretable Model agnostic Explanations) (Ribeiro et al., 2016). LIME uses a local approximation of the model to generate explanations. More precisely, given an input x, LIME will sample n points neighboring x: $(x_i)_{i=1}^n$, and query their predictions by the model f. Then, weighting each sample by its distance to x, LIME will learn a linear model that approximates the model around x (see Fig. 5.2). However, LIME can be very sensitive depending on the starting point and the complexity of the model. Since the approximating model is both local and linear, it is very well possible to find a completely different approximation for another neighboring instance.

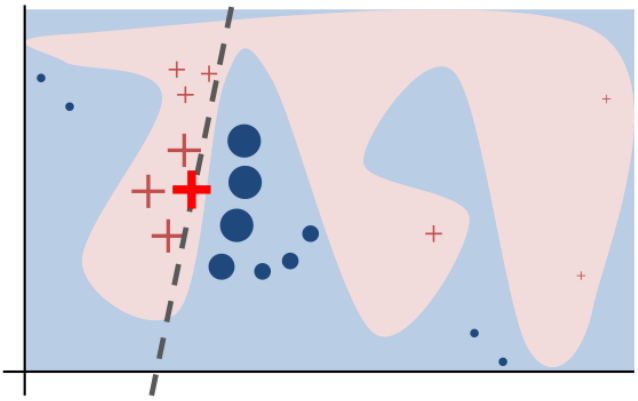


Figure 5.2.: Illustration of LIME. LIME aims to locally approximate the black-box model (blue/pink background) in the neighborhood of a certain point x (bold red cross) by sampling instances in that neighborhood (other crosses and circles). Then, LIME weighs each instance relative to their distance to x (represented by their size). This allows LIME to learn a linear model which gives local explanations, but cannot be generalized to the whole dataset. Credits: Ribeiro et al. (2016).

5.1.3. Example-based methods

Finally, example-based methods offer an alternative point of view by generating an explanation in the form of an example similar to a certain input, that can explain the prediction. Among example-based methods are the ones using prototypes, adversarial examples, and counterfactual explanations, the latter which we will go into more detail in Section 5.2.

Prototypes. Prototypes approaches aim at selecting or generating a set of instances $(p_i)_{i=1}^m$ which describes the data distribution well. For instance, one can create a set of prototypes by averaging the instances over each class, or over certain features. Then, when trying to explain the prediction $y = f(x)$ of a certain instance x, one would look at the prototype p_i with the same predicted label y which is the most similar (by some

definition) to x, and the explanation would be that prototype. One could also look at the prototypes that are not similar to x to give a contrastive explanation, i.e. the model predicted y for x because x is dissimilar to some prototype p_j for which the prediction is $y' \neq y$.

However prototypes can be difficult to implement, as it requires to answer many questions that are very dependent on the dataset and model considered: how many prototypes ? How do we choose or construct them ? How can we be sure these prototypes explain the distribution very well ? Like many explanation methods, prototypes might be best suited in conjunction with other types of explanation techniques. Examples of prototypes-based methods for explainable AI include the K-medoids algorithm (Kaufman and Rousseeuw, 1990) or, more recently, MMD-critic (Kim et al., 2016).

Adversarial examples. *Adversarial examples* (Szegedy et al., 2014) are examples x_{adv} that "fools" the network into predicting a wrong label y', given that the true label of x_{adv} is $y \neq y'$. Complex machine learning models with lots of parameters, like neural networks, are especially susceptible to these adversarial examples.

In a sense, adversarial examples aim to explain the model by showing how they fail to generalize the underlying data distribution $p(x, y)$. They can also be used as a debugging tool to pinpoint certain modes of failure of the model.

Goodfellow et al. (2015) proposed a simple way to generate adversarial examples for a model f_ω, by perturbing the current instance x by the sign of the gradient of the loss function with respect to x, a method called FGSM (Fast Gradient Sign Method)

$$x_{adv} := x + \varepsilon \operatorname{sign}\left(\nabla_x \mathcal{L}(f_\omega(x), y)\right) , \tag{5.6}$$

with $\varepsilon \in \mathbb{R}^+$.

Intuitively, the instance x is perturbed in the direction given by the gradient which maximizes the loss, therefore reducing the accuracy of the model on that exact example. ε represents the step size in the adversarial example generation. An example of adversarial example generation using FGSM is shown on Fig. 5.3.

Goodfellow et al. (2015) also proposed a way to make neural networks more robust to these adversarial examples, with adversarial training. The adversarial training objective is described as

$$\mathcal{L}_{adv}(f_\omega(x), y) := \alpha \mathcal{L}(f_\omega(x), y) + (1 - \alpha)\mathcal{L}\left(f_\omega(x + \varepsilon \operatorname{sign}\left(\nabla_x \mathcal{L}(f_\omega(x), y)\right)), y\right) , \tag{5.7}$$

with $\alpha \in [0, 1]$ and $\varepsilon \in \mathbb{R}^+$.

This adversarial loss function adds an additional term weighted by $1 - \alpha$ that serves as a regularization term to the adversarial examples generated by FGSM to make sure the model classifies them correctly.

Adversarial examples are still an ongoing issue with neural networks, making them vulnerable to unwanted behavior which can cause serious problems with regards to trust, reliability and security. They are also heavily correlated to the notion of counterfactual examples, which we will describe in the following section, as they will be the focus of the rest of this chapter, and the next one.

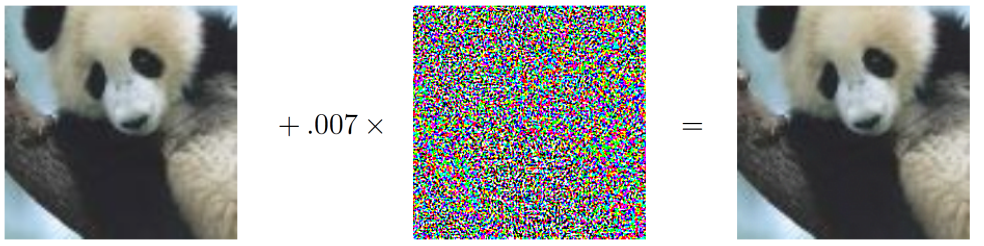


Figure 5.3.: FGSM adversarial example. The panda image on the left is correctly classified by the model, with an average confidence. Once the image has been perturbed by the gradient noise, the model is now confident it represents a gibbon, even though it clearly does not look like one. Credits: Goodfellow et al. (2015).

5.2. Counterfactual explanations

Counterfactual explanations (Wachter et al., 2018; Keane et al., 2021) are an example-based explanation method in the form of "If x' happened instead of x, then y' would have occurred instead of y". More precisely, counterfactual explanations aim to generate examples by analyzing the *minimal actionable changes* in the input features such that the modified instance x' has a different label $y' \neq y$ under the data distribution $p(x, y)$. x' is then called a *counterfactual example* of x. The label y' is, in most counterfactual explanation algorithms, given as an input to guide the explanation process towards that particular class. In that case, y' is called the *target class* of the counterfactual example.

Counterfactual explanations aim for *minimal* changes in the sense that the counterfactual example x' must be close to x by some definition (for instance, a distance metric). They must also be *actionable* in the sense that one could reasonably realize the changes to go from x to x'. A corollary of actionability is *realism* (or *plausibility*: a counterfactual example (x', y') must be likely under the data distribution $p(x, y)$.

Take this counterfactual explanation of a binary classification problem as an example: "You were denied your graduation because your grade in mathematics was 8/20. If your grade was 10/20, you would have graduated.". Let us assume that $x = (x_{math}, x_{physics}, x_{major}) = (8, 15, \text{math})$ is the current instance with its set of features: the grade in maths, the grade in physics, and the current major of the student.

The counterfactual example here is the instance x' where the feature corresponding to the mathematics grade has been changed from 8 to 10. This explanation is not only minimal (since the minimum grade to pass is effectively 10), but also actionable (it is possible to achieve a grade of 10), and realistic. A non minimal counterfactual example could have been one where the math grade is changed to 18, and the physics grade to 19, even though the latter was already originally 15. A non actionable counterfactual example would have been one where x_{major} has been changed to English, since it is

not feasible for the student to change their major of studies. Finally, a non realistic counterfactual example would have been one with $x'_{math} = 22$, since a grade cannot exceed 20.

5.2.1. Searching for counterfactual explanations

Several approaches to find counterfactual explanations have been proposed in the literature. The most prevalent type of algorithm to search for counterfactual example is the one that is based on solving an optimization problem.

Given a model f_ω parameterized by $\omega \in \Omega$, an instance $x \in \mathbb{R}^n$, and a target label $y' \in \mathbb{R}^m$, Wachter et al. (2018) proposed the following objective function to search for counterfactuals

$$x_{CF} := \underset{x' \in \mathcal{X}}{\operatorname{argmin}} \mathcal{L}(f_\omega(x'), y') + \lambda\, d(x, x')\,, \tag{5.8}$$

with $\lambda \geq 0$ and $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ a distance defined on $\mathbb{R}^n$. The distance term ensures the minimality property of counterfactuals, by penalizing counterfactual examples that are too far from the original instance x.

Usually, d is defined as the L_2 or L_1 distance, but Wachter et al. (2018) used the following distance for their counterfactual search

$$d(x, x') := \sum_{i=1}^{n} \frac{|x_i - x'_i|}{\text{MAD}_i} \tag{5.9}$$

$$\text{with} \quad \text{MAD}_i := \text{median}_{x \in \mathbf{X}}(|x_i - \text{median}_{x' \in \mathbf{X}}(x_i)|)\,, \tag{5.10}$$

with $\mathbf{X} := \{x^{(i)}\}_{i=1}^{N}$ the set of all instances of the dataset.

This distance represents the L_1 norm normalized by the inverse median absolute deviation. They argue that this particular metric better captures the volatility of the input space, and that the use of the median makes it more robust to outliers. Also, they argue the use of a L_1 related distance for its sparsity-inducing properties, something that is valuable in counterfactual search.

From there, the explanation is generated through a usual gradient descent algorithm, such as Adam (Kingma and Ba, 2015).

However, this first approach to counterfactual explanations does not necessarily ensure actionability or realism. In fact, this optimization search might be susceptible to generate counterfactual examples that are closer to adversarial examples; more detail on that particular fact in Section 5.2.2.

To try to solve this issue, Dhurandhar et al. (2018) added a term to the optimization objective to account for plausibility

$$x_{CF} := \underset{x' \in \mathcal{X}}{\operatorname{argmin}}\, c\,\mathcal{L}(x') + \beta\,||x - x'||_1 + ||x - x'||_2^2 + \gamma\,||x' - \text{AE}(x')||_2^2\,, \tag{5.11}$$

where $c, \gamma, \beta \geq 0$, and AE is an autoencoder, such that $||x' - \text{AE}(x')||_2^2$ is a reconstruction error evaluated on that autoencoder. This added term encourages the search algorithm

to find a counterfactual x' that lies in the data distribution, given the assumption that the reconstruction term would be higher for outliers, which is not necessarily true. The authors also use a specific type of loss function, which compares the element-wise discrepancy on the output vector of the original class coefficient relative to the others

$$\mathcal{L}(x') := \max\{[f_\omega(x')_{i^*} - \max_{i \neq i^*} f_\omega(x')_i, -\kappa\}\,, \tag{5.12}$$

$$i^* := \arg\max_i f_\omega(x)\,, \tag{5.13}$$

where $\kappa \geq 0$ is a parameter that controls the separation between $f_\omega(x')_{i^*}$ and $\max_{i \neq i^*} f_\omega(x')_i$. This way, the target class is not given but is defined as the second most confident class in the original input.

Looveren and Klaise (2021), on the other hand, proposed an approach based on prototypes to generate their counterfactual examples. They expand on Dhurandhar et al. (2018) objective function by adding a regularization term related to the distance to some prototype

$$x_{CF} := \underset{x' \in \mathcal{X}}{\operatorname{argmin}}\, c\,\mathcal{L}(f_\omega(x'), y') + \beta\,||x - x'||_1 + ||x - x'||_2^2 + \gamma\,||x' - \mathrm{AE}(x')||_2^2 + \theta\,\mathcal{L}_{proto}(x')\,, \tag{5.14}$$

with $c, \beta, \gamma, \theta \geq 0$. The prototype loss term, $\mathcal{L}_{proto}$, is defined as follows. Given the initial instance x, the prototype of a certain class y is

$$\mathrm{proto}_y := \frac{1}{K}\sum_{i=1}^{K} \mathrm{ENC}(x_k^y)\,, \tag{5.15}$$

where $(x_k^y)_{k=1}^K$ are the K nearest neighbors of x with label y in the latent space of the autoencoder, and ENC is the encoder of the autoencoder. For this counterfactual explanation method, the target class is not given beforehand but is defined as the label of the nearest prototype:

$$y' := \underset{y' \neq y}{\operatorname{argmin}}\, ||\mathrm{ENC}(x) - \mathrm{proto}_{y'}||_2\,. \tag{5.16}$$

Finally, $\mathcal{L}_{proto}(x')$ is defined as:

$$\mathcal{L}_{proto}(x') := ||\mathrm{ENC}(x') - \mathrm{proto}_{y'}||_2^2\,. \tag{5.17}$$

Adding this term would incentivize the algorithm to generate plausible counterfactual examples as the prototype would serve as a reference, possibly preventing the counterfactual example to end up in a non realistic or actionable state. However, this is still an assumption and it does not ensure these properties, but rather only adds a heuristic towards them to the optimization objective.

While Dhurandhar et al. (2018) and Looveren and Klaise (2021) used an autoencoder to regularize their counterfactual example, Joshi et al. (2019) on the other hand used a generative model $\mathcal{G}_\theta(z)$ to directly search a counterfactual example in the latent space $\mathcal{Z}$. They assume that their generative model is able to encode the instance space $\mathcal{X}$ into the latent space, i.e. a function $\mathcal{F}_\psi : \mathcal{X} \to \mathcal{Z}$ parameterized by ψ. Examples of generative models with an ability to encode instances are VAEs (Kingma and Welling, 2014) or even novel GAN architectures, such as VAE-GAN (Larsen et al., 2016), AttGAN (He et al., 2017) or BiGAN (Donahue et al., 2017).

Their method, called REVISE, uses the following objective function

$$x_{CF} := \mathcal{G}_\theta(z_{CF})\,, \tag{5.18}$$

$$z_{CF} := \underset{z \in \mathcal{Z}}{\operatorname{argmin}}\, \mathcal{L}(f_\omega(\mathcal{G}_\theta(z)), y') + \lambda\, d(x, \mathcal{G}_\theta(z))\,, \tag{5.19}$$

with $\lambda \geq 0$ and $z = \mathcal{F}_\psi(x)$. By using a latent space, the search space is then restricted to a constrained latent space which has been learned by the generative model to represent the data distribution more faithfully. Therefore, the resulting counterfactuals will be more realistic as an unconstrained data manifold can yield explanations that are unlikely or unrealistic. See Section 5.3.1 for experimental results supporting this fact. More precisely, the second term in Equation B.15 constrains the reconstruction of the counterfactual encoding to not be too far from the original image. As such, the counterfactual encoding z_{CF} is encouraged to be the closest to z in the latent space, relative to the image space, making the constraint more semantic.

Finally, Schut et al. (2021) proposed a Bayesian approach to generate counterfactual examples. Their following objective function is

$$x_{CF} := \underset{x' \in \mathcal{X}}{\operatorname{argmin}}\, \mathcal{L}(f(x'), y') \tag{5.20}$$

with $\mathcal{L}$ the cross-entropy loss function and f is some classifier. Since minimizing the cross-entropy also minimizes the predictive entropy, this objective implicitly minimizes both the aleatoric and epistemic uncertainties. Their idea is that realism is linked to low epistemic uncertainty, and having a low aleatoric uncertainty should ensure a counterfactual instance that is unambiguous.

More precisely, they consider an ensemble of models $\{f_m\}_{m=1}^M$ to take uncertainties into account, such that their empirical objective function is

$$\mathcal{L}_{Schut}(x') := \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}(f_m(x'), y')\,. \tag{5.21}$$

Notice that they do not use a measure of proximity to the original instance x. Instead, they expand on a technique called Jacobian-based Saliency Map Attack (JSMA) (Papernot et al., 2016), which considers the feature with the highest gradient

$$i' := \underset{1 \leq i \leq n}{\operatorname{argmax}} \left|\nabla_{x'} \mathcal{L}_{Schut}(x', y')_i\right|\,. \tag{5.22}$$

Then, at each step, the counterfactual x' is updated feature-wise with respect to that feature with highest gradient:

$$x'_{i'} \leftarrow x'_{i'} - \delta \operatorname{sign}\left(\nabla_{x'} \mathcal{L}_{Schut}(x', y')\right)_{i'} \tag{5.23}$$

with $\delta > 0$ the step size. Because the changes are feature-wise, they are sparse and is a way to ensure the minimality property of counterfactual examples.

This feature-wise approach however might come at the cost of computation time, as element-wise changes to the instance can make the convergence to the target class longer, let alone having to compute the gradient each time to update one feature.

This list of counterfactual explanation algorithms is by no mean exhaustive, as many new methods have been developed during the last few years. Optimization-based algorithms are not the only way to find a counterfactual explanation; as an example Keane and Smyth (2020) proposed an instance-based method which searches counterfactual examples by choosing instances that are already in the dataset. We will however still focus in the context of optimization-based methods, as they have been the most developed in the literature, are easy to implement and fast to compute in the context of deep neural networks.

5.2.2. Adversarial examples or counterfactual explanations ?

Before going forward, let us address the problem of distinguishing adversarial examples from counterfactual explanations. In the case of an optimization-based method, both approaches end up generating an example according to a loss function which aims to make the model f predict the new instance x' to another label y'. Even Wachter et al. (2018) approach can generate adversarial examples, which breaks the property of realism for counterfactual examples. Dhurandhar et al. (2018) added a regularization term involving an autoencoder to prevent that, and Joshi et al. (2019) used a generative model to search the counterfactual example in a latent space, but there is still no way to ensure adversarial examples won't be generated: an autoencoder can very well reproduce data that is unlike the original dataset, and the same can be said for any generative model. This can also cause a problem for conceiving a metric for realism. Looveren and Klaise (2021) proposed two metrics IM1 and IM2 based on the reconstruction losses autoencoders

$$\text{IM1}(x', y, y') := \frac{||x' - \text{AE}_{y'}(x')||_2^2}{||x' - \text{AE}_y(x')||_2^2 + \varepsilon} \tag{5.24}$$

$$\text{IM2}(x', y') := \frac{||\text{AE}(x') - \text{AE}_{y'}(x')||_2^2}{||x'||_1 + \varepsilon} \tag{5.25}$$

with $\varepsilon > 0$, $\text{AE}_y, \text{AE}_{y'}$ two autoencoders trained only on instances from class y and y' respectively and AE an autoencoder trained on all classes. Because those metrics use autoencoders, they can cause issues where outlier data might have low IM1 or IM2 metrics, especially for more complex data like images.

Schut et al. (2021) argue that realism is linked to epistemic uncertainty, as an example with low epistemic uncertainty is recognized by the model and therefore not an outlier. However it has been shown that epistemic uncertainty can be low even in parts of the instance space that are beyond the regions of high density of $p(x)$ (Smith and Gal, 2018), i.e. outlier data with low epistemic uncertainty. This has also been observed empirically, as we will see in Section 5.3.

Dhurandhar et al. (2018) on the other hand, chose to evaluate their method by asking experts to manually validate their generated counterfactual examples. While this approach is not easily scalable, it still is the most reliable one. In the same spirit, Freiesleben (2022) differenciates counterfactual examples from adversarial ones by asking an oracle if the counterfactual example x_{CF} is indeed of its intended label y', while an adversarial example wouldn't. This is the same as asking an expert to validate the counterfactuals.

However, one fact that has been noted is that adversarial training has been shown to help generate more realistic counterfactual examples (Schut et al., 2021; Boreiko et al., 2022), which makes sense: ensuring that the model does not recognize adversarial examples help counterfactual search algorithms to not end up generating an adversarial instance.

In conclusion, the threshold between counterfactual search and adversarial search is still quite blur and not properly defined, especially when dealing with data where realism becomes more and more difficult to quantify reliably, like images. In the following section and chapter, we will focus on counterfactual generation for image data, and we will try to understand the underlying reasons why some optimization-based approaches fail to reliably generate realistic instances, and ideas on how to deal with this issue.

5.3. Counterfactual visual explanations

A more specific subset of counterfactual explanations are counterfactual *visual* explanations, i.e. counterfactual explanations applied to images. Images are notoriously more complex than tabular data in many ways, being on average high dimensional with both local and global spatial information. The semantic behind an image is a very high level type of information which can be very difficult to explain by using lower-level features such as pixels. Some classification tasks involving images separate their dataset between different labels that can be very different in terms of Euclidean or Manhattan distance in the image space, which can make the heuristic to find counterfactual explanations even more challenging.

Several attempts at generating counterfactual visual explanations with an optimization-based approach have been made in the past years (Liu et al., 2019; Looveren and Klaise, 2021; Boreiko et al., 2022; Schut et al., 2021; Joshi et al., 2019), and while they achieve convincing results on some instances, they are also sensible to create less realistic counterfactual examples, due to many factors.

In this section, we will try to analyze the underlying causes that prevent most of these methods to produce realistic counterfactual visual explanations, and gradually explain how to deal with these causes.

5.3.1. Generating counterfactuals from the image space

The first idea to generate counterfactual visual explanations is to directly work on the image space to create a counterfactual example. For this, we will consider two approaches:

- Wachter et al. (2018) approach by applying the gradient of the objective function on the entire image.
- Schut et al. (2021) algorithm by iteratively updating one pixel at a time, which has the highest gradient.

The dataset used is MNIST (LeCun et al., 2010), as it is a very common and simple image dataset to serve as a benchmark. The MNIST dataset consists of 28×28 pixels greyscale images representing handwritten digits with their labels corresponding from 0 to 9, i.e. $\mathcal{X} \subset \mathbb{R}^{28\times 28}$ and $\mathcal{Y} = [\![0, 9]\!]$.

We train an ensemble of $M = 50$ convolutional neural networks, with 3 convolutional layers with Batch Normalization (Ioffe and Szegedy, 2015) and PReLU activation (He et al., 2015). Batch normalization re-centers and re-scales inputs to make the training process more stable. Then, the last feature map is flattened and is followed with three hidden layers of 16 neurons with ReLU activation, and a final output layer of 10 neurons with softmax activation. Each CNN is trained for 50 epochs with a learning rate of 0.001 and batch size of 256, reaching an accuracy on the validation dataset of around 99%. Each CNN has been trained with adversarial training with $\alpha = 0.85$ (see Equation 5.7). The loss function used to train the neural networks is the categorical crossentropy

$$\mathcal{L}(f_\omega(X), y) := -\sum_{i=0}^{9} y_i \log f_\omega(X)_i \,. \tag{5.26}$$

For Wachter et al. (2018)'s approach, we choose d to be the L1 distance on $\mathbb{R}^{28\times 28}$, with $\lambda = 0.001$ (see Equation 5.8). We have noticed empirically that higher values of λ prevent the algorithm to converge and instead stay around the original image without reaching an explanation. We chose to let the gradient descent run for at most 500 steps or until the model predicts that $\mathbb{P}(Y = y'|X') \geq 0.99$, and using Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of $\eta = 0.005$. Given an image X with a label $y = 5$, we aim to find a counterfactual example X' such that $y' = 1$. The result can be seen in Fig. 5.4.

As we can see, without any form of regularization on the counterfactual objective function, the generated example does not resemble an image of the target class, even with adversarial training: it lacks realism. Given that the gradient is updated on the whole image, it can also very easily add noise into the image in spaces that are usually empty, which can cause unrealistic counterfactuals.

Schut et al. (2021)'s approach was also proposed with the idea to prevent too much unnecessary noise, and only update areas of the image that are relevant to the explanation. Therefore, we also apply their algorithm on the same picture with the same conditions, except that the distance term in the loss is removed. We also set the step size to $\delta = 0.2$, with a maximum of 5 updates per pixel. The generated counterfactual explanation is shown in Fig. 5.5.

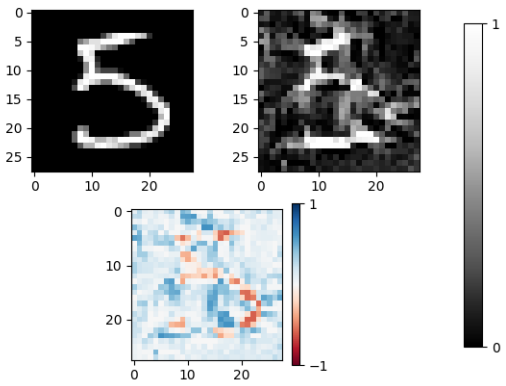


Figure 5.4.: MNIST counterfactual explanation: $5 \rightarrow 1$ from the image space using Wachter et al. (2018)'s approach. Top left: original image. Top right: Counterfactual explanation. Bottom: Difference between explanation and original image.

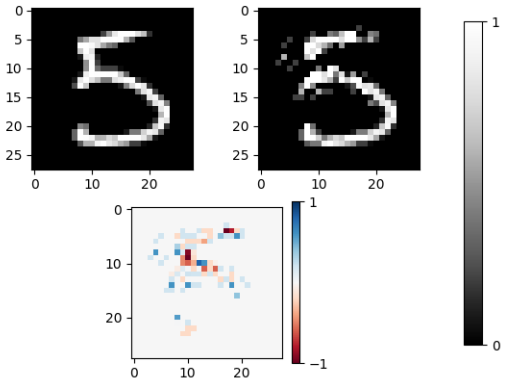


Figure 5.5.: MNIST counterfactual explanation: $5 \rightarrow 1$ from the image space using Schut et al. (2021)'s approach. Top left: original image. Top right: Counterfactual explanation. Bottom: Difference between explanation and original image.

As we can see, even though the changes are sparser than the previous example in Fig. 5.4, the end result is still far from representing the class $y' = 1$. In fact, the algorithm failed to converge, reaching 500 iterations with a final probability of $\mathbb{P}(Y = 1|X') \approx 10^{-5}$. There is also still some noise present on the image. In fact, the originals authors admitted that this approach was not suited for every couple (y, y') of initial and target classes. More examples showing how both of these image-space approaches to counterfactual explanations fail are shown later in Table 6.1.

5.3.2. Generating counterfactuals from a latent space

We now understand that trying to generate realistic counterfactual explanations in a highly dimensional space such as an image space, can be challenging. A fundamental issue is that we are trying to generate a realistic image representing a high-level concept by modifying low-level data.

We decide to implement Joshi et al. (2019)'s algorithm, by training and using a VAE to search for counterfactuals, according to Equation B.15. We still use an $L1$-norm for the regularization term with $\lambda = 0.001$. The VAE's encoder has the same convolutional architecture than the classifier, with a latent space of dimension 16. We more specifically use a β-VAE (Higgins et al., 2017) with $\beta = 0.001$, putting more importance on the reconstruction term to help generate counterfactuals that are more detailed and realistic. We let the VAE train for 50 epochs, batch size 256 and learning rate of 0.001.

We test REVISE using both just one classifier in a non-Bayesian setup, and with an ensemble of classifiers $\{f_m\}_{m=1}^{50}$. We call the ensemble variant "REVISE-ENSEMBLE" (or REVISE-E), and it uses the following objective

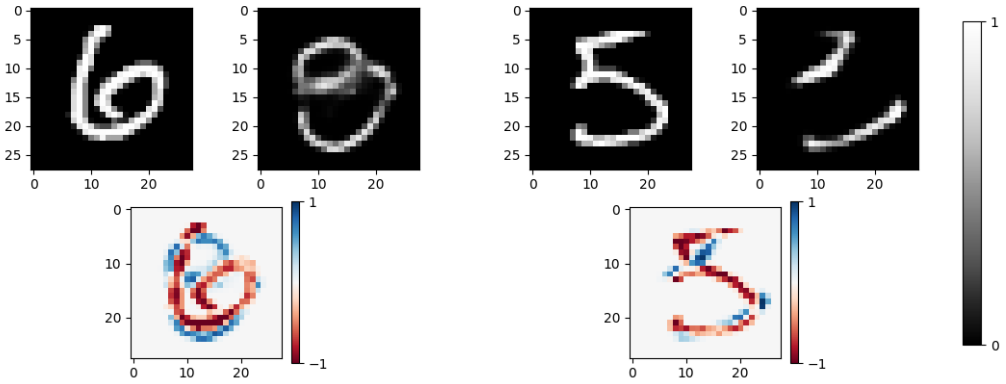
$$\begin{aligned} z_{CF} &:= \underset{z \in \mathcal{Z}}{\operatorname{argmin}}\, \mathcal{L}_{Revise-e}(z) \\ &:= \underset{z \in \mathcal{Z}}{\operatorname{argmin}}\, \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}(f_m(\mathcal{G}_\theta(z)), y') + \lambda\, d(X, \mathcal{G}_\theta(z))\,, \qquad (5.27) \\ \text{with} \quad & z \sim q_\phi(z \,|\, X) \quad \text{and} \quad x_{CF} = \mathcal{G}_\theta(z_{CF}) \qquad (5.28) \end{aligned}$$

where X is the input image, z the latent encoding sampled from the encoder given X, y' the target class, $\mathcal{G}_\theta$ the decoder parameterized by θ and q_ϕ the variational distribution that samples from the latent space, parameterized by ϕ.

In both cases, we set the same hyperparameters for the gradient descent algorithm: it stops whenever $\mathbb{P}(Y = y'|X') \geq 0.99$ or if it exceeds 500 iterations. The optimization algorithm used is Adam (Kingma and Ba, 2015) with an initial learning rate of $\eta = 0.05$.

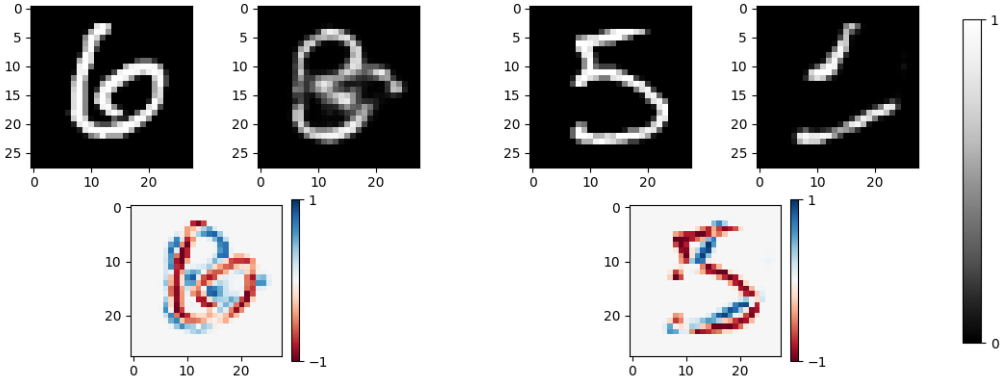
We show in Fig. 5.6 the results on the MNIST dataset for two explanations: $5 \to 1$ and $6 \to 3$. Each time, the algorithm converged to the target probability of 0.99 before reaching 500 iterations. We can see that the counterfactual images obtained by REVISE are generally better and less noisy, but there are still some cases where the counterfactual examples are not realistic. Fig. 5.6a is ambiguous and presents some artifacts, and Fig. 5.6b does not look like any number at all but rather an arbitrary symbol that fools the

classifier into believing it is a 1. As for REVISE-E, Figures 5.6c and 5.6d show the qualitative results of this modification. The improvements, if any, are marginal in the realism of these counterfactuals.



(a) 6 → 3 (REVISE)

(b) 5 → 1 (REVISE)



(c) 6 → 3 (REVISE-E)

(d) 5 → 1 (REVISE-E)

Figure 5.6.: Counterfactual explanations using REVISE (5.6a, 5.6b) and REVISE-E (5.6c, 5.6d). On each subfigure: top left: original image. Top right: Counterfactual explanation. Bottom: Difference between explanation and original image.

Projected gradient. In order to compare REVISE's approach to Wachter et al. (2018)'s (or any other method based on computing the gradient in the image space), we can study how the information contained in the image space gradient $\frac{\partial \mathcal{L}}{\partial X} = \left(\frac{\partial \mathcal{L}}{\partial x_i}\right)_{i=1}^{n}$ is filtered when using the gradient in the latent space $\frac{\partial \mathcal{L}}{\partial z} = \left(\frac{\partial \mathcal{L}}{\partial z_i}\right)_{i=1}^{k}$.

Using the chain rule, the latent space gradient depends on the image space gradient according to the following relationship

$$\frac{\partial \mathcal{L}}{\partial z} = \frac{\partial X}{\partial z}^T \frac{\partial \mathcal{L}}{\partial X}, \tag{5.29}$$

$$\frac{\partial \mathcal{L}}{\partial z} = \left(\left\langle \frac{\partial X}{\partial z_i}, \frac{\partial \mathcal{L}}{\partial X} \right\rangle \right)_{i=1}^{k}. \tag{5.30}$$

We want to "reproject" the latent gradient to the image space. We define

$$U_i := \left\| \frac{\partial X}{\partial z_i} \right\|_2^{-1} \frac{\partial X}{\partial z_i}. \tag{5.31}$$

U_i represents a normalized image, such that it represents the gradient of the generated image X according to the latent component z_i. Then, we can define

$$P(z) := \sum_{i=1}^{k} \left\langle U_i, \frac{\partial \mathcal{L}}{\partial X} \right\rangle U_i. \tag{5.32}$$

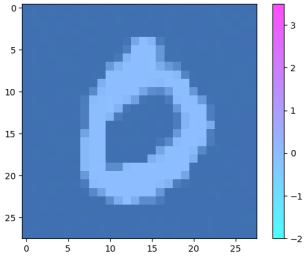
One way to interpret $P(z)$ is to consider it as a projection of the image space gradient $\frac{\partial \mathcal{L}}{\partial X}$ in the latent space, but represented in the image space. Finally, this projected gradient can be compared to the original gradient by computing the error between the two gradients

$$\text{Error}(z) := \frac{\partial \mathcal{L}}{\partial X} - P(z), \tag{5.33}$$

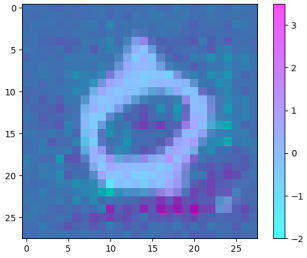
with $X = \mathcal{G}_\theta(z)$.

In Fig. 5.7, we show the gradient error $P(z)$ at different steps of the counterfactual example generation algorithm. At the start (Fig. 5.7a) and end (Fig. 5.7c), the difference between image and latent gradient is low, as the current point z (or X in the image space) is at a mode of the distribution, and the gradient is near 0 in both cases. At the half way point however is when the gradient becomes the steepest, and we can see that the latent gradient filters a lot of "gradient noise" around the image that are irrelevant to change the current number (see Fig. 5.7b). This kind of gradient noise is certainly the reason why counterfactuals generated by computing the gradient in image space fail to look more realistic and are closer to adversarial examples. Interestingly, we observed that the pixels that are filtered are the same during all of the gradient descent, indicating that the generative model that is the decoder not only acts as a denoiser on the image itself, but also denoises the gradient of a given classifier.

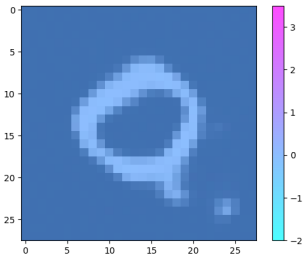
In conclusion, the challenge of generating realistic counterfactuals is hard, even when using a latent space to search for them. Indeed, a gradient descent algorithm on the latent space can remove the noise which is harmful to generate realistic images. However, it does not mean that the underlying structure of the latent space is used effectively. Moreover, assessing realism still remains a human dependent task and for which there is not a single metric that is reliable. In Chapter 6, we will present my recent contribution Theobald et al. (2022), that provides insights to better exploit the latent space structure to produce realistic visual counterfactual explanations that are unambiguous.



(a) Gradient error at the first step of the algorithm.



(b) Gradient error at the half way point.



(c) Gradient error at the final step.

Figure 5.7.: Visualization of the gradient filtering from the latent space, at different steps of the counterfactual generation using REVISE. The counterfactual explanation here is $0 \rightarrow 9$.

Chapter 6.

Clarity: counterfactual visual explanations with an ensemble of latent space classifiers

This chapter will focus on the contribution presented in Theobald et al. (2022). Following the previous chapter, we will expand on the notion of counterfactual visual explanations. We will propose insights on the underlying reasons of why current approaches fail to reliably generate realistic counterfactual images and how to exploit them to improve these counterfactual explanations.

6.1. The latent space as a basis for classification models

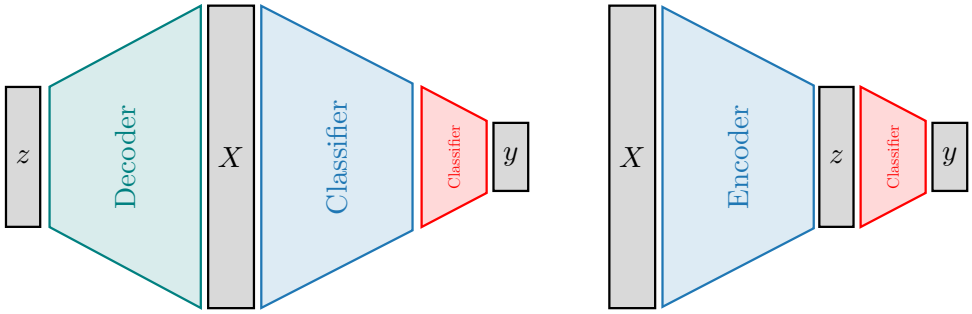
6.1.1. Defining a latent space classifier

We will now describe how using a classifier directly trained onto the latent space of a VAE leads to higher quality explanations. We will also give some insights and discuss the results.

Similarly to REVISE-E, we will use the latent space $\mathcal{Z}$ of some generative model as the search space. In this case, we will consider the generative model to be a VAE. The main difference is that the classifier will be trained in that predefined and learned latent space instead of the instance space $\mathcal{X}$. This means that instead of having a function defined as $f_\omega : \mathcal{X} \to \mathbb{R}^n$, we rather have $f_\omega : \mathcal{Z} \to \mathbb{R}^n$. By having a classifier directly trained in the latent space, we should be able to notice differences in how counterfactual explanations are generated in that space.

In Fig. 6.1, we show the main difference in architecture from REVISE-E to a latent space classifier. In REVISE-E, the classifier is already trained in the latent space, using all of the possible parameters to optimize the classification of the instances into the labels (blue and red blocks in Fig. 6.1a). Then, the counterfactual search is done in a latent space using a decoder to translate the latent embedding into an instance X (green block). Our approach is different in the sense that, while keeping the same architecture and numbers of parameters, the first part is actually the encoder of the VAE, and only the last layers after the latent embedding are trained for the classification task (see Fig. 6.1b).

Therefore, our new objective function is simply



(a) REVISE architecture (b) Latent space classifier

Figure 6.1.: Difference between between REVISE-E and our architecture and thus same parameterization. Blocks of same color share the same architecture. (a) In REVISE-E, the two "Classifier" blocks are trained for the classification task as a single model. (b) On the other hand, the latent space classifier (in red) is trained on top of an already learned encoder (in blue), which is frozen during the training of the classifier.

$$x_{CF} := \mathcal{G}_\theta(z_{CF})\,, \tag{6.1}$$

$$z_{CF} := \underset{z' \in \mathcal{Z}}{\operatorname{argmin}}\, \mathcal{L}(f_\omega(z'), y') + \lambda\, d(z, z'), \tag{6.2}$$

where $\lambda \geq 0$ and $d : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}^+$ is a distance defined in the latent space $\mathcal{Z}$.

Note that the distance d is now indeed defined in the latent space rather than in the image space. This distance represents a form of "semantic" separation between two instances. While we have not found much difference using a latent space distance rather than image space in the MNIST dataset, it yielded better qualitative results on the CelebA dataset (Liu et al., 2015). See Section 6.2 for a discussion on this aspect.

Another interesting aspect of using a latent space classifier is the computation of the gradient in the algorithm to find a counterfactual example. In both REVISE-E and our approach, the goal is to compute $\frac{\partial \mathcal{L}_{obj}}{\partial z}$, where $\mathcal{L}_{obj}$ is the current objective function for counterfactual explanations. However, while we can directly compute it from a latent space classifier, in REVISE-E we have to go all the way through the decoder

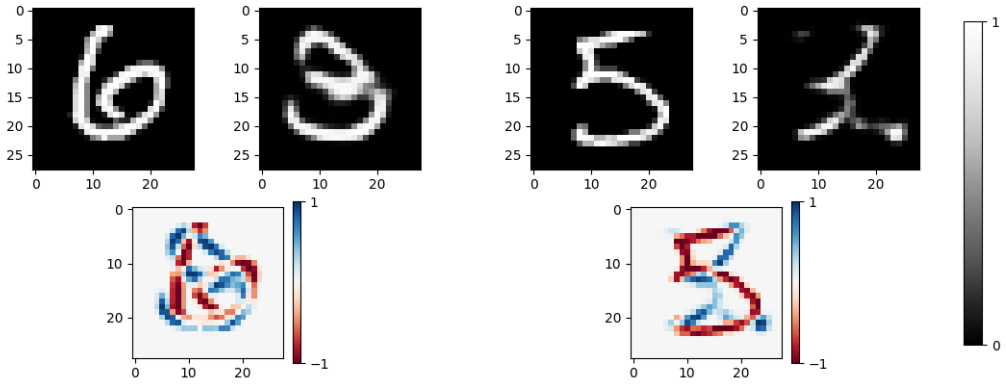
$$\frac{\partial \mathcal{L}_{obj}}{\partial z} = \frac{\partial X}{\partial z}^T \frac{\partial \mathcal{L}_{obj}}{\partial X}\,. \tag{6.3}$$

In practice we have observed a computational gain up to an order of magnitude faster by using our approach compared to REVISE-E.

Just like the classifiers defined and used in Section 5.3, we use adversarial training to train our latent space classifier. The explanations are generated using $\lambda = 0.001$ with d being the L_1 distance on $\mathcal{Z}$. We stop if the target probability satisfies $\mathbb{P}(Y = y' \mid X') \geq 0.99$ or if we reach 500 iterations.

In Fig. 6.2 we show qualitative results of this approach. As we can see, the counterfactual explanation $6 \to 3$ is very realistic as it presents no artifacts nor ambiguity. The

explanation even preserves features such as handwriting since the thickness of the stroke is the same in both the original image and the counterfactual explanation. However, the explanation $5 \to 1$ is not quite satisfactory, as it still presents artifacts and ambiguity. For instance, apart from the fact that the symbol does not look like a 1 exactly, some parts of the image are still blurry and not well defined, like in the bottom right or top left.



(a) $6 \to 3$ (b) $5 \to 1$

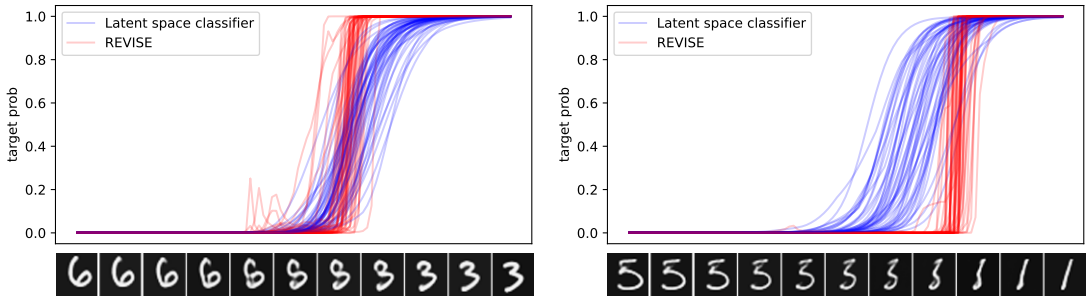
Figure 6.2.: Counterfactual explanations using a latent space classifier. Each subfigure shows original image, the counterfactual and the difference between counterfactual and original image. Color scale may vary.

6.1.2. The effects of using the latent space for the classification model

Given these previous results, one might ask how training a model directly on a VAE latent space can achieve more realistic explanations. We will now describe the underlying causes that might explain this observation.

In Fig. 6.3, we interpolate two images of two different classes into the latent space given the equation $z(t) = (1-t)\, z_1 + t\, z_2$, $t \in [0,1]$. Both z_1 and z_2 are latent representations from the encoder of the VAE of the two images X_1 and X_2, with distinct respective classes y_1 and y_2. We also plot the target probability $t \mapsto \mathbb{P}(Y = y_2 \,|\, z(t))$ for both the image and the latent space classifiers. For the image space classifier, we first decode $X(t) = \mathcal{G}_\theta(z(t))$ and then compute the predictive probability of $X(t)$. For each type of model, we trained 50 of them and plotted their curves in order to make sure the behavior observed is independent of the epistemic variance.

Notice how steep the curves for the image classifiers are when interpolating the latent space, in contrast to the classifiers directly trained in it. To understand the reason for this, let us recall that the density induced by the training data is extremely sparse when expressed in the image space: each class forms an "isolated island" of density. To go from one class to another, it is therefore necessary to cross spaces of very low density, corresponding to out-of-distribution non-realistic images. A classifier trained directly



(a) $6 \to 3$ (b) $5 \to 1$

Figure 6.3.: Probability of the target class with respect to the interpolation in the latent space. In red: REVISE. In blue: latent space classifier.

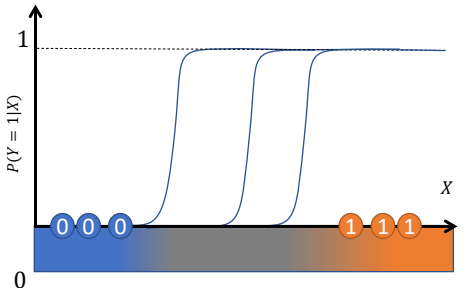
in the image space will therefore have the possibility to arbitrarily place separating boundaries between classes within this space, as illustrated on Fig. 6.4a.

Moreover, the cross-entropy will encourage transitions to be as steep as possible, unless there is a strong regularization on weights. Consequently, outside of these transition zones, the probability of the target class will be particularly stable, often equal to $0 + \varepsilon$ or $1 - \varepsilon$. This is exactly the "plateau effect" observable on the left and right-hand sides of the curves in Fig. 6.3.

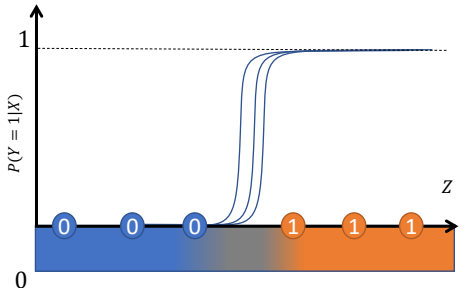
How do these facts translate into the VAE's latent space? During training, the VAE compresses the information contained in the density underlying the training data in such a way that the high density subspaces occupy a preponderant part of the latent space and, on the contrary, that the low density areas are mapped to very small volumes. The distribution of each class of images being represented by numerous examples (e.g., many characteristic images in MNIST) is thus mapped to significant volumes of the latent space, whereas the inter-class transition zones, corresponding to unrealistic images not observed during training, are reduced to very thin layers of the latent space, as illustrated on Fig. 6.4b. Experimental results shown on Figure 6.5 confirm this phenomenon: if we consider the interpolated segments of images either in the image or in the latent space, we see that the probability transitions of classifiers trained in the image space clearly appear more tightened in the latent space than in the image space, reflecting the compression effect on low density areas.

Since the probability transition areas are very small when viewed from the latent space, the smoothing of the probabilities by an ensemble of classifiers is very localized and thus rendered inefficient, which explains why REVISE-ENSEMBLE does not add much compared to REVISE.

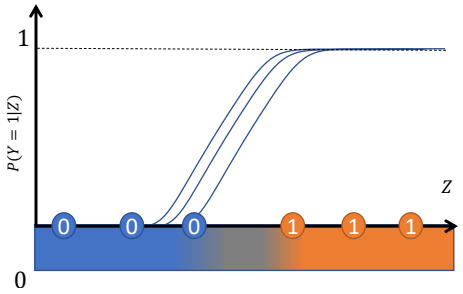
As a consequence, the VAE distorts the geometry in the latent space in such a way that unrealistic transition areas can appear very close to realistic image plateaus, and thus attracting the gradient in a wrong direction, all the more easily as probability transitions are steep. Once the transition is underway, the probability of the target class reaches the extreme value of 1 very quickly so that the gradient descent stops, without the corresponding counterfactual being realistic.



(a) Classifier trained and viewed from image space.

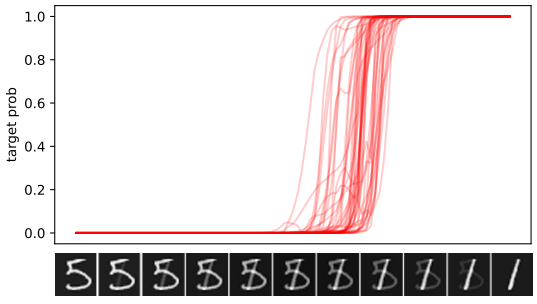


(b) Classifier trained in image space, viewed from latent space.

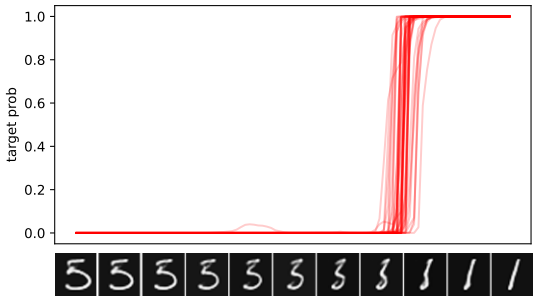


(c) Classifier trained and viewed from latent space.

Figure 6.4.: Illustration of the probability of the target class $Y = 1$ for three different classifiers (representing the epistemic uncertainty), in multiple settings. The 1D x-axis represents either the high dimensional image or latent space. Examples of classes 0 and 1 are represented as circles on the x-axis. The grey color in the x-axis gradient represents the area between the two classes with out-of-distribution, unrealistic images, which is compressed in the latent space.



(a) Image space interpolation



(b) Latent space interpolation

Figure 6.5.: Probabilities of the target class from the ensemble of image space classifiers, with respect to the interpolation in the image and latent space respectively, from 5 to 1.

On the other hand, by training the classifier directly in the latent space, the model is forced to be constrained by the lower dimension, which leads to smoother separations less prone to overfitting, as illustrated in Fig. 6.4c. We can see in Fig. 6.3 how the model trained in latent space is not confident as long as the example is ambiguous. This can explain why latent-space based classifiers are more interpretable, as the gradients lead more naturally to examples closer to the target class and not to the outliers.

In the case of REVISE we can also notice in Fig. 6.3a how the probability curve is not monotonous and has multiple local maxima. These instabilities again illustrate how the image space classifier is not well calibrated when observed from the latent space. They also entail the risk for REVISE-E to being stuck in local maxima when trying to generate an explanation. This has been noticed in a few cases, like in Fig. 6.12k where we ask REVISE-E to come up with a counterfactual explanation from a 3 to an 8. REVISE-E struggles for more than 200 steps to get out of a local maximum and fails by returning an unrealistic counterfactual. This is contrasted by the results of classifiers trained on the latent space, which not only yield smooth curves but monotonous as well.

However, we can notice in Figure 6.3 the variance between the different classifiers in the interpolated probability curves. We will explain in the next section how to get better explanations by leveraging an ensemble of classifiers to minimize the epistemic uncertainty, in addition to the latent space classifiers.

6.2. Clarity: counterfactual explanations with a Bayesian latent space classifier

We will now present *Clarity*, a type of classifier that subsumes every component we have seen so far to produce quality visual counterfactual explanations. The basis behind *Clarity* is to use a Bayesian classifier trained on a latent space of a VAE. In Section 6.1.2, we have seen how using a latent space classifier helps to produce classifiers that are less overconfident on ambiguous examples, due to the fact that the latent space, being lower-dimensional and continuous, removes sparsity in the input space. However, in Fig. 6.3, we can see that, given an ensemble of latent space classifiers $\{f_m\}_{m=1}^{M}$, there is a noticeable variance on how each of the classifiers predict its target probability. In order to minimize the epistemic uncertainty during the process of generating a counterfactual explanation, we propose to use an ensemble of classifiers just like in Schut et al. (2021).

6.2.1. Clarity: presentation and insights

The new objective function is given Equation 6.5. We choose as a starting point the mean of the variational posterior, and thus our algorithm is deterministic.

$$x_{CF} := \mathcal{G}_\theta(z_{CF})\,, \tag{6.4}$$

$$z_{CF} := \underset{z' \in \mathcal{Z}}{\operatorname{argmin}} \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}(f_m(z'), y') + \lambda\, d(z, z'), \tag{6.5}$$

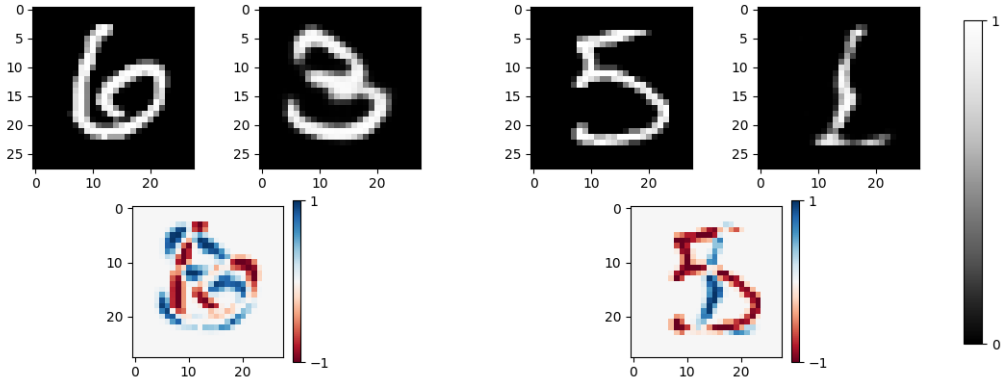
Following this new objective function, we define our counterfactual generation algorithm in Algorithm 5. First, we encode the original image X to the latent space by retrieving the mean of the Gaussian distribution $\mathcal{N}(\mu, \Sigma \,|\, X)$ from the VAE. Then, we apply a gradient descent on the objective function described in Equation 6.5 by using an optimizer like Adam. The algorithm stops once it reaches a certain number of steps N or the classifier has reached a high enough confidence level in the target class. Finally, the resulting latent counterfactual is decoded to retrieve the actual counterfactual image.

Algorithm 5 *Clarity* counterfactual example generation

Input: Original image X, target class y', target probability γ, maximum number of iterations N, hyperparameter λ, VAE with variational posterior $q_\theta(z \,|\, X) = \mathcal{N}(\mu, \Sigma)$ and decoder $\mathcal{G}_\theta(z)$, ensemble of latent space classifiers $\{f_m\}_{m=1}^{M}$, optimizer *opt*.
Output: counterfactual image X_{CF}

$\mu, \Sigma \leftarrow q_\theta(z \,|\, X)$
$z \leftarrow \mu$, $z' \leftarrow z$, $i \leftarrow 0$
while $\frac{1}{M}\sum_{m=1}^{M} p(y' \,|\, z', f_m) \leq \gamma$ *and* $i \leq N$ **do**
  $S(z', y') = \nabla_{z'} \left(\frac{1}{M} \sum_{m=1}^{M} [\mathcal{L}(f_m(z'), y')] + \lambda\, d(z, z') \right)$
  $z' \leftarrow opt(z', S(z', y'))$
  $i \leftarrow i + 1$
end
return $X_{CF} = \mathcal{G}_\theta(z')$

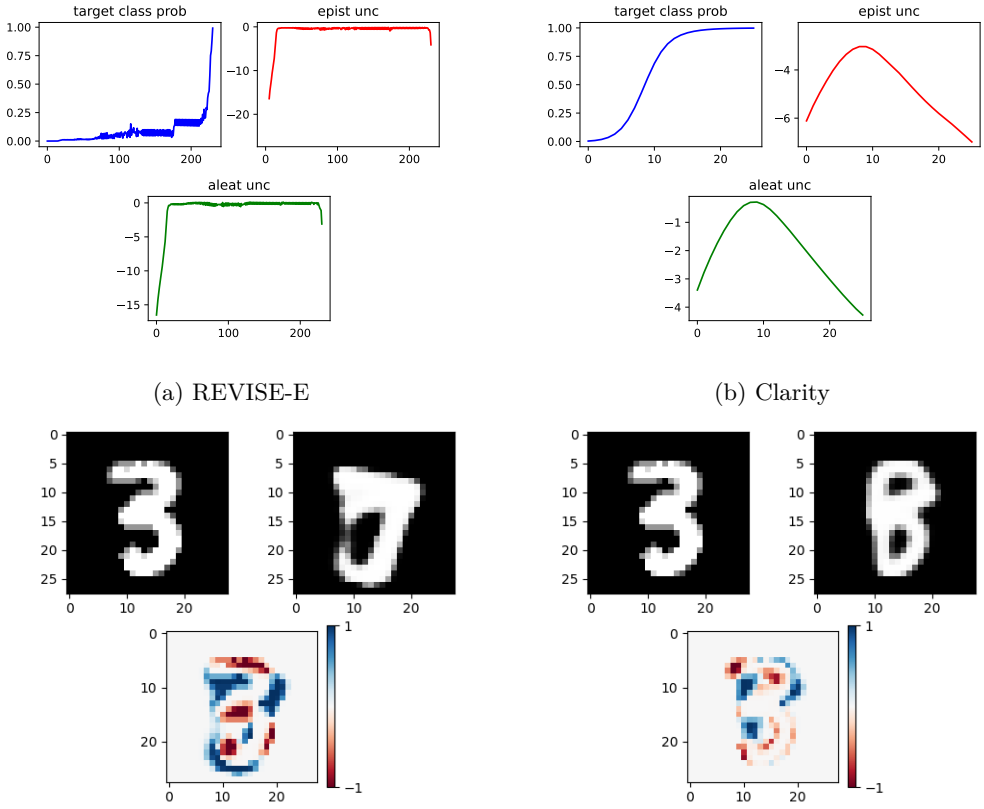
We show qualitative results of this new method. In Fig. 6.6 Fig. 6.6a displays some slight improvements on the counterfactual explanation, as the 3 is more continuous, especially in the middle. However, the most noticeable improvements are seen in Fig. 6.6b. The counterfactual image seems more realistic and does qualitatively represents a digit 1 as opposed to an arbitrary symbol.



(a) $6 \rightarrow 3$ (b) $5 \rightarrow 1$

Figure 6.6.: Counterfactual explanations using *Clarity*

In the example of Figure 6.7, Clarity produces a more realistic counterfactual than REVISE-E. Clarity is also more consistent than REVISE-E in terms of uncertainty, since with REVISE-E the final uncertainties, both aleatoric and epistemic, are much higher than the uncertainties of the original example, whereas the initial and final uncertainties are around the same value with our method. More details about this behavior are detailed in Section 6.2.2.



(a) REVISE-E

(b) Clarity

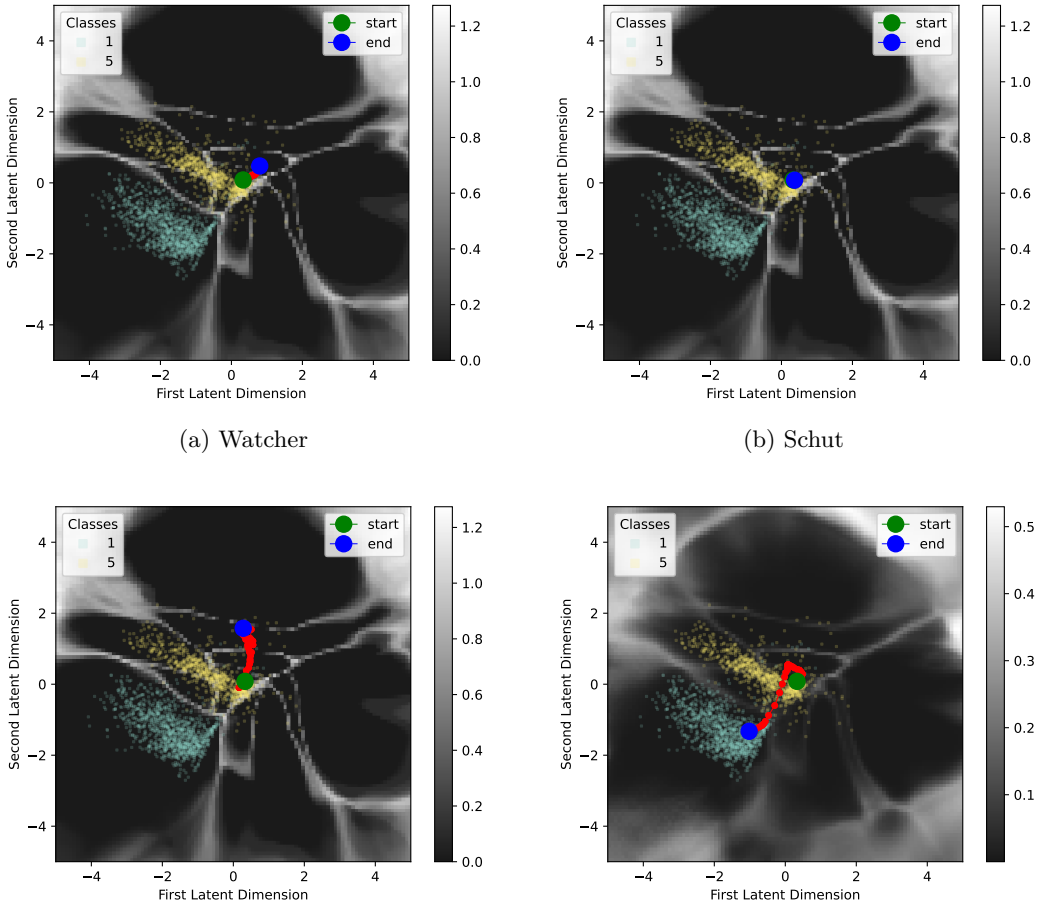
(c) 3 to 8 (REVISE-E)

(d) 3 to 8 (Clarity)

Figure 6.7.: Probability curves of the target class (blue), logarithm of epistemic uncertainty (red) and aleatoric uncertainty (green) along the steps of the counterfactual explanation generation, for both Clarity and REVISE-E. The explanation generated is $3 \rightarrow 8$.

In addition to those counterfactual images, we show the trajectories of the $5 \rightarrow 1$ counterfactual explanations for each method on a 2D latent space, in Fig. 6.8. This 2D latent space has been generated by another VAE, as described in the work of Smith and Gal (2018). The background grey scale represents the epistemic uncertainty of the current classifier (latent space based for *Clarity*, image space based for the others), for each sampled point in the 2D latent space. Firstly, notice how the image space classifier is overconfident in areas that are well beyond the test samples, while the latent space classifier is more conservative with its uncertainty estimates. Secondly, among all

counterfactual generation algorithms, the one given by *Clarity* ends up in the middle of the data distribution of the class "1" (in cyan), which is what we expected. Image space based algorithms fail to change the image in a meaningful way, as the counterfactual is very close to the original example in the 2D latent space, even though the image has been clearly modified. Finally, REVISE-E produces an example that is indeed different in the 2D latent space, but that does not end up in among the examples of the class "1". This is in accordance to the fact that the counterfactual does not look like a 1, see Fig. 5.6d.



(a) Watcher

(b) Schut

(c) REVISE-E

(d) Clarity

Figure 6.8.: Counterfactual explanation $5 \rightarrow 1$ trajectory in a 2D latent space from a VAE. Green dot is starting point and blue dot is end point. Yellow and cyan points are data points from the test distribution of class 5 and 1 respectively. Grey scale represents epistemic uncertainty. Note that Fig. 6.8d has a different uncertainty scale as the classifier is different for Clarity. Zoom for better visualisation.

These empirical results show how the combined techniques given by *Clarity* help to compute a gradient that is more meaningful since the direction of the gradient descent

algorithm leads to a realistic example that lies in the data distribution. Training a classifier on a latent space forces the model to learn class separations that are well-fitted in terms of semantic and high-level features. Furthermore, the use of an ensemble of classifiers helps to reduce the variance in the gradient and to follow a direction that ends up minimizing both epistemic and aleatoric uncertainties.

6.2.2. Additional experiments

We will now describe a series of further experiments that help to better understand the behavior of *Clarity* vs. other counterfactual explanation methods. We will also apply *Clarity* to other datasets.

MNIST experiments. In Table 6.1 we show additional empirical results on the MNIST dataset. Each row represents a particular example, and each column a different counterfactual explanation algorithm, with the exception of the first column being the original image. The VAE encoder uses 3 convolutional layers with Batch Normalization (Ioffe and Szegedy, 2015) with PReLU activation (He et al., 2015). Then, the latent space is of size 16 and the decoder mirrors the encoder with deconvolution layers. For *Clarity*, the latent space classifier is composed of an input layer of size 16, two dense layers of size 16 with ReLU activation, and the final output layer. For all other methods, as explained above, the classifier used has the architecture of the encoder plus the latent space classifier, but now all layers are trained for the classification task. The image space classifiers reach on average an accuracy of around 99% on the test dataset, while our latent space classifier averages its test accuracy at 97%. While the latent space classifiers are a bit less accurate than the image space classifiers, there is room for improvement by using more advanced generative models, which will be left for further research.

Influence of the latent space dimension. Another interesting aspect to analyze is the influence of the dimension of the latent space k on the counterfactual explanations generated by Clarity. In all of the previous experiments, the latent space dimension has been set to $k = 16$. Fig. 6.9 shows the qualitative difference in counterfactual examples for a latent space dimension $k \in \{8, 16, 32, 64\}$. When the dimension is too low ($k = 8$), the explanation collapses to an example is too faint and that does not resemble the original image. The latent space is too compressed to reproduce realistic counterfactuals. On the other hand, for dimensions $k = 32$ and 64, the generated counterfactual examples both represent an arbitrary symbol. As the dimension increases, the latent space becomes sparser and the chance that the gradient stops at a sparse area of the latent space increases, leading to an unrealistic example. Dimension 16 gave the best qualitative results, which is why we kept it for our experiments.

Robustness to noise. We also wanted to validate the robustness of *Clarity* to noise in the data. We experimented by perturbing the images X with a Gaussian noise. More precisely, the new image $X' = (x'_{i,j})$ is defined such that $\forall i, j,\ x'_{i,j} = x_{i,j} + \varepsilon_{i,j}$, with

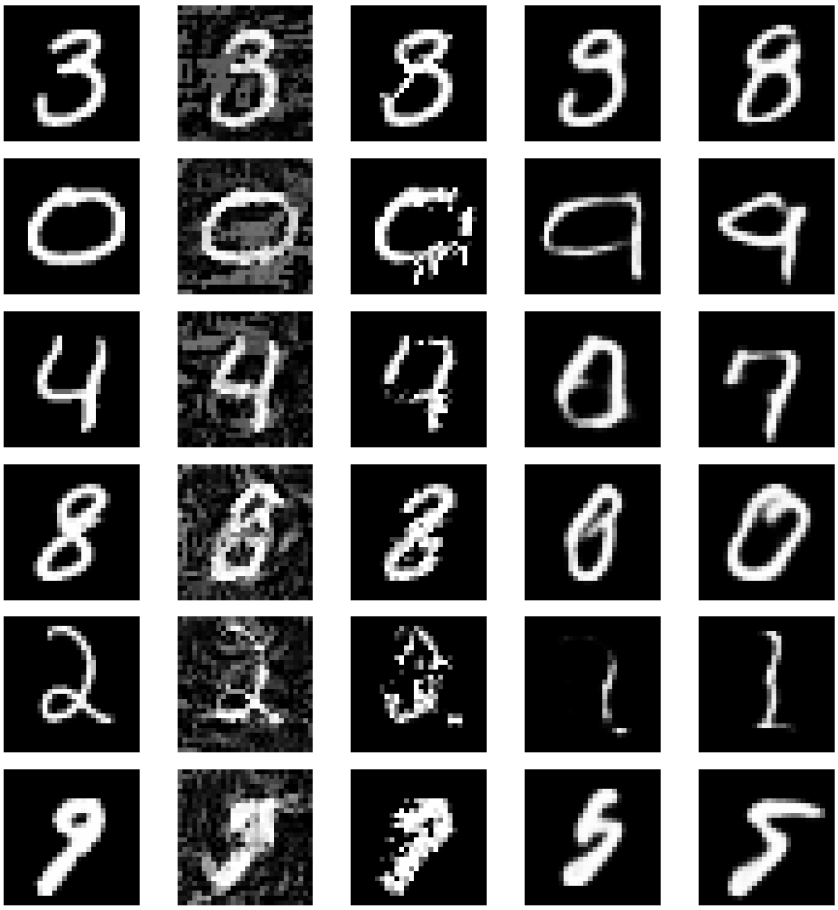
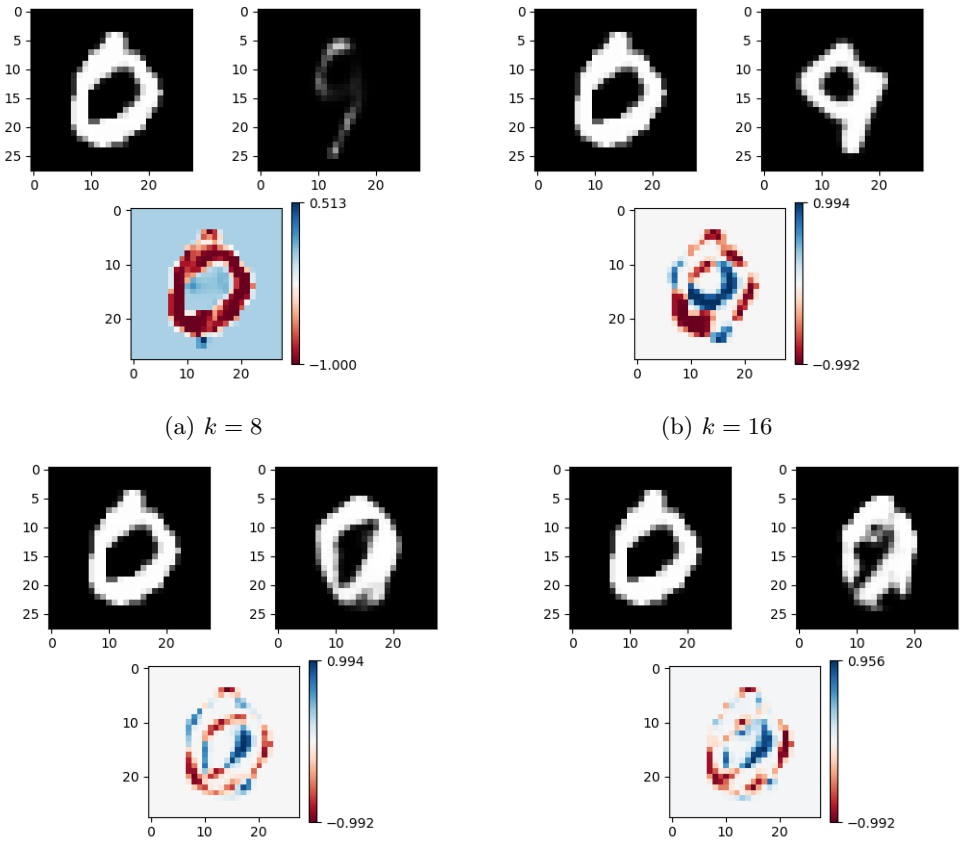
Original	Watcher	Schut	REVISE-E	Clarity
				

Table 6.1.: Counterfactual explanations comparison on the MNIST dataset. From top to bottom, the explanations are: $3 \rightarrow 8, 0 \rightarrow 9, 4 \rightarrow 7, 8 \rightarrow 0, 2 \rightarrow 1, 9 \rightarrow 5$.



(a) $k = 8$ (b) $k = 16$

(c) $k = 32$ (d) $k = 64$

Figure 6.9.: Qualitative comparison of the latent space dimension on the counterfactual explanations. On each subfigure: top left: original image. Top right: Counterfactual explanation. Bottom: Difference between explanation and original image.

$\varepsilon_{i,j} \sim \mathcal{N}(0, 0.01)$, and then we generate a counterfactual X'_{CF} from that perturbed image. We also repeated the same experiment but this time by perturbing the latent encoding z of the image X.

In Fig. 6.10, we can see that Clarity is robust to noise perturbation in the image space. Not only is the generated counterfactual stable from multiple noise samples, but the decoding process of the VAE also acts as a denoiser, as the resulting counterfactual image is noise-free.

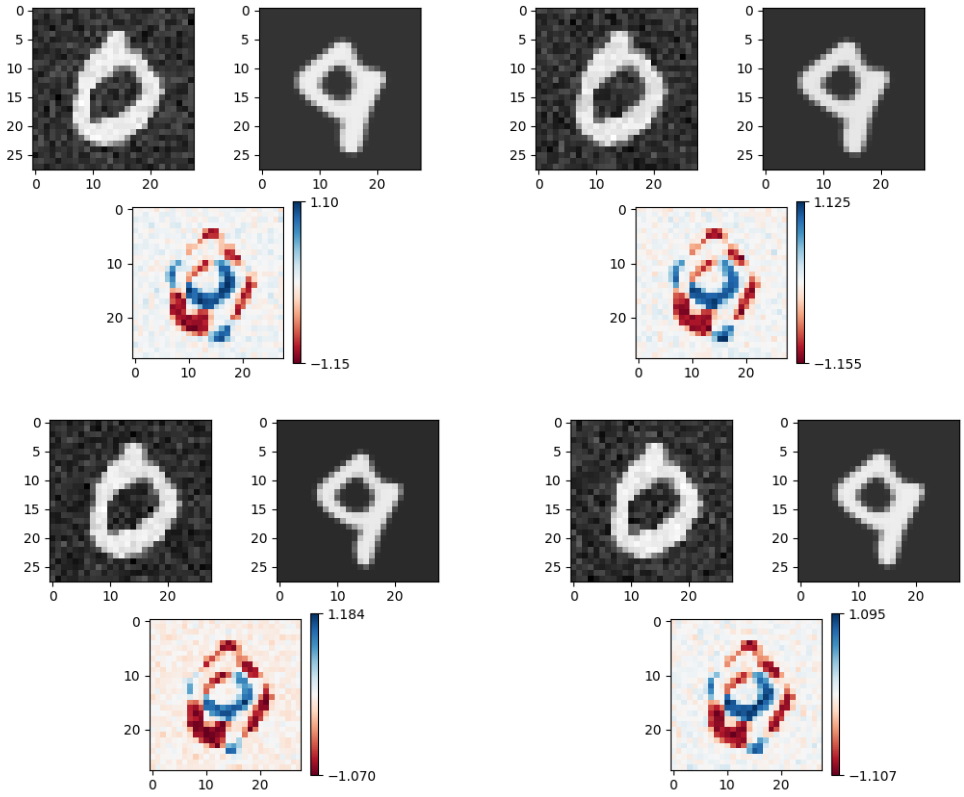


Figure 6.10.: Robustness of Clarity to image space noise perturbation. Each original image has been perturbed by four different and independent samples of a Gaussian noise. The counterfactual explanation is $0 \rightarrow 9$. Color scale may vary.

Fig. 6.11 shows the robustness of Clarity to noise perturbation in the latent space. The original perturbed images were decoded for visualization. We can already see that small perturbations in the latent space induce small perturbations in the image space, an illustration of the continuous property of the latent space. Then, each generated counterfactual has converged around the same mode, with small variations around it. These results indicate that the loss landscape of the trained ensemble of classifiers is "smooth" (this was already hinted in the analysis in Section 6.1.2) and "stable" in the sense that sampling around a latent point won't radically change the closest local minimum for the gradient descent algorithm.

In conclusion, Clarity is robust to both image and latent noise, showing how it can be consistent in its explanations.

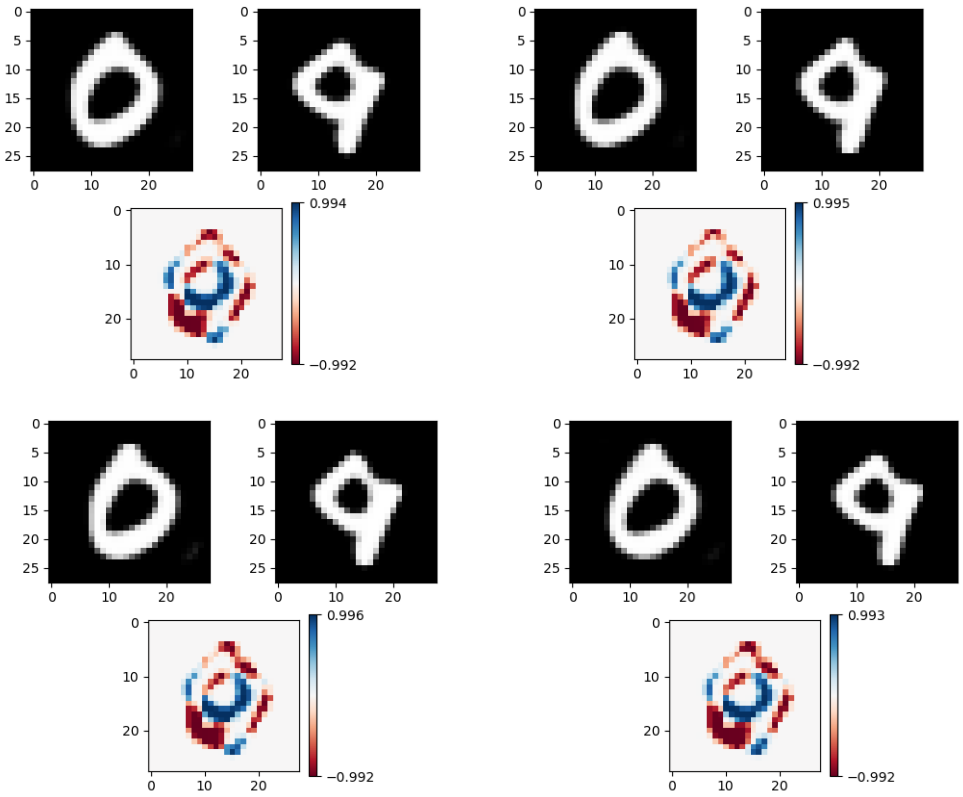


Figure 6.11.: Robustness of Clarity to latent space noise perturbation. Each original image's latent encoding has been perturbed by four different and independent samples of a Gaussian noise. The counterfactual explanation is $0 \rightarrow 9$.

Influence of the hyperparameter λ. As a final experiment on the MNIST dataset, we studied how the hyperparameter λ, which weighs the penalization term related to the distance between the latent encodings (see Equation 6.5), influences the qualitative results of the counterfactuals. We tested values of λ from 0.001 to 1 on several examples. The results are shown in Table 6.2. They highlight the fact that a high λ is detrimental to the quality of the generated counterfactuals. In fact, when $\lambda \geq 0.1$, the regularization term in the objective prevails and the algorithm fails to converge to the target class. Thus we chose a value of $\lambda = 0.001$ for the rest of our experiments.

Realism and uncertainty consistency in both REVISE-ENSEMBLE and *Clarity*. This following experiment analyzes the relation between the consistency in uncertainty estimation in the generated counterfactual explanations and the realism of said explanations in both REVISE-E and *Clarity*. It is important to understand that

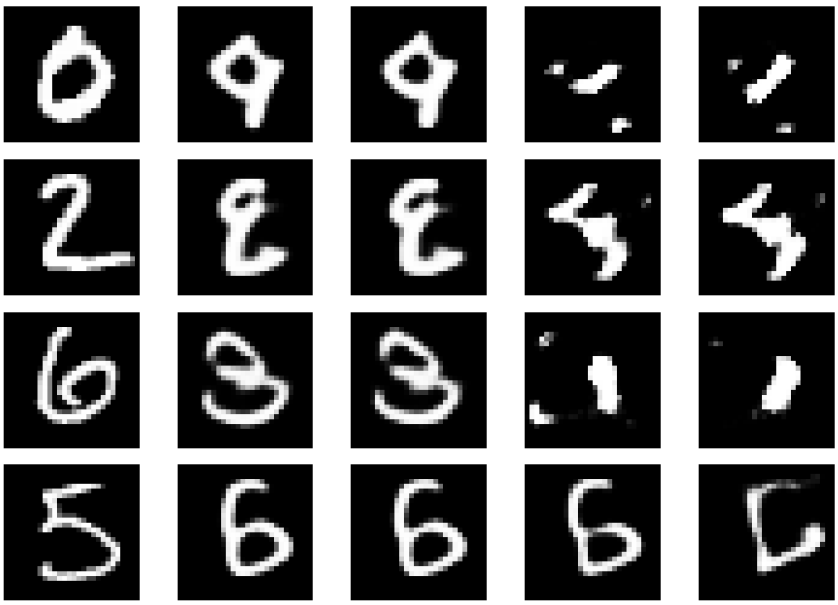
Original	$\lambda = 0.001$	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 1$
				

Table 6.2.: Influence of the hyperparameter λ on the MNIST dataset. From top to bottom, the explanations are: $0 \to 9, 2 \to 8, 6 \to 3, 5 \to 6$.

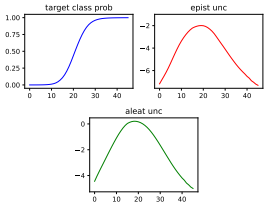
the estimation of these uncertainties is relative to the ensemble of classifiers that is considered. However, a reasonable assumption would be that a realistic counterfactual explanation has a similar scale of uncertainty as the original image. Indeed, a higher uncertainty would mean that the model is under-confident, while a lower uncertainty shows an overconfident prediction. In both cases, it would be difficult to trust the model as these scenarios would highlight a lack of consistency.

In Fig. 6.12, we show the convergence and log-uncertainty curves for several counterfactual explanations, for both REVISE-E and *Clarity*. We can clearly see that in the case of REVISE-E, the model starts by being very confident, but the generated counterfactual has an uncertainty (both aleatoric and epistemic) that is way higher, in several orders of magnitude, making the prediction on the counterfactual example under-confident. On the other hand, *Clarity* ends up being more consistent in terms of uncertainty between the original image and the counterfactual. Indeed, the generated explanation ends up with either the same uncertainty as the the original example or only slightly below, in comparison to REVISE-E. At the same time, the corresponding counterfactual explanations are clearly more qualitative for *Clarity*, which might suggest a probable correlation between these two behaviors.

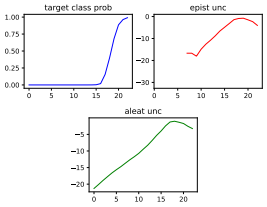
Of course, it is possible to note that the reason REVISE-E is not consistent is because it often starts by being overconfident, and the algorithm stops "too soon" to reach a confidence level that is similar to the original image, leaving the final prediction to be quite under-confident. There are however some issues with prolonging the algorithm for REVISE-E. First, the gradient descent might reach a plateau, preventing it from ever reaching a confidence level similar from the starting point. For instance, see Fig. 6.12 (f) : the slope of the epistemic uncertainty in red is not steep enough to quickly reach a low enough uncertainty. Second, letting the algorithm run for too many steps would extend the computation time, while also applying too many modifications to the original instance, defeating the principle of minimal changes when producing counterfactual explanations.

Thus, we suggest that having a model that is consistent with regards to its uncertainty estimations between the instances and their counterfactuals can help to generate realistic explanations. Further analysis of this behavior to confirm if such link between uncertainty consistency and realism does exist would be an interesting research direction.

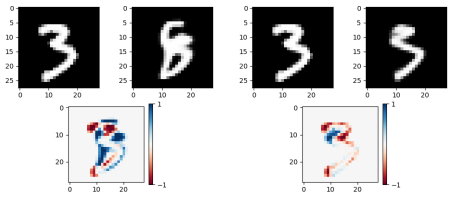
CelebA experiments. We also wanted to validate Clarity on a more "realistic" and detailed dataset. We selected the CelebA dataset (Liu et al., 2015), which is comprised of RGB pictures of the faces of celebrities. These pictures are resized to a resolution of $3 \times 64 \times 64$ pixels. Table 6.3 shows results on that dataset. We used a β-VAE (Higgins et al., 2017) with the same architecture proposed by Subramanian (2020). The latent space itself is of dimension 128. For both REVISE-E and Clarity, we used an ensemble of 20 models with adversarial training. There are two versions of REVISE-E: R-E (image) used a penalty on the image space $d(\mathcal{G}_\theta(z'), X)$, and R-E (latent) uses a penalty on the latent space $d(z', z)$, similarly to Clarity. The classifiers were trained to predict the hair color. The goal of these explanations is to change it without modifying other features. Overall, both methods offer similar results, but Clarity seems closer to the original image,



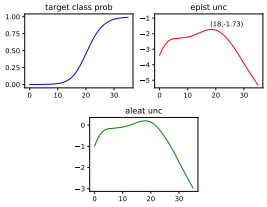
(a) 3 to 8 (Clarity)



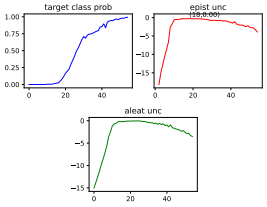
(b) 3 to 8 (REVISE-E)



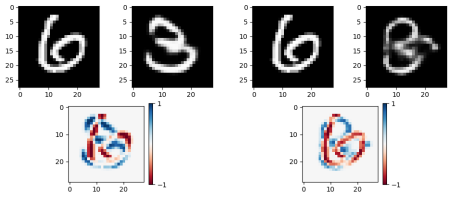
(c) 3 to 8 (Clarity) (d) 3 to 8 (REVISE-E)



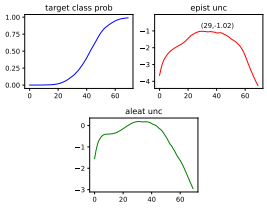
(e) 6 to 3 (Clarity)



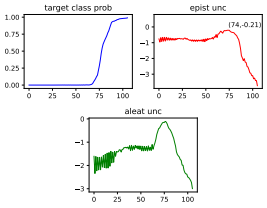
(f) 6 to 3 (REVISE-E)



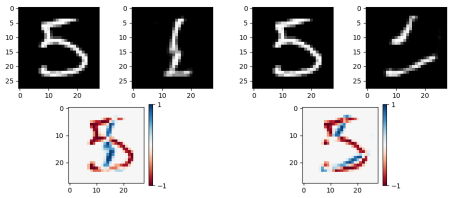
(g) 6 to 3 (Clarity) (h) 6 to 3 (REVISE-E))



(i) 5 to 1 (Clarity)



(j) 5 to 1 (REVISE-E)



(k) 5 to 1 (Clarity) (l) 5 to 1 (REVISE-E)

Figure 6.12.: Analysis of the consistency in uncertainty with both Clarity and REVISE-E. On the left: Probability curves of the target class (blue), logarithm of epistemic uncertainty (red) and aleatoric uncertainty (green) along the steps of the counterfactual explanation generation, for both Clarity and REVISE-E. On the right: the corresponding counterfactual explanations. The explanations generated are $3 \rightarrow 8$, $6 \rightarrow 3$ and $5 \rightarrow 1$. Special note in Fig. (b): the epistemic curve is cut at the beginning as the uncertainty is too small to be defined.

Original	VAE	R-E (latent)	R-E (image)	Clarity
				

Table 6.3.: Hair color counterfactual explanation comparison on the CelebA dataset. From top to bottom, the explanations are: Brown → Black, Black → Grey, Blond → Black, Brown → Blond, Black → Grey, Black → Blond. Columns respectively represent the original image, the same image reconstructed by the VAE and the images obtained by the different counterfactual explanation methods. R-E is REVISE-E, with an image or latent distance penalization.

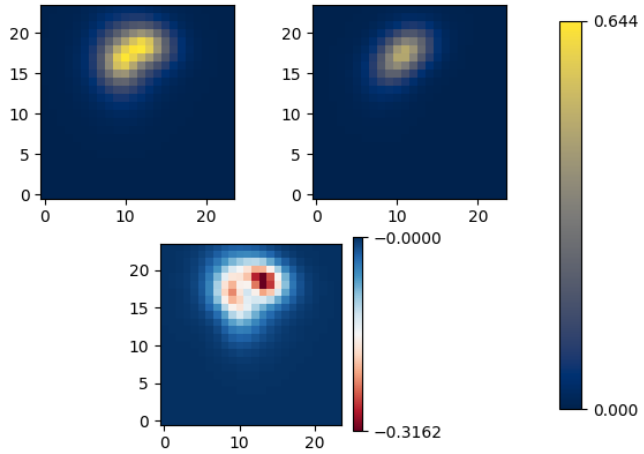
unlike REVISE-E that changes the gender or the face of the person, their skin color or add unnecessary attributes like makeup.

Galaxy images. Finally, we apply Clarity to galaxy images generated by Galsim (Rowe et al., 2014) (see Section 4.1.2). Instead of predicting the complex ellipticity ε, we change the task by training a neural network to classify between isolated and blended scenes, and then ask the model to generate counterfactual explanations from this binary classification model.

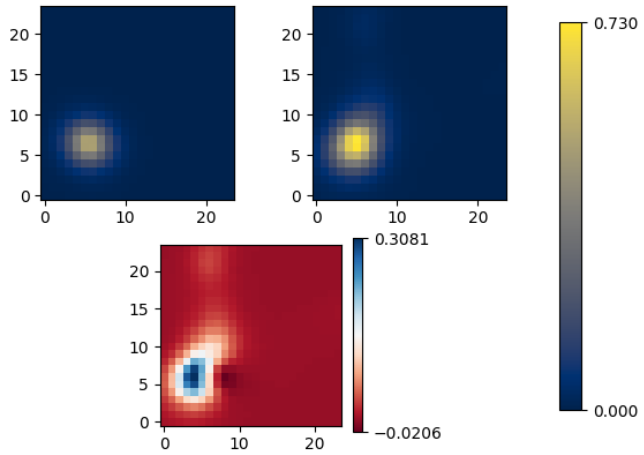
Since we noticed that the variance in brightness is high between the galaxies (some can be very bright and some very faint, up to a factor of 10^5 between the faintest and the brightest), we normalized the brightness between all galaxies so the network cannot rely on brightness to separate the two classes, but rather focus on the shape of the galaxies. More precisely, to have maximum brightness of 1 in the case of two perfectly blended

galaxies, all galaxies' brightness have been normalized such that, for an isolated scene X, $\max X_{i,j} = 0.5$.

Finally, still in the goal of analyzing how the network would capture the shape of the galaxies without any perturbation, the images generated for the training dataset are noiseless.



(a) Blended → Isolated



(b) Isolated → Blended

Figure 6.13.: Counterfactual explanations using *Clarity* on galaxy images

In Fig. 6.13 we can see the two possible counterfactual explanations possible in this binary classification problem. Obviously, in the case of the Blended → Isolated explanation, the algorithm does not act as a deblended per se, but rather "merges" the two blended objects together to form a new galaxy with the correct brightness, which is 0.5. Then in the other way, the algorithm has a tendency to add a new galaxy on top of the existing one, resulting in two superimposed galaxies with a peak brightness higher

Table 6.4.: IM1 and L1 latent distance for 100 and 50 random counterfactual explanations on the MNIST dataset and CelebA dataset respectively, for the different counterfactual algorithms. Value is mean ± standard deviation. Lower is better. In bold is the best value.

	Watcher et al.	REVISE-E	Clarity
IM1	1.06 ± 0.28	0.93 ± 0.35	$\mathbf{0.68 \pm 0.24}$
L1	4.82 ± 1.22	$\mathbf{2.76 \pm 1.02}$	3.34 ± 1.07

(a) MNIST

	REVISE-E	Clarity
IM1	$\mathbf{1.24 \pm 0.73}$	1.35 ± 0.82
L1	10.38 ± 7.7	$\mathbf{9.73 \pm 6.97}$

(b) CelebA

than a single galaxy (or two separated ones). These explanations give us the insight that the neural network still mostly relies on brightness to separate isolated scenes from blended ones, even if the dataset contains some cases of totally separated galaxies where the peak brightness does not exceed 0.5. Still, the generated counterfactuals are in each case plausible and realistic, as they feature objects that are not unlike galaxies.

Quantitative analysis. Table 6.4 highlights the means and standard deviations of the IM1 metric (see Section 5.2.2) over a sample of images and target classes on both the CelebA and MNIST dataset, with a sample of target classes, for the different counterfactual algorithms. We omit Schut as it fails to converge in most of the cases anyway (Schut et al., 2021). In addition, we also show the L1 distance in the latent space of the counterfactuals. On the MNIST dataset, we can see that the techniques described in this paper help to achieve a better IM1 value on average, emphasizing the importance of exploiting the structure of the latent space to produce better visual counterfactual explanations. In contrast, REVISE-E achieves a lower latent distance, but this is at the cost of realism for the explanations. Inversly, on the CelebA dataset, REVISE-E achieves a lower IM1 score at the cost of a higher L1 distance. Using an ensemble of latent space classifiers on the other hand still offers a good, albeit slightly higher, IM1 metric with a lower L1 distance. This is coherent with the results shown in Table 6.3 our method is more conservative, so the picture is closer to the original image, while REVISE-E can generate significant changes which will reduce the IM1 score at the cost of the L1 distance. In conclusion, the IM1 score alone cannot help to conclude whether a counterfactual explanation is satisfactory or not, as one must also consider the balance between realism of a given target class, and distance from the original picture.

6.3. Gradient algorithm based on Student's t-tests

So far, we have exploited the ensemble of classifiers by averaging their loss functions during the gradient descent algorithm. However, it is possible to consider another approach. A policy that we can define is the following: for each step of the gradient descent algorithm, we want to update the components of the counterfactual for which the gradient mean is the farthest from zero. Theoretically, these components would indicate the step in one dimension which minimizes the objective the most. This is done by analyzing the distribution of gradient samples from the ensemble, and identifying which non-zero components the models agree on. In this section, we will describe the theoretical foundations exploiting these ideas and applying them experimentally.

6.3.1. Testing the null hypothesis

Consider the gradient of the mean of the ensemble

$$S(z', y') := \nabla_{z'} \left(\frac{1}{M} \sum_{m=1}^{M} \left[\mathcal{L}(f_m(z'), y') \right] + \lambda \, d(z, z') \right) . \tag{6.6}$$

Another possibility is to consider the ensemble of gradients $(S_m(z', y'))_{m=1}^{M}$, with

$$S_m(z', y') := \nabla_{z'} \left(\mathcal{L}(f_m(z'), y') + \lambda \, d(z, z') \right) . \tag{6.7}$$

More generally, we can define a matrix $\mathcal{S}(z', y') \in \mathbb{R}^{k \times M}$ such that the m^{th} column of $\mathcal{S}$ is equal to $S_m(z', y')$ where $\mathcal{S}(z', y')_{i,m} = s_m^i$. For each row of $\mathcal{S}(z', y')$, we have a set of samples of the i^{th} component of the gradient $(s_m^i)_{m=1}^{M}$. This sample has an empirical mean $\bar{s}_i$ and unbiased empirical standard deviation $\bar{\sigma}_i$ such that

$$\bar{s}_i := \frac{1}{M} \sum_{m=1}^{M} s_m^i \, , \tag{6.8}$$

$$\bar{\sigma}_i := \sqrt{\frac{1}{M-1} \sum_{m=1}^{M} (s_m^i - \bar{s}_i)^2} \, . \tag{6.9}$$

It is then possible to test, for each gradient component i, the null hypothesis $H_0^i : \mu_i = 0$ where μ_i is the mean of the distribution generating the samples s_m^i: $\mu_i = \mathbb{E}[s^i]$. The components who fail the hypothesis, depending on a risk level $\alpha \in [0, 1]$, have a mean gradient that is far from zero and would be preferred for the counterfactual search.

Updating fewer components would help to ensure the minimality property of counterfactual explanations, while keeping the uncertainty low. This policy would allow for more efficiency in that regard, by only selecting the most relevant components in the minimization of the objective function.

To test these hypotheses, we define for each component a statistic ζ_i

$$\zeta_i := \sqrt{M}\frac{\bar{s}_i}{\bar{\sigma}_i}\,. \tag{6.10}$$

Each such statistic follows a Student's t-distribution with $M-1$ degrees of freedom under H_0^i. Given a risk level $\alpha \in [0,1]$, the hypothesis H_0^i fails if $|\zeta_i| \geq t_{1-\frac{\alpha}{2}}^{M-1}$, where t_α^k is the α-quantile of the Student's t-distribution with k degrees of freedom.

Algorithm 6 details the algorithm to generate a counterfactual explanation using these Student's t-tests. We first compute the ensemble of gradients, and then compute the statistics for each component of the latent space. Then, we update the latent embedding using only the components failing the test. Another possibility would be to only select the top-N components with the highest value of the statistic $|\zeta_i|$, which allows us to control the number of components which are updated, and to make sure that a minimal (and maximal) number of components are selected every time.

Algorithm 6 *Clarity* counterfactual example generation based on Student's t-test

Input: Original image X, target class y', target probability γ, maximum number of iterations N, hyperparameter λ, VAE with variational posterior $q_\theta(z \mid X) = \mathcal{N}(\mu, \Sigma)$ and decoder $\mathcal{G}_\theta(z)$, ensemble of latent space classifiers $\{f_m\}_{m=1}^M$, optimizer *opt*, risk level α.
Output: counterfactual image X_{CF}

$\mu, \Sigma \leftarrow q_\theta(z \mid X)$
$z \leftarrow \mu$, $z' \leftarrow z$, $c \leftarrow 0$
while $\frac{1}{M}\sum_{m=1}^M p(y' \mid z', f_m) \leq \gamma$ *and* $c \leq N$ **do**
  for $m = 1$ *to* M **do**
    $S_m(z', y') := \nabla_{z'}\left(\mathcal{L}(f_m(z'), y') + \lambda\, d(z, z')\right)$
  end
  for $i = 1$ *to* k **do**
    $\bar{s}_i := \frac{1}{M}\sum_{m=1}^M s_m^i$
    $\bar{\sigma}_i := \sqrt{\frac{1}{M-1}\sum_{m=1}^M (s_m^i - \bar{s}_i)^2}$
    $\zeta_i := \sqrt{M}\frac{\bar{s}_i}{\bar{\sigma}_i}$
  end
  $I_\alpha := \left\{ i \in [\![1, k]\!] \,\middle|\, |\zeta_i| \geq t_{1-\frac{\alpha}{2}}^{M-1} \right\}$
  for $i \in I_\alpha$ **do**
    $z_i' \leftarrow \mathit{opt}(z_i', \bar{s}_i)$
  end
  $c \leftarrow c + 1$
end
return $X_{CF} = \mathcal{G}_\theta(z')$

6.3.2. Experimental results

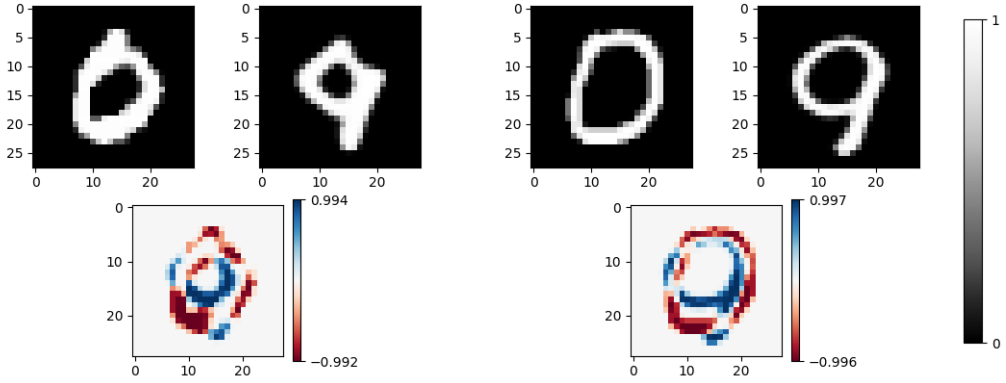
We have conducted several experiments aiming to analyze the performance of the gradient algorithm based on Student's t-tests. Notably, by looking at the dependency on the risk

level α, and the selected components, compared to the usual approach without Student's t-tests.

Using a risk level α. In the following experiments, we use the MNIST dataset (LeCun et al., 2010), using the same setup as in Section 6.2.2, except that we are using Algorithm 6 instead of Algorithm 5. Firstly, as a point of comparison, Fig. 6.14 shows the qualitative results for the previous Clarity algorithm (Algorithm 5).

Fig. 6.15 on the other hand, presents qualitative results for the same counterfactual examples but generated with Algorithm 6, with $\alpha = 0.01$. We empirically noticed that a higher value of α is too lenient and is basically equivalent to selecting all components. On average around 11 or 12 components fail the null hypothesis with a risk $\alpha = 0.01$. When the target probability gets close to 0.99, fewer components fail the null hypothesis as we approach a local minimum. If α is too low, the algorithm stops before reaching the local minimum as the hypothesis becomes too strict.

Overall, we can see in Fig. 6.15 that the generated counterfactuals are qualitatively on par to the ones generated withtout Student's t-tests, but no significant improvement in any side can be seen.

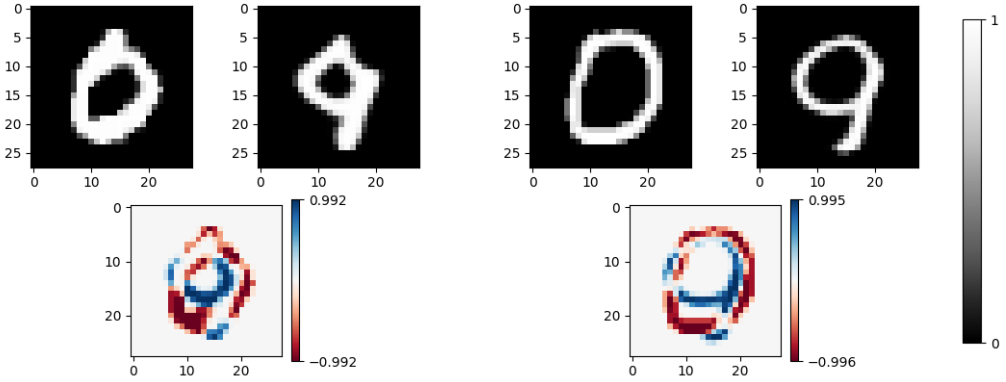


(a) $0 \to 9$ (b) $0 \to 9$

Figure 6.14.: Counterfactual explanations with algorithm 5. On each subfigure: top left: original image. Top right: Counterfactual explanation. Bottom: Difference between explanation and original image.

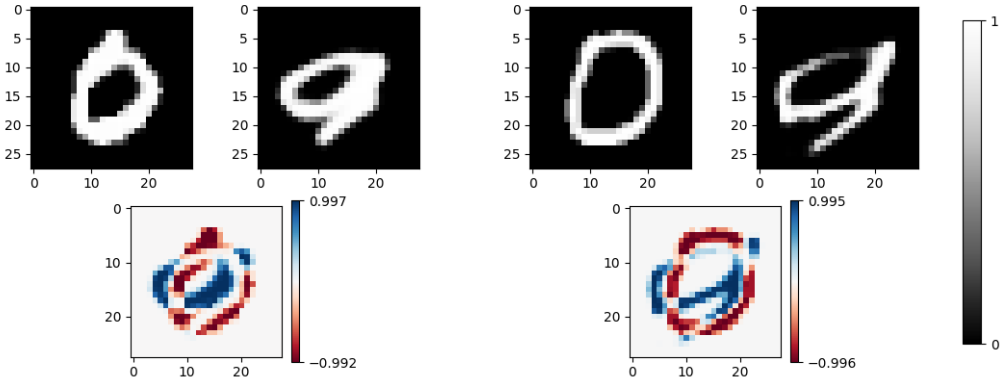
Selecting the components with the highest statistic. We also tried different variations of Algorithm 6. Firstly, we tried to select the 10 components of the gradient with the highest value of the statistic $|\zeta_i|$. This allows us to remove the hyperparameter α and ensure a minimal (and maximal) number of components are updated everytime. On Fig. 6.16, we can see that the generated counterfactuals are less qualitative compared to other methods. We therefore decided not to use this approach.

Then, we tried selecting a unique gradient component which has the highest statistic $|\zeta_i|$. So instead of updating multiple components of the latent embedding at the same



(a) $0 \to 9$ (b) $0 \to 9$

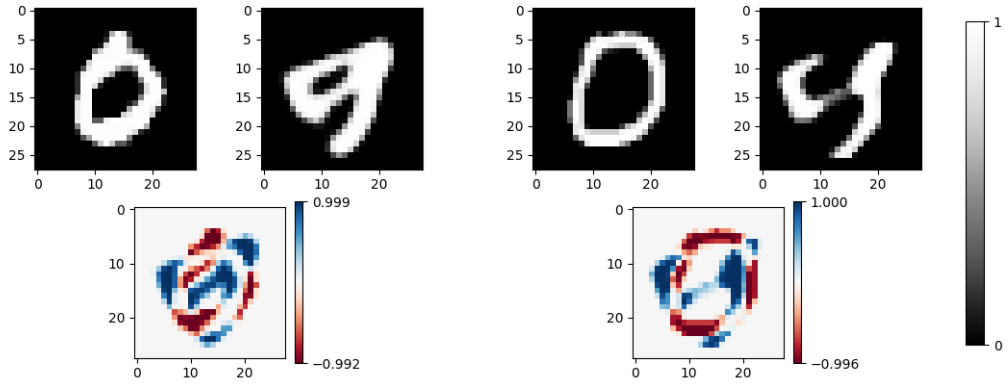
Figure 6.15.: Counterfactual explanations with algorithm 6 and $\alpha = 0.01$. On each subfigure: top left: original image. Top right: Counterfactual explanation. Bottom: Difference between explanation and original image.



(a) $0 \to 9$ (b) $0 \to 9$

Figure 6.16.: Counterfactual explanations by selecting the 10 components with the highest statistic $|\zeta_i|$. On each subfigure: top left: original image. Top right: Counterfactual explanation. Bottom: Difference between explanation and original image.

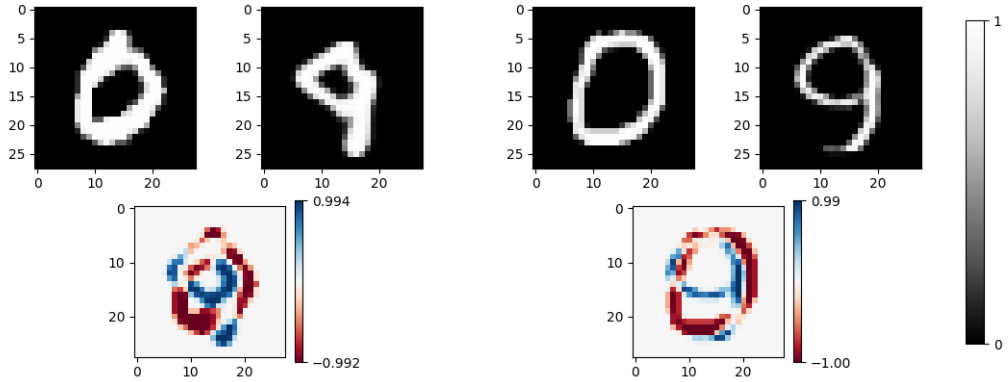
time, we only update one at a time. The counterfactual explanations in Fig. 6.17 are still less qualitative compared to selecting all components at the same time.



(a) $0 \to 9$ (b) $0 \to 9$

Figure 6.17.: Counterfactual explanations by selecting the component with the highest statistic $|\zeta_i|$. On each subfigure: top left: original image. Top right: Counterfactual explanation. Bottom: Difference between explanation and original image.

Finally, we tried a final variant removing the Student's $t-$test and the statistics $|\zeta_i|$. Instead, we update all the components of the gradient at the same time, but we update each component i by the median of the gradients $(s_i^m)_{m=1}^M$. In Fig. 6.18, the results are more qualitative than selecting the top statistics, but a bit subpar to the Student's $t-$test method or the first, average method.



(a) $0 \to 9$ (b) $0 \to 9$

Figure 6.18.: Counterfactual explanations by selecting the median of each component. On each subfigure: top left: original image. Top right: Counterfactual explanation. Bottom: Difference between explanation and original image.

6.3.3. Conclusion

In this section, we exploited the statistical information contained by the ensemble of classifiers $\{f_m\}_{m=1}^M$ by considering the ensemble of gradients S_m rather than the gradient of the loss average. By using Student's $t-$tests, we achieved qualitative results on par with the average method used in Section 6.2. We also studied how selecting certain components depending on the value of their statistic or by their median can influence the resulting counterfactuals. While the results are not clearly better, they provide useful insights. It is possible that the latent space is too dense and "smooth" for these statistical tests to be truly relevant, and it would be interesting to see, for instance, how this technique could behave in latent spaces of higher dimension that can be used for more complex and detailed data.

6.4. Discussion and conclusion

Clarity contributes to make models more interpretable by design, and capable of generating realistic and unambiguous counterfactual images with minimal unnecessary changes. Moreover, such counterfactual images are an order of magnitude faster to compute.

At a more fundamental level, we give insights, supported by experiments, into the reason why applying a gradient-based method to a classifier trained in the latent space of a VAE is more likely to produce quality counterfactual images in comparison to a classifier trained in the image space.

We will now discuss the possible implications of the practical use of these findings, the current limitations of our method as well as potential directions for further research. While counterfactual explanations are a useful tool to debug deep neural networks and to show modes of failure, they are often useless to end users if they do not make sense to them. Therefore, we hope that the results shown in this paper will help to put in place models that are designed to be understandable to any person. In addition, we believe that models interpretable "by design" might be a key to produce fairer models that do not discriminate in salient/sensible features irrelevant to the prediction task.

On the other hand, this approach is dependent on the presence of a generative model that, if not trained properly, could lead to undesirable side effects, such as introduction of possible biases in the generation process. For instance, a malicious user could train the generative model in a biased manner to create a classifier that is also biased and unfair by design. Another limitation of this method is the fact that the level of detail, and thus, the level of realism, of the produced counterfactuals rely on the quality of the generative model of the VAE, which is known to be inferior to other types of generative models such as GANs. Special care must therefore be taken in the training stage of the VAE.

This last remark leads us to a perspective: all of the techniques presented on this paper could be adapted to any generative model, such as GANs or Normalizing Flows (Rezende and Mohamed, 2015) or more sophisticated VAE architectures (Larsen et al., 2016; Dai and Wipf, 2019). Several counterfactual explanation methods already use GANs (Liu et al., 2019; Lang et al., 2021; Zhao, 2020; Arrieta and Ser, 2020) and even

diffusion models (Sanchez and Tsaftaris, 2022). This opens several new and interesting directions of further research, namely, investigating the behavior of these techniques on such generative models, in order to overcome the limitations of VAEs in terms of image sharpness and level of detail, and the performance of the latent space classifiers.

Chapter 7.

Conclusion and future prospects

7.1. Summary on the main contributions

In this thesis, we focused our attention on the problematic of developing ways to make neural networks more transparent in the context of astronomical data analysis. In order to achieve this goal, two main ideas were proposed. The first was to use Bayesian Neural Networks as a mean to estimate two separate sources of uncertainty, i.e. epistemic and aleatoric uncertainties. Then, we studied how Bayesian deep learning can improve the explainability of deep neural networks, and how the knowledge about the uncertainties can be added to other techniques to further enhance the quality and interpretability of the explanations.

The foundation of this thesis is built upon the notion of uncertainty, and as such, it was necessary to ensure we were able to learn models giving a calibrated estimation of the uncertainty. In Chapter 3, we first proposed an algorithm based on Dropout Regulation, which allows for an automated adjustment of the dropout rate with respect to the rate of overfitting. We found that this approach allowed for the Bayesian Neural Network to estimate uncertainties which were more reliable than other approaches in the state-of-the-art. Then, Chapter 4 described how Bayesian Neural Networks can be used in an astrophysical context. More precisely, we developed a Bayesian Convolutional Neural Network to learn a multivariate regression task whose goal was to estimate the complex ellipticity of galaxy images. This model was able to estimate a calibrated aleatoric uncertainty depending on the presence of noise in the image. Also, the epistemic uncertainty was very reliable to separate images of isolated galaxies from blended scenes, which were considered to be outliers here. This proved that these uncertainty estimates were very useful in the context of astronomical data. Finally, we showed how Bayesian Neural Networks can be used to correlate galaxy positions with uncertainty, and how they can also be used to enhance the learning process via Network coaching.

Following these deep insights on uncertainty estimation with Bayesian Neural Networks, we turned our eyes in Chapter 5 to another facet of transparency in deep learning: explainability. More precisely, we asked how Bayesian Deep Learning can improve explainability with the added information given by a probability distribution on the parameters of the network. Given the fact that uncertainty gives information about the data distribution itself, it was natural to take interest in an example-based method of explainability: counterfactual explanations. After looking at different state-of-the-art methods for counterfactual visual explanations and how they all had some form of failure

point, we proposed in Chapter 6 a new method, called *Clarity*, which combined the exploitation of a latent space from a generative model with Bayesian Deep Learning. We presented ideas and insights about why using a latent space to train a classifier instead of the image space allows for more realistic uncertainty, especially when combined with a Bayesian Neural Network. Our method achieved results on par, if not better than the state-of-the-art, in terms of counterfactual explanation's quality and realism. Finally, we presented variants of *Clarity* exploiting the Bayesian aspect by using Student's t-tests in an attempt to further improve the quality of counterfactual explanations.

Overall, the research done in this thesis culminated to the development of new methods and techniques to enhance the transparency of neural networks in the context of astronomical data. The added insights about these contributions give new light to explainability and uncertainty estimation of neural networks in general, especially in the context of image processing and computer vision, and we believe that they will be useful in many domains of application. However, we also believe that there are still many new practices, methods and behaviors left to be discovered and studied, and we will now give some avenues for further research.

7.2. Future prospects

The width of the research subjects addressed in this thesis was large, going from uncertainty estimation, data analysis on astronomical data, to counterfactual explanations. We will give several ideas that could be developed for further research in each subject.

7.2.1. On the study of astronomical data and uncertainty estimation

The first contribution of this thesis was the development of the Dropout Regulation algorithm in Chapter 3. We presented and tested the regulation algorithm using a PI controller on the dropout rate. However, there may be other controllers which might be better suited to adapt the dropout rate, such as the full PID controller instead of just using the PI component. Moreover, it is realistically possible to use the idea of regulation to the hyperparameters of other variational approaches of Bayesian Neural Networks.

In Chapter 4, we developed several methods and models to improve the estimation of uncertainty on astronomical data, more specifically galaxy images and the estimation of their complex ellipticity. However, what we developed was more of a proof of concept, as the dataset was simulated and not collected from real data. Galaxy images are usually registered in 6 different light bands, making them more similar to RGB data with a notion of depth in the image. This added information could be useful to study how the model could learn from blended scenes more efficiently, as the light bands could help the network to separate those galaxies. An interesting prospect would be to study the influence of uncertainty estimation with the added complexity of these real images collected from LSST.

Another obvious part that could be studied more in depth is the idea of Network Coaching. The preliminary results were promising, showing a potential with the technique

of slowly adding complexity in the dataset. We believe that this algorithm could be improved, and would maybe perform better in a dataset that is more suited for this kind of approach. Finally, the idea of Network Coaching is not only related to active learning, but also transfer learning. Performance could possibly be improved by intelligently combining all of these approaches.

7.2.2. On the study of counterfactual explanations

Chapter 5 and 6 focused on explainable AI, and more precisely counterfactual visual explanations. While *Clarity* can produce quality counterfactual explanations, it still presents some issues. The generation process relies on a VAE, which adds artifacts to the generated images such as blur, reducing the quality and realism of the explanations. It could be interesting to adapt *Clarity* to other generative models. Sanchez and Tsaftaris (2022) for instance used a diffusion model to generate counterfactual explanations, which can achieve very high image quality even on high resolution images.

Also, the idea of a gradient based method can still fail even in the presence of a Bayesian latent space classifier. Other researchers, such as Looveren and Klaise (2021) or Downs et al. (2020) used a method to search counterfactuals based on heuristics and sampling. Robustness to adversarial examples and ambiguous explanations could be improved by combining ideas from other approaches.

Subsequently, the prospect of using Student's t-tests to improve the quality of counterfactual explanations was interesting, but perhaps the dataset and setup were not the ones best suited for this technique. We believe that this approach deserves more in-depth research, possibly by using it on more complex datasets with latent spaces of higher dimension.

Finally, techniques used to generate realistic counterfactual explanations could also be used in the context of attribute transfer. As opposed to counterfactuals, attribute transfers is not a way to explain a model but rather to train the model itself to generate specific changes in the input. Li et al. (2016), for instance, use a CNN to generate a new image from a reference image while keeping the identity of the original person, while He et al. (2017) proposed the AttGAN architecture for attribute manipulation on images using a GAN. It would be interesting to adapt counterfactual explanation methods in this context and see how they could benefit each other.

7.3. List of publications

- Claire Theobald, Frédéric Pennerath, Brieuc Conan-Guez, Miguel Couceiro and Amedeo Napoli "A Bayesian Neural Network based on Dropout Regulation". In *Workshop on Uncertainty in Machine Learning (WUML) at ECML-PKDD 2020 conference.*
- Claire Theobald, Bastien Arcelin, Frédéric Pennerath, Brieuc Conan-Guez, Miguel Couceiro and Amedeo Napoli "A Bayesian Convolutional Neural Network for robust galaxy ellipticity estimation". In *Machine Learning and Knowledge Discovery in*

Databases. Applied Data Science Track - European Conference, ECML PKDD 2021, Bilbao, Spain, September 13-17, 2021, Proceedings, Part V, volume 12979 of *Lecture Notes in Computer Science*, pages 135-150, Springer.

Papers submitted to an international conference

- Claire Theobald, Frédéric Pennerath, Brieuc Conan-Guez, Miguel Couceiro and Amedeo Napoli "Clarity: an improved gradient method for producing quality visual counterfactual explanations.". Submitted to an international conference.

Part I.

Appendices

Appendix A.

General definitions

- Given a discrete random variable $X \in \mathcal{X}$ with probability distribution p, and a function $g : \mathcal{X} \to \mathbb{R}$, the expected value of $g(X)$ given p, noted $\mathbb{E}_{p(X)}[g(X)]$, is equal to

$$\mathbb{E}_{p(X)}[g(X)] := \sum_{x \in \mathcal{X}} g(x)\, p(x)\,. \tag{A.1}$$

  In the case of a continuous variable, with a probability density p, it is defined as

$$\mathbb{E}_{p(X)}[g(X)] := \int g(x)\, p(x)\, dx\,. \tag{A.2}$$

- Given a discrete random variable $X \in \mathcal{X}$ with probability distribution $p(X)$, the entropy of X, noted $\mathbb{H}(X)$, is equal to

$$\mathbb{H}(X) := \mathbb{E}_{p(X)}[-\log p(X)] = -\sum_{x \in \mathcal{X}} p(x) \log p(x)\,. \tag{A.3}$$

- Given two discrete probability distributions p and q defined on an input space $\mathcal{X}$, the Kullback-Leibler divergence between p and q, noted $D_{KL}(p \,||\, q)$, is defined as

$$D_{KL}(p \,||\, q) := \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}\,. \tag{A.4}$$

  For two continuous distributions with respective densities p and q, it is defined as

$$D_{KL}(p \,||\, q) := \int p(x) \log \frac{p(x)}{q(x)}\, dx\,. \tag{A.5}$$

- Given two random variables (X, Y) defined on $\mathcal{X} \times \mathcal{Y}$, with a joint probability distribution $p(X, Y)$ and respective marginal distributions $p(X)$ and $p(Y)$, the mutual information of X and Y, noted $\mathbb{I}(X, Y)$, is defined as

$$\mathbb{I}(X, Y) := D_{KL}(p(X, Y) \,||\, p(X)\, p(Y))\,. \tag{A.6}$$

Appendix B.

Résumé détaillé

B.1. Introduction

Alors que les systèmes d'intelligence artificielle deviennent de plus en plus performants, la complexité de ces modèles et la difficulté liée à la compréhension et à la confiance que l'on peut avoir envers ces derniers augmentent également en conséquence. Ce besoin d'obtenir des modèles intelligibles et transparents va cependant bien au-delà des questions de réglementation et d'éthique : ils sont également très précieux dans le domaine scientifique. À mesure que l'utilisation de modèles d'apprentissage automatique continue d'augmenter dans de nombreux domaines scientifiques, le besoin de modèles fiables et interprétables augmente également. Par exemple, le *Legacy Survey of Space and Time* (LSST Science Collaboration, 2009), mené à l'Observatoire Vera C. Rubin, devrait collecter plus de 15 pétaoctets d'images non compressées de l'Univers observable. De telles quantités massives de données sont impossibles à analyser par des humains, c'est pourquoi les astrophysiciens se tournent vers les systèmes d'apprentissage automatique pour examiner les données collectées.

Figure B.1.: L'observatoire Vera C. Rubin Observatory en construction au Chile. Crédits: Rubin Obs./NSF/AURA.

Dans ce contexte, le projet ANR AstroDeep a été mis en place en novembre 2019 pour mener des recherches sur l'utilisation des réseaux neuronaux bayésiens en cosmologie. Les réseaux neuronaux bayésiens (BNN) sont une variante des réseaux neuronaux capables de quantifier les incertitudes provenant de différentes sources, et cela en plus de leurs prédictions, ce qui les rend très adaptés à la tâche à accomplir. Plus précisément, les BNNs séparent l'incertitude aléatoire, inhérente aux données, de l'incertitude épistémique,

liée au manque de connaissance du modèle sur la distribution des données.

Cependant, l'estimation de l'incertitude n'est qu'un aspect de la tâche globale de déconstruction de la « boîte noire » des réseaux neuronaux profonds. La capacité des systèmes d'IA à expliquer leurs prédictions est une autre caractéristique très importante recherchée par les cosmologistes. Disposer d'un réseau neuronal capable d'expliquer ses décisions de manière intelligible, tant qualitativement que quantitativement, peut aider à détecter des motifs ou des biais qui auraient été ignorés autrement.

Plus précisément, la question des « explications contrefactuelles » consiste à demander au modèle quel changement dans l'image aurait entraîné une autre décision. De telles explications pourraient révéler un comportement problématique du réseau neuronal, remettant ainsi en question sa fiabilité. Une problématique intéressante consisterait à étudier comment ces explications pourraient bénéficier d'une approche bayésienne de l'apprentissage profond, afin de les rendre plus intelligibles et plausibles, notamment lorsqu'elles sont appliquées à des données d'images.

La problématique est alors claire : comment pouvons-nous utiliser des approches bayésiennes de l'apprentissage profond pour fournir des estimations d'incertitude calibrées et des explications fiables, dans le contexte des données cosmologiques ?

Répondre à cette question permettra de mettre en lumière des techniques pouvant être appliquées au-delà de l'astrophysique : comme nous l'avons vu, ces caractéristiques sont recherchées dans de nombreux domaines d'application. De plus, l'étude des systèmes d'apprentissage automatique apprenant à partir d'images astronomiques peut également nous aider à comprendre des comportements complexes et à développer des techniques liées à la vision par ordinateur en général.

La suite de ce résumé est organisée comme suit. La Section B.2 détaille les notions d'apprentissage profond bayésien. La Section B.3 applique les réseaux neuronaux bayésiens dans le cadre d'un problème de régression sur des images de galaxies, et les chapitres B.4 et B.5 détaillent l'état de l'art et les travaux autour de l'explicabilité et, plus précisément, les explications contrefactuelles. Enfin la Section B.6 conclut ce résumé détaillé.

B.2. Estimation des incertitudes en apprentissage profond

B.2.1. Incertitudes aléatoires et épistémiques

Le problème consiste à définir une manière de non seulement connaître la prédiction d'un modèle d'apprentissage automatique, mais aussi à quel point il est certain de sa prédiction. Pour résoudre cela, nous devons d'abord définir deux types d'incertitudes : *l'incertitude aléatoire* et *l'incertitude épistémique*.

L'incertitude aléatoire se définit comme celle liée aux données elles-mêmes en raison de facteurs aléatoires influents mais non observés, et dans ce sens, cette incertitude n'est pas réductible. Par exemple, nous pouvons prendre une image et y ajouter lentement du bruit gaussien, et à un certain stade, le modèle arrêtera de prédire de manière fiable le résultat. Peu importe la complexité du modèle ou la taille de l'ensemble de données : si l'information dans les données est insuffisante pour effectuer une prédiction correcte,

cette incertitude aléatoire sera élevée et il n'y aura aucun moyen de la réduire à moins de modifier les données. Le bruit n'est qu'une source possible d'incertitude aléatoire. Une autre façon de voir les choses est de considérer le fait que la relation entre l'espace d'entrée $\mathcal{X}$ et l'espace de sortie $\mathcal{Y}$ est très souvent non déterministe, laissant une incertitude dans la distribution des données $p(x, y)$ même si elle a été parfaitement apprise (Hüllermeier and Waegeman, 2021).

D'autre part, l'incertitude épistémique (ou modèle) représente le manque de connaissances du modèle sur la distribution des données $p(x, y)$. En général, un modèle d'apprentissage automatique montrera une forte incertitude épistémique lorsqu'il traite des données aberrantes. Un exemple simple de cela est de considérer un modèle qui a appris à différencier les images de chats et de chiens, et d'examiner ce qui se passe lorsque soudainement l'image d'entrée est une photo d'une espèce rare de chat, ou une image stylisée d'un chat comme dans un dessin animé ? Ce sont des exemples d'images que le modèle n'a très probablement pas vues lors de l'entraînement. Par conséquent, pour réduire l'incertitude épistémique, on peut rechercher des ensembles de données qui couvrent mieux la distribution $p(x, y)$.

B.2.2. Apprentissage profond bayésien

L'apprentissage profond bayésien est motivé par l'objectif d'estimer non seulement l'incertitude aléatoire d'un modèle d'apprentissage profond, mais aussi son incertitude épistémique. Considérons un réseau neuronal $f_\omega : \mathcal{X} \to \mathcal{Y}$. Dans une configuration non bayésienne, le modèle renvoie une distribution postérieure prédictive ponctuelle $p(y, |, x, \omega)$. Comme nous l'avons vu dans la section précédente, une partie de l'incertitude épistémique provient de l'espace des paramètres Ω. En effectuant une estimation ponctuelle dans Ω, nous excluons l'estimation de cette source d'incertitude sur la distribution prédictive.

Les réseaux neuronaux bayésiens (MacKay, 1992; Neal, 1995), ou BNN (Bayesian Neural Networks), remplacent les estimations ponctuelles des poids ω et supposent plutôt une distribution a priori $p(\omega)$ sur ces derniers. Typiquement, la distribution a priori est supposée être gaussienne. Étant donné l'ensemble de données $\mathcal{D}$, l'objectif est d'estimer la distribution postérieure $p(\omega, |, \mathcal{D})$. Cela nous permettrait d'estimer la distribution prédictive postérieure en marginalisant sur la distribution postérieure

$$p(y \,|\, x, \mathcal{D}) = \int_\Omega p(y \,|\, x, \omega)\, p(\omega \,|\, \mathcal{D})\, d\omega\,. \tag{B.1}$$

Etant donné le théorème de Bayes, la distribution postérieure $p(\omega \,|\, \mathcal{D})$ est reliée à la distribution a priori $p(\omega)$ par la relation

$$p(\omega \,|\, \mathcal{D}) = \frac{p(\mathcal{D} \,|\, \omega)\, p(\omega)}{p(\mathcal{D})}\,. \tag{B.2}$$

Le problème vient du fait que l'équation 3.4 (et B.1) est souvent impossible à calculer analytiquement. Par conséquent, il est nécessaire de trouver une manière d'approximer la distribution postérieure. Cela se fait généralement par une optimisation de Bayes variationnelle pour obtenir une distribution postérieure approximative $q_\theta(\omega)$.

L'une des approximations variationnelles les plus notables est le *MC Dropout*, proposé par Gal and Ghahramani (2016). Le Dropout (Srivastava et al., 2014), initialement proposé comme méthode de régularisation des réseaux neuronaux, ajoute un bruit binaire sur les paramètres du réseau. En fait, le dropout peut être considéré comme un postérieur variationnel $q_\theta(\omega)$ qui applique une distribution de Bernoulli sur les paramètres du réseau:

$$\begin{aligned} W_i &= M_i \cdot \text{diag}\left([r_j^i]_{j=1}^{n_i-1}\right) \\ r_j^i &\sim \mathcal{B}(1-p_i) \quad \forall i = 1, \ldots, L, \forall j = 1, \ldots, n_{i-1}\,, \end{aligned} \tag{B.3}$$

avec $\omega = \{W_i\}_{i=1}^L$ les poids du réseau, et $\theta = \{M_i\}_{i=1}^L$ sont les paramètres variationels de $q_\theta(\omega)$. $p_i \in [0, 1]$ est le taux de Dropout de la i^{eme} couche.

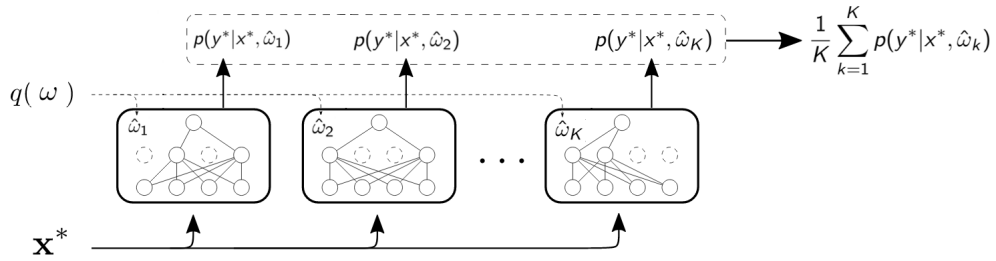


Figure B.2.: Echantillonnage MC Dropout. Crédits: Xiao and Wang (2019).

Etant donné une distribution postérieure variationnelle, on peut obtenir une estimation des incertitudes aléatoires, épistémiques, et prédictives:

$$\hat{\mathcal{U}}_{pred.}(X) := -\sum_{y\in\mathcal{Y}} \left(\frac{1}{M}\sum_{i=1}^{M} p(y\,|\,\hat{\omega}_i, X)\right) \log\left(\frac{1}{M}\sum_{i=1}^{M} p(y\,|\,\hat{\omega}_i, X)\right) \tag{B.4}$$

$$\hat{\mathcal{U}}_{aleat.}(X) := -\frac{1}{M}\sum_{i=1}^{M}\sum_{y\in\mathcal{Y}} p(y\,|\,\hat{\omega}_i, X)\log p(y\,|\,\hat{\omega}_i, X) \tag{B.5}$$

$$\hat{\mathcal{U}}_{epist.}(X) := \hat{\mathcal{U}}_{pred}(X) - \hat{\mathcal{U}}_{aleat}(X)\,, \tag{B.6}$$

B.2.3. Régulation du Dropout dans un réseau neuronal bayésien

Nous proposons une méthode, appelée *Dropout Regulation*, inspirée des boucles d'automatisation et de contrôle. L'objectif est de réguler le taux de dropout en fonction de l'écart de performance entre les données d'entraînement et les données de validation, car le dropout a été initialement conçu comme une technique de régularisation pour éviter le surajustement. Cela signifie que cette méthode est différente des approches basées sur l'optimisation, avec l'avantage d'être très simple à mettre en œuvre et à utiliser. L'idée est que lorsque le modèle présente des performances similaires entre les données d'entraînement et de validation, le dropout n'est pas nécessaire, mais lorsque le modèle

commence à mieux performer sur les données d'entraînement, nous avons besoin d'un taux de dropout plus élevé pour une régularisation plus forte.

Nous utilisons une fonction de contrôle PI qui régule le taux de Dropout via une boucle fermée:

$$u(t) = K_p \left(\epsilon(\omega_t) + \frac{1}{T_i} \sum_{k=1}^{t} \epsilon(\omega_k) \right) , \tag{B.7}$$

Le tableau B.1 montre la précision et la métrique PAvPU pour les différentes méthodes testées. La métrique PAvPU mesure la qualité de l'incertitude et a été évaluée avec un seuil d'incertitude égal à la moyenne de l'incertitude sur l'ensemble de test. La précision de la méthode Regulation est légèrement inférieure à celle de Concrete, mais supérieure à celle de Deterministic et MC Dropout.

Method	Accuracy	PAvPU
Deterministic	0.932	0.7155
MC Dropout	0.928	0.70734
Concrete	**0.935**	0.7575
Regulation	0.934	**0.7579**

Table B.1.: Accuracy and PAvPU metrics for the different methods. Bold indicates the best metric obtained by the method used.

B.3. Régression bayésienne profonde pour l'estimation d'ellipticités de galaxies

B.3.1. Régression d'ellipticités

Dans cette section, nous allons définir l'ellipticité des galaxies et préciser quels types de bruit peuvent corrompre l'ensemble de données.

D'après la théorie astrophysique il est possible d'estimer la déformation cosmique due à des objets célestes massifs en combinant les mesures individuelles de la forme des galaxies. Cette information sur la forme est quantifiée par *l'ellipticité complexe*, définie en cosmologie en tant que:

$$\varepsilon = \varepsilon_1 + \varepsilon_2\, i = \frac{1-q^2}{1+q^2}\, e^{2i\varphi}, \tag{B.8}$$

avec $q = \frac{b}{a}$ le rapport entre petit axe b et grand axe b de l'ellipse, et φ l'angle d'orientation de l'ellipse.

Pour définir l'incertitude aléatoire, nous devons modéliser l'ellipticité complexe comme une variable aléatoire suivant une distribution normale multivariée (MVN) : $p(\varepsilon, |, X, \omega) = \mathcal{N}(\mu_\omega(X), \Sigma_\omega(X))$, avec $f_\omega(X) = (\mu_\omega(X), \Sigma_\omega(X))$. Le support de cette distribution est $\mathbb{R}^2$ car nous considérons l'ellipticité complexe comme un couple $(\varepsilon_1, \varepsilon_2) \in \mathbb{R}^2$, composé

à la fois de la partie réelle et de la partie imaginaire de ε. Cette MVN nous permet de modéliser l'impact du bruit lié aux données sur la prédiction de l'ellipticité complexe. f_ω, le réseau neuronal, produira les paramètres de la MVN et sera entraîné en maximisant la log-vraisemblance de la MVN, ou de manière équivalente en minimisant le risque empirique associé à la fonction de perte suivante :

$$\mathcal{L}(\varepsilon, f_\omega(X)) := \det \Sigma_\omega(X) + (\varepsilon - \mu_\omega(X))^T \Sigma_\omega(X)^{-1} (\varepsilon - \mu_\omega(X)), . \tag{B.9}$$

Les incertitudes prédictives, aléatoires, et épistémiques sont alors définies par

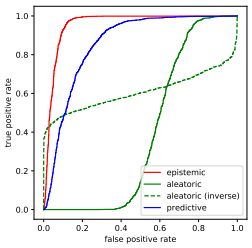
$$\Sigma_{aleat.}(X, \mathcal{D}) := \mathbb{E}_{p(\omega \mid \mathcal{D})} \left[\mathrm{Cov}(\varepsilon \mid X, \omega) \right] \tag{B.10}$$

$$\Sigma_{epist.}(X, \mathcal{D}) := \mathrm{Cov}_{p(\omega \mid \mathcal{D})} \left[\mathbb{E}(\varepsilon \mid X, \omega) \right] \tag{B.11}$$

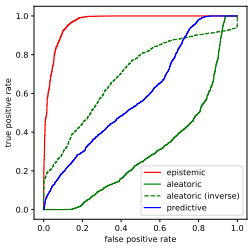
$$\Sigma_{pred.}(X, \mathcal{D}) = \Sigma_{aleat.}(X, \mathcal{D}) + \Sigma_{epist.}(X, \mathcal{D}) . \tag{B.12}$$

B.3.2. Résultats

Pour quantifier la qualité de l'incertitude épistémique pour la détection des prédictions incohérentes dues aux valeurs aberrantes, nous avons calculé les courbes ROC pour chaque type d'incertitude. Plus précisément, nous réduisons chaque matrice de covariance (aleatorique, épistémique et prédictive) à un scalaire en calculant son déterminant. Nous interprétons ces estimations comme une fonction de score pour évaluer si une image est une valeur aberrante, c'est-à-dire une image de galaxies superposées : plus le score est élevé, plus il est probable que l'image contienne une superposition de galaxies. Enfin, nous calculons pour chaque fonction de score sa courbe ROC. Nous répétons ce processus à la fois pour les réseaux entraînés avec des données bruitées et sans bruit. Les résultats sont présentés dans la figure B.3. Nous rajoutons à cela les valeurs d'aire sous la courbe ROC (AUC) dans la table B.2.



(a) Courbe ROC, modèle sans bruit



(b) Courbe ROC, modèle avec bruit

Figure B.3.: Courbes ROC pour toutes les incertitudes

Incertitude	AUC sans bruit	AUC bruit
Epistemique	**0.956**	**0.969**
Aléatoire	0.394	0.306
Aléatoire (inverse)	0.606	0.694
Prédictive	0.856	0.594

Table B.2.: Valeurs AUC pour toutes les incertitudes

B.4. Explicabilité et interprétabilité en apprentissage profond

Les *explications contrefactuelles* (Wachter et al., 2018; Keane et al., 2021) sont une méthode d'explication basée sur des exemples, sous la forme « Si x' avait eu lieu à la place de x, alors y' se serait produit à la place de y ». Plus précisément, les explications contrefactuelles visent à générer des exemples en analysant les « changements actionables minimaux » dans les attributs d'entrée de telle sorte que l'instance modifiée x' ait une étiquette différente $y' \neq y$ selon la distribution des données $p(x, y)$. x' est alors appelé un « exemple contrefactuel » de x. L'étiquette y' est, dans la plupart des algorithmes d'explication contrefactuelle, donnée en entrée pour guider le processus d'explication vers cette classe particulière. Dans ce cas, y' est appelée la « classe cible » de l'exemple contrefactuel.

Les explications contrefactuelles visent des changements « minimaux » au sens où l'exemple contrefactuel x' doit être proche de x selon une certaine définition (par exemple, une métrique de distance). Elles doivent également être « actionnables » au sens où il serait raisonnable de réaliser les changements pour passer de x à x'. Un corollaire de l'actionnalité est le « réalisme » (ou la « plausibilité ») : un exemple contrefactuel (x', y') doit être probable selon la distribution des données $p(x, y)$.

Parmi les méthodes d'explications contrefactuelles, on compte notamment celles basées sur un processus d'optimisation via descente de gradient.

Étant donné un modèle f_ω paramétré par $\omega \in \Omega$, une instance $x \in \mathbb{R}^n$ et une étiquette cible $y' \in \mathbb{R}^m$, Wachter et al. (2018) a proposé la fonction objective suivante pour rechercher des exemples contrefactuels :

$$x_{CF} := \underset{x' \in \mathcal{X}}{\operatorname{argmin}} \left[\mathcal{L}(f_\omega(x'), y') + \lambda d(x, x') \right], \tag{B.13}$$

avec $\lambda \geq 0$ et $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ une distance définie sur $\mathbb{R}^n$. Le terme de distance garantit la propriété de minimalité des exemples contrefactuels en pénalisant les exemples contrefactuels qui sont trop éloignés de l'instance originale x.

Joshi et al. (2019) ont quant à eux utilisé un modèle génératif $\mathcal{G}\theta(z)$ pour rechercher directement un exemple contrefactuel dans un espace latent $\mathcal{Z}$. Ils supposent que leur modèle génératif est capable d'encoder l'espace des instances $\mathcal{X}$ dans l'espace latent, c'est-à-dire une fonction $\mathcal{F}_\psi : \mathcal{X} \to \mathcal{Z}$ paramétrée par ψ. Des exemples de modèles génératifs capables d'encoder des instances sont les VAE (Kingma and Welling, 2014)

ou même de nouvelles architectures GAN, telles que VAE-GAN (Larsen et al., 2016), AttGAN (He et al., 2017) ou BiGAN (Donahue et al., 2017).

Leur méthode, appelée REVISE, utilise la fonction objectif suivante :

$$x_{CF} := \mathcal{G}_\theta(z_{CF}), \tag{B.14}$$

$$z_{CF} := \underset{z \in \mathcal{Z}}{\operatorname{argmin}} \left[\mathcal{L}(f_\omega(\mathcal{G}_\theta(z)), y') + \lambda d(x, \mathcal{G}_\theta(z))\right], , \tag{B.15}$$

avec $\lambda \geq 0$ et $z = \mathcal{F}_\psi(x)$. En utilisant un espace latent, l'espace de recherche est alors restreint à un espace latent contraint qui a été appris par le modèle génératif pour représenter plus fidèlement la distribution des données. Par conséquent, les contre-exemples obtenus seront plus réalistes, car une variété de données non contrainte peut donner des explications peu probables ou irréalistes.

B.5. Clarity: explications contrefactuelles avec un ensemble de classifieurs latents

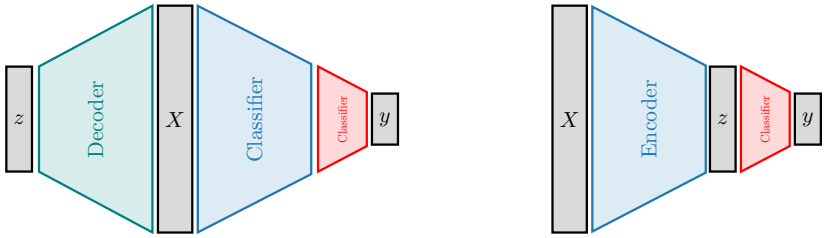
B.5.1. Présentation

Nous présentons dans cette section notre contribution dans le domaine des explications contrefactuelles, une méthode appellée *Clarity*.

De manière similaire à REVISE-E, nous utiliserons l'espace latent $\mathcal{Z}$ d'un modèle génératif comme espace de recherche. Dans ce cas, nous considérerons que le modèle génératif est un VAE. La principale différence est que le classifieur sera entraîné dans cet espace latent préalablement défini et appris, plutôt que dans l'espace des instances $\mathcal{X}$. Cela signifie qu'au lieu d'avoir une fonction définie comme $f_\omega : \mathcal{X} \to \mathbb{R}^n$, nous avons plutôt $f_\omega : \mathcal{Z} \to \mathbb{R}^n$. En ayant un classifieur directement entraîné dans l'espace latent, nous devrions être en mesure de remarquer des différences dans la génération des explications contrefactuelles dans cet espace.

Dans la figure B.4, nous montrons la principale différence d'architecture entre REVISE-E et un classifieur dans l'espace latent. Dans REVISE-E, le classifieur est déjà entraîné dans l'espace latent, en utilisant tous les paramètres possibles pour optimiser la classification des instances en fonction des étiquettes (blocs bleus et rouges dans la figure B.4a). Ensuite, la recherche contrefactuelle est effectuée dans un espace latent à l'aide d'un décodeur pour traduire l'encodage latent en une instance X (bloc vert). Notre approche est différente dans le sens que, tout en conservant la même architecture et le même nombre de paramètres, la première partie correspond en réalité à l'encodeur du VAE, et seules les dernières couches après l'encodage latent sont entraînées pour la tâche de classification (voir figure B.4b).

Clarity utilise un ensemble de classifieurs entraînés dans l'espace latent, tel que la nouvelle fonction objectif est définie par



(a) REVISE architecture (b) Latent space classifier

Figure B.4.: Différence architecturale entre Clarity et REVISE.

$$x_{CF} := \mathcal{G}_\theta(z_{CF})\,, \tag{B.16}$$

$$z_{CF} := \underset{z' \in \mathcal{Z}}{\operatorname{argmin}} \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}(f_m(z'), y') + \lambda\, d(z, z'), \tag{B.17}$$

Notons ici que la distance est définie dans l'espace latent, et représente donc plutôt une sorte de distance "sémantique" entre z et z'.

B.5.2. Résultats expérimentaux

Nous avons voulu valider *Clarity* sur un ensemble de données "réaliste" et détaillé. Nous avons sélectionné l'ensemble de données CelebA (Liu et al., 2015), qui est composé de photos en RGB de visages de célébrités. Ces photos sont redimensionnées à une résolution de $3 \times 64 \times 64$ pixels. Le tableau B.3 montre les résultats sur cet ensemble de données. Nous avons utilisé un β-VAE (Higgins et al., 2017) avec la même architecture proposée par Subramanian (2020). L'espace latent lui-même est de dimension 128. Pour comparer *Clarity* à REVISE, nous avons choisi de rajouter un ensemble de classifieurs à REVISE. Nous appellerons cette modification "REVISE-E". Pour REVISE-E et Clarity, nous avons utilisé un ensemble de 20 modèles avec un entraînement adversarial. Il existe deux versions de REVISE-E : R-E (image) qui utilise une pénalité sur l'espace des images $d(\mathcal{G}_\theta(z'), X)$, et R-E (latent) qui utilise une pénalité sur l'espace latent $d(z', z)$, de manière similaire à Clarity. Les classifieurs ont été entraînés pour prédire la couleur des cheveux. L'objectif de ces explications est de la changer sans modifier les autres caractéristiques. Globalement, les deux méthodes offrent des résultats similaires, mais Clarity semble plus proche de l'image d'origine, contrairement à REVISE-E qui modifie le genre ou le visage de la personne, leur couleur de peau ou ajoute des attributs inutiles comme le maquillage.

B.6. Conclusion

Dans cette thèse, nous avons concentré notre travail sur la problématique du développement de moyens pour rendre les réseaux neuronaux plus transparents dans le contexte de l'analyse des données astronomiques. Afin d'atteindre cet objectif, deux idées principales

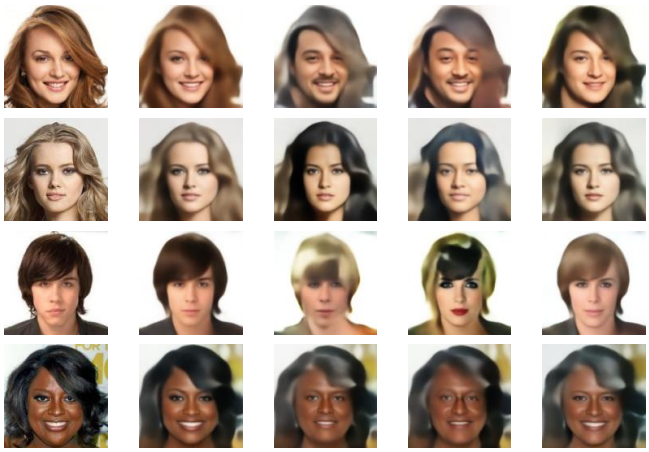
Original	VAE	R-E (latent)	R-E (image)	Clarity
				

Table B.3.: Explications contrefactuelles sur la couleur des cheveux sur le jeu de données CelebA

ont été proposées. La première consistait à utiliser des réseaux neuronaux bayésiens comme moyen d'estimer deux sources distinctes d'incertitude, à savoir l'incertitude épistémique et l'incertitude aléatoire. Ensuite, nous avons étudié comment l'apprentissage profond bayésien peut améliorer l'explicabilité des réseaux neuronaux profonds, et comment les connaissances sur les incertitudes peuvent être ajoutées à d'autres techniques pour améliorer davantage la qualité et l'interprétabilité des explications.

Dans l'ensemble, les recherches réalisées dans cette thèse ont abouti au développement de nouvelles méthodes et techniques pour améliorer la transparence des réseaux neuronaux dans le contexte des données astronomiques. Les connaissances supplémentaires apportées par ces contributions éclairent de nouvelles perspectives en matière d'explicabilité et d'estimation d'incertitude des réseaux neuronaux en général, notamment dans le contexte du traitement d'images et de la vision par ordinateur, et nous pensons qu'elles seront utiles dans de nombreux domaines d'application. Cependant, ce champ de recherche est vaste et nous pensons qu'il reste encore de nombreuses pratiques, méthodes et comportements à découvrir et à étudier.

Bibliography

Apley, D. W. and Zhu, J. (2016). Visualizing the effects of predictor variables in black box supervised learning models.

Araujo, A., Norris, W., and Sim, J. (2019). Computing receptive fields of convolutional neural networks. *Distill*. https://distill.pub/2019/computing-receptive-fields.

Arcelin, B., Doux, C., Aubourg, E., and Roucelle, C. (2020). Deblending galaxies with variational autoencoders: A joint multiband, multi-instrument approach. *Monthly Notices of the Royal Astronomical Society*, 500(1):531–547.

Arrieta, A. B. and Ser, J. D. (2020). Plausible counterfactuals: Auditing deep learning classifiers with realistic adversarial examples. In *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*, pages 1–7. IEEE.

Bertin, E. and Arnouts, S. (1996). Sextractor: Software for source extraction. *Astron. Astrophys. Suppl. Ser.*, 117(2):393–404.

Boluki, S., Ardywibowo, R., Dadaneh, S. Z., Zhou, M., and Qian, X. (2020). Learnable bernoulli dropout for bayesian deep learning. In Chiappa, S. and Calandra, R., editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3905–3916. PMLR.

Boreiko, V., Augustin, M., Croce, F., Berens, P., and Hein, M. (2022). Sparse visual counterfactual explanations in image space. In Andres, B., Bernard, F., Cremers, D., Frintrop, S., Goldlücke, B., and Ihrke, I., editors, *Pattern Recognition - 44th DAGM German Conference, DAGM GCPR 2022, Konstanz, Germany, September 27-30, 2022, Proceedings*, volume 13485 of *Lecture Notes in Computer Science*, pages 133–148. Springer.

Bosch, J., Armstrong, R., Bickerton, S., Furusawa, H., Ikeda, H., Koike, M., Lupton, R., Mineo, S., Price, P., Takata, T., Tanaka, M., Yasuda, N., AlSayyad, Y., Becker, A. C., Coulton, W., Coupon, J., Garmilla, J., Huang, S., Krughoff, K. S., Lang, D., Leauthaud, A., Lim, K.-T., Lust, N. B., MacArthur, L. A., Mandelbaum, R., Miyatake, H., Miyazaki, S., Murata, R., More, S., Okura, Y., Owen, R., Swinbank, J. D., Strauss, M. A., Yamada, Y., and Yamanoi, H. (2017). The hyper suprime-cam software pipeline. *Publications of the Astronomical Society of Japan*, 70(SP1).

Craven, M. and Shavlik, J. (1995). Extracting tree-structured representations of trained networks. In Touretzky, D., Mozer, M., and Hasselmo, M., editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press.

Cybenko, G. (1992). Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.*, 5(4):455.

Dai, B. and Wipf, D. P. (2019). Diagnosing and enhancing VAE models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Dash, S. and Goncalves, J. (2021). Lprules: Rule induction in knowledge graphs using linear programming. *CoRR*, abs/2110.08245.

Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., and Udluft, S. (2018). Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In Dy, J. G. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1192–1201. PMLR.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Dhariwal, P. and Nichol, A. Q. (2021). Diffusion models beat gans on image synthesis. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 8780–8794.

Dhurandhar, A., Chen, P., Luss, R., Tu, C., Ting, P., Shanmugam, K., and Das, P. (2018). Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 590–601.

Dieleman, S., Willett, K. W., and Dambre, J. (2015). Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, 450(2):1441–1459.

Dinh, L., Krueger, D., and Bengio, Y. (2015). NICE: non-linear independent components estimation. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on*

Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings.

Donahue, J., Krähenbühl, P., and Darrell, T. (2017). Adversarial feature learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Downs, M., Chu, J., Yacoby, Y., Doshi-Velez, F., and WeiWei, P. (2020). Cruds: Counterfactual recourse using disentangled subspaces. *ICML Workshop on Human Interpretability in Machine Learning*, pages 1–23.

Duchi, J. C., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159.

Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *ArXiv e-prints*.

ESA (2013). Planck reveals an almost perfect universe.

Fort, S., Hu, H., and Lakshminarayanan, B. (2019). Deep ensembles: A loss landscape perspective. *CoRR*, abs/1912.02757.

Freiesleben, T. (2022). The intriguing relation between counterfactual explanations and adversarial examples. *Minds Mach.*, 32(1):77–109.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202.

Gal, Y. (2016). Uncertainty in deep learning.

Gal, Y. and Ghahramani, Z. (2015). Bayesian convolutional neural networks with bernoulli approximate variational inference. *CoRR*, abs/1506.02158.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1050–1059. JMLR.org.

Gal, Y., Hron, J., and Kendall, A. (2017a). Concrete dropout. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Gal, Y., Islam, R., and Ghahramani, Z. (2017b). Deep bayesian active learning with image data. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1183–1192. PMLR.

Ghosh, B., Malioutov, D., and Meel, K. S. (2022). Efficient learning of interpretable classification rules. *J. Artif. Intell. Res.*, 74:1823–1863.

Glynn, P. W. (1990). Likelihood ratio gradient estimation for stochastic systems. *Commun. ACM*, 33(10):75–84.

Goldstein, A., Kapelner, A., Bleich, J., and Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A. C., and Bengio, Y. (2013). Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1319–1327. JMLR.org.

Graves, A. (2013). Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Graves, A., Mohamed, A., and Hinton, G. E. (2013). Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778.

Guynn, J. (2015). Google photos labeled black people 'gorillas'. *USA Today*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International*

Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, pages 1026–1034. IEEE Computer Society.

He, Z., Zuo, W., Kan, M., Shan, S., and Chen, X. (2017). Arbitrary facial attribute editing: Only change what you want. *CoRR*, abs/1711.10678.

Higgins, I., Matthey, L., Pal, A., Burgess, C. P., Glorot, X., Botvinick, M. M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

Houlsby, N., Huszar, F., Ghahramani, Z., and Lengyel, M. (2011). Bayesian active learning for classification and preference learning. *CoRR*, abs/1112.5745.

Hron, J., de G. Matthews, A. G., and Ghahramani, Z. (2018). Variational bayesian dropout: pitfalls and fixes. In Dy, J. G. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2024–2033. PMLR.

Hüllermeier, E. and Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Mach. Learn.*, 110(3):457–506.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. R. and Blei, D. M., editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org.

Islam, S. R., Eberle, W., Ghafoor, S. K., and Ahmed, M. (2021). Explainable artificial intelligence approaches: A survey. *CoRR*, abs/2101.09429.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Mach. Learn.*, 37(2):183–233.

Joshi, S., Koyejo, O., Vijitbenjaronk, W., Kim, B., and Ghosh, J. (2019). Towards realistic individual recourse and actionable explanations in black-box decision making systems. *CoRR*, abs/1907.09615.

Kaggle (2013). Dogs vs. cats. https://www.kaggle.com/c/dogs-vs-cats.

Kaiser, N., Squires, G., and Broadhurst, T. (1995). A Method for Weak Lensing Observations. *The Astrophysical Journal*, 449:460.

Kaufman, L. and Rousseeuw, P. (1990). *Partitioning Around Medoids (Program PAM)*, chapter 2, pages 68–125. John Wiley & Sons, Ltd.

Keane, M. T., Kenny, E. M., Delaney, E., and Smyth, B. (2021). If only we had better counterfactual explanations: Five key deficits to rectify in the evaluation of counterfactual XAI techniques. In Zhou, Z., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4466–4474. ijcai.org.

Keane, M. T. and Smyth, B. (2020). Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable AI (XAI). In Watson, I. and Weber, R. O., editors, *Case-Based Reasoning Research and Development - 28th International Conference, ICCBR 2020, Salamanca, Spain, June 8-12, 2020, Proceedings*, volume 12311 of *Lecture Notes in Computer Science*, pages 163–178. Springer.

Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Kilbinger, M. (2015). Cosmology with cosmic shear observations: a review. *Reports on Progress in Physics*, 78(8):086901.

Kim, B., Khanna, R., and Koyejo, O. O. (2016). Examples are not enough, learn to criticize! criticism for interpretability. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Bartlett, P. L., Pereira, F. C. N., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6402–6413.

Lang, O., Gandelsman, Y., Yarom, M., Wald, Y., Elidan, G., Hassidim, A., Freeman, W. T., Isola, P., Globerson, A., Irani, M., and Mosseri, I. (2021). Explaining in style: Training a GAN to explain a classifier in stylespace. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 673–682. IEEE.

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O. (2016). Autoencoding beyond pixels using a learned similarity metric. In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1558–1566. JMLR.org.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551.

LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

LeNail, A. (2019). Nn-svg: Publication-ready neural network architecture schematics. *Journal of Open Source Software*, 4(33):747.

Li, M., Zuo, W., and Zhang, D. (2016). Convolutional network for attribute-driven and identity-preserving human face generation. *CoRR*, abs/1608.06434.

Lintott, C. J., Schawinski, K., Slosar, A., Land, K., Bamford, S., Thomas, D., Raddick, M. J., Nichol, R. C., Szalay, A., Andreescu, D., Murray, P., and Vandenberg, J. (2008). Galaxy Zoo: morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey*. *Monthly Notices of the Royal Astronomical Society*, 389(3):1179–1189.

Liu, S., Kailkhura, B., Loveland, D., and Han, Y. (2019). Generative counterfactual introspection for explainable deep learning. In *2019 IEEE Global Conference on Signal*

and Information Processing, GlobalSIP 2019, Ottawa, ON, Canada, November 11-14, 2019, pages 1–5. IEEE.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Loaiza-Ganem, G. and Cunningham, J. P. (2019). The continuous bernoulli: fixing a pervasive error in variational autoencoders. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13266–13276.

Looveren, A. V. and Klaise, J. (2021). Interpretable counterfactual explanations guided by prototypes. In Oliver, N., Pérez-Cruz, F., Kramer, S., Read, J., and Lozano, J. A., editors, *Machine Learning and Knowledge Discovery in Databases. Research Track - European Conference, ECML PKDD 2021, Bilbao, Spain, September 13-17, 2021, Proceedings, Part II*, volume 12976 of *Lecture Notes in Computer Science*, pages 650–665. Springer.

LSST Science Collaboration (2009). LSST Science Book, version 2.0.

Lundberg, S. M. and Lee, S. (2017). A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874.

MacKay, D. J. C. (1992). A practical bayesian framework for backpropagation networks. *Neural Comput.*, 4(3):448–472.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2017). The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Mandelbaum, R. (2018). Weak lensing for precision cosmology. *Annual Review of Astronomy and Astrophysics*, 56(1):393–433.

Mandelbaum, R., Rowe, B., Bosch, J., Chang, C., Courbin, F., Gill, M., Jarvis, M., Kannawadi, A., Kacprzak, T., Lackner, C., Leauthaud, A., Miyatake, H., Nakajima, R., Rhodes, J., Simet, M., Zuntz, J., Armstrong, B., Bridle, S., Coupon, J., Dietrich, J. P., Gentile, M., Heymans, C., Jurling, A. S., Kent, S. M., Kirkby, D., Margala, D., Massey, R., Melchior, P., Peterson, J., Roodman, A., and Schrabback, T. (2014). The third gravitational lensing accuracy testing (great3) challenge handbook. *The Astrophysical Journal Supplement Series*, 212(1):5.

Melchior, P., Moolekamp, F., Jerdee, M., Armstrong, R., Sun, A.-L., Bosch, J., and Lupton, R. (2018a). scarlet: Source separation in multi-band images by constrained matrix factorization. *Astronomy and Computing*, 24:129–142.

Melchior, P., Moolekamp, F., Jerdee, M., Armstrong, R., Sun, A.-L., Bosch, J., and Lupton, R. (2018b). Scarlet: Source separation in multi-band images by constrained matrix factorization. *Astronomy and Computing*, 24:129–142.

Mukhoti, J. and Gal, Y. (2018). Evaluating bayesian deep learning methods for semantic segmentation. *CoRR*, abs/1811.12709.

Neal, R. M. (1995). *Bayesian learning for neural networks*. PhD thesis, University of Toronto, Canada.

Nelder, J. A. and Wedderburn, R. W. M. (1972). Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370–384.

NHTSA (2017). Tesla crash preliminary evaluation report. *Technical report, U.S. Department of Transportation, National Highway Traffic Safety Administration, Jan 2017*.

Paisley, J. W., Blei, D. M., and Jordan, M. I. (2012). Variational bayesian inference with stochastic search. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress.

Papernot, N., McDaniel, P. D., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016). The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 372–387. IEEE.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with CLIP latents. *CoRR*, abs/2204.06125.

Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows. In Bach, F. R. and Blei, D. M., editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1530–1538. JMLR.org.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. In Krishnapuram, B., Shah, M., Smola, A. J., Aggarwal, C. C., Shen, D., and Rastogi, R., editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144. ACM.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 10674–10685. IEEE.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Rowe, B., Jarvis, M., Mandelbaum, R., Bernstein, G. M., Bosch, J., Simet, M., Meyers, J. E., Kacprzak, T., Nakajima, R., Zuntz, J., Miyatake, H., Dietrich, J. P., Armstrong, R., Melchior, P., and Gill, M. S. S. (2014). Galsim: The modular galaxy image simulation toolkit. https://github.com/GalSim-developers/GalSim.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Sanchez, J., Mendoza, I., Kirkby, D. P., and Burchat, P. R. (2021). Effects of overlapping sources on cosmic shear estimation: Statistical sensitivity and pixel-noise bias. *Journal of Cosmology and Astroparticle Physics*, 2021(07):043.

Sanchez, P. and Tsaftaris, S. A. (2022). Diffusion causal models for counterfactual estimation. In Schölkopf, B., Uhler, C., and Zhang, K., editors, *1st Conference on Causal Learning and Reasoning, CLeaR 2022, Sequoia Conference Center, Eureka, CA, USA, 11-13 April, 2022*, volume 177 of *Proceedings of Machine Learning Research*, pages 647–668. PMLR.

Schut, L., Key, O., McGrath, R., Costabello, L., Sacaleanu, B., Corcoran, M., and Gal, Y. (2021). Generating interpretable counterfactual explanations by implicit minimisation of epistemic and aleatoric uncertainties. In Banerjee, A. and Fukumizu, K., editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 1756–1764. PMLR.

Shaker, M. H. and Hüllermeier, E. (2020). Aleatoric and epistemic uncertainty with random forests. In Berthold, M. R., Feelders, A., and Krempl, G., editors, *Advances in Intelligent Data Analysis XVIII - 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27-29, 2020, Proceedings*, volume 12080 of *Lecture Notes in Computer Science*, pages 444–456. Springer.

Shapley, L. S. (1953). A value for n-person games. In Kuhn, H. W. and Tucker, A. W., editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Smith, L. and Gal, Y. (2018). Understanding measures of uncertainty for adversarial example detection. In Globerson, A. and Silva, R., editors, *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 560–569. AUAI Press.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In Bach, F. R. and

Blei, D. M., editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2256–2265. JMLR.org.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Subramanian, A. (2020). Pytorch-vae. https://github.com/AntixK/PyTorch-VAE.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2014). Intriguing properties of neural networks. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Theobald, C., Arcelin, B., Pennerath, F., Conan-Guez, B., Couceiro, M., and Napoli, A. (2021). A bayesian convolutional neural network for robust galaxy ellipticity regression. In Dong, Y., Kourtellis, N., Hammer, B., and Lozano, J. A., editors, *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track - European Conference, ECML PKDD 2021, Bilbao, Spain, September 13-17, 2021, Proceedings, Part V*, volume 12979 of *Lecture Notes in Computer Science*, pages 135–150. Springer.

Theobald, C., Pennerath, F., Conan-Guez, B., Couceiro, M., and Napoli, A. (2020). A bayesian neural network based on dropout regulation. In Hüllermeier, E. and Destercke, S., editors, *Workshop on Uncertainty in Machine Learning (WUML) at ECML-PKDD 2020 Conference, N.A. (online)*.

Theobald, C., Pennerath, F., Conan-Guez, B., Couceiro, M., and Napoli, A. (2022). Clarity: an improved gradient method for producing quality visual counterfactual explanations. *CoRR*, abs/2211.15370.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning. Technical report.*

Tolstikhin, I. O., Bousquet, O., Gelly, S., and Schölkopf, B. (2018). Wasserstein auto-encoders. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Wachter, S., Mittelstadt, B., and Russell, C. (2018). Counterfactual explanations without opening the black box: automated decisions and the gdpr. *Harvard Journal of Law and Technology*, 31(2):841–887.

Walmsley, M., Smith, L., Lintott, C., Gal, Y., Bamford, S., Dickinson, H., Fortson, L., Kruk, S., Masters, K., Scarlata, C., Simmons, B., Smethurst, R., and Wright, D. (2019). Galaxy zoo: probabilistic morphology through bayesian CNNs and active learning. *Monthly Notices of the Royal Astronomical Society*, 491(2):1554–1574.

Wyden, B. (2019). Algorithm accountability act of 2019. https://www.wyden.senate.gov/imo/media/doc/Algorithmic%20Accountability%20Act%20of%202019%20Bill%20Text.pdf.

Xiao, Y. and Wang, W. Y. (2019). Quantifying uncertainties in natural language processing tasks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7322–7329. AAAI Press.

Yin, M. and Zhou, M. (2019). ARM: augment-reinforce-merge gradient for stochastic binary networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yu, J., Ignatiev, A., Stuckey, P. J., and Bodic, P. L. (2021). Learning optimal decision sets and lists with SAT. *J. Artif. Intell. Res.*, 72:1251–1279.

Zhao, Y. (2020). Fast real-time counterfactual explanations. *CoRR*, abs/2007.05684.