POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Politecnico di Torino

EPFL

POLITECNICO MILANO 1863

Master's Degree Thesis

On the Impact of Adversarial Training on Uncertainty Estimation and Uncertainty Targeted Attacks

Supervisors

Prof. Barbara CAPUTO

Prof. Martin JAGGI

Prof. Matteo MATTEUCCI

Advisors

Tatjana CHAVDAROVA

Matteo PAGLIARDINI

Candidate

Gilberto MANUNZA

October 2021

Summary

State of the art deep learning models, despite being successful in many applications, have the problem of being sensitive to small perturbations in the input data. These perturbations can be crafted by an adversary in order to "fool" a neural network into making wrong predictions. This problem raises many reliability and security concerns about the deployment of deep learning models in real world applications. Adversarial training methods aim at improving the robustness of the model to such attacks, but many of them—including state of the art techniques like Projected Gradient Descent (PGD)—often lead to networks with lower unperturbed (clean) accuracy. Additionally, some fast adversarial training techniques, e.g. the Fast Gradient Sign Method (FGSM), suffer from a problem called catastrophic overfitting, which occurs when a model becomes very robust to a particular adversarial attack used during training, but can not generalize to others. Starting from these considerations and building on the notions of uncertainty estimation techniques the aforementioned problems will be tackled by introducing a novel class of adversarial attacks that instead of having the goal of fooling the network, aim at maximizing its uncertainty. These attacks will be extensively analyzed in various settings and under several uncertainty estimation frameworks, such as Bayesian Neural Networks (BNNs), Monte Carlo Dropout (MCD), and a Gaussian Processes based method called Deterministic Uncertainty Estimation (DUE). It will be shown, using the *MNIST*, *FashionMNIST* and the *CIFAR*-10 datasets, how this approach, implemented both in the image and in the latent space of a neural network, does not deteriorate the clean accuracy of the model, is robust to catastrophic overfitting and to PGD attacks.

Acknowledgements

I would first like to thank Professor Martin Jaggi, Tatjana Chavdarova and Matteo Pagliardini that supervised me throughout this project at EPFL and gave me lots of incredibly helpful advice, knowledge and support.

I spent some time at the Machine Learning and Optimization laboratory during which I learned a lot of new research topics and I enriched my skills. I would like to thank all the people that I met there in person and remotely. It has been a fantastic experience that I will never forget.

I would also like to thank my supervisor at Politecnico di Torino, Professor Barbara Caputo and my supervisor at Politecnico di Milano, Professor Matteo Matteucci for the help given in this Alta Scuola Politecnica double degree program.

I would like to acknowledge the student association IEEE-Eta Kappa Nu, in particular the Mu Nu Chapter and the guys of the board that helped me during the lockdown months. It has been great spending some time with you and I thank you for all the stimulating discussions and fun that we had.

Furthermore I would like to thank my parents and my grandparents that helped me in several ways, including financially, throughout this journey. I could have not completed this thesis and university course without the support of my girlfriend Carlotta, that has always been together with me in good and bad times.

Additionally I would like to thank all my relatives in Turin that helped me, especially in the first years as non-resident student, making me feel like at home.

Last but not least I would like to thank all of my friends that helped me with happy distractions during my academic studies.

Table of Contents

List of Tables

List of Figures

Acronyms

AT
Adversarial Training

FC
Fully Connected

NN
Neural Networks

DNN
Deep Neural Networks

FGSM
Fast Gradient Sign Method

R-FGSM
Random start - FGSM

PGD
Projected Gradient Descent

VAE
Variational Autoencoder

GAN
Generative Adversarial Network

CO
Catastrophic Overfitting

MLE
Maximum Likelihood Estimation

BNN
Bayesian Neural Network

RV
Random Variable

VI
Variational Inference

MC
Monte Carlo

GP
Gaussian Processes

MCMC
Markov Chain Monte Carlo

KL
Kullback–Leibler

ELBO
Evidential Lower Bound

MI
Mutual Information

RBF
Radial Basis Function

MLP
Multi Layer Perceptron

DUM
Deterministic Uncertainty Methods

DUE

Deterministic Uncertainty Estimation

SNGP

Spectral-normalized Neural Gaussian Process

DKL

Deep Kernel Learning

ReLU

Rectified Linear Unit

UTA

Uncertainty Targeted Attacks

SGD

Stochastic Gradient Descent

WRN

Wide ResNet

Chapter 1

Introduction

Deep Learning techniques are nowadays used to solve problems with performance levels that were unthinkable several years ago. For example these methods had a great success in visual tasks, like image classification [1] [2], segmentation [3] and clustering [4].

However, deep learning models can be easily fooled by *adversaries* that are able to craft ad hoc samples with the goal of making the predictions of the network wrong [5][6]. In the particular case of image classification, it has been shown that with a relatively small effort it is possible to create some adversarial images that, given in input to a network with high accuracy on clean (non adversarial) data, lead to very poor output predictions. These samples appear practically *identical* to the original ones, but are able to fool a state of the art deep model. This is usually done by adding very small *perturbations* to the input images.

The existence of adversarial samples leads to a wide variety of problems in the fields of security, reliability, and interpretability of neural networks. Several solutions have been proposed as defenses to adversarial attacks [7], in general, they can be divided in two categories:

- *Adversarial Training.* This approach consists in including in the training samples the perturbed adversarial ones (see Section 2.2). In this way, it is possible to make the network more robust. Unfortunately most of the times this robustness comes at the expenses of the *clean accuracy*, i.e. the accuracy measured on non perturbed samples. Furthermore, when doing adversarial training with attacks that are fast to compute, like the Fast Gradient Sign Methods (FGSM) [5] it is possible to run into the problem of Catastrophic Overfitting (CO) [8], that happens when the network becomes very accurate in classifying samples perturbed with the attack that has been used during training, but is not able to generalize to other, stronger attacks during testing.
- *Adversarial Detection.* This strategy on the other hand consists in finding

ways to detect adversarial samples in order remove them from the dataset. Detection methods aim at *discriminating adversarial samples* from clean ones.

In this thesis, we focus on the former, namely on Adversarial Training (AT).

A separate line of work focuses on the *estimation of the uncertainty* associated with the predictions of a model [9]. In the case of image classification the most common framework for Neural Networks (NNs) consists in providing a Softmax output, i.e. for each class an estimation of the probability of a sample to belong to that class. This output is only a point estimate of such probability, but in a lot of different settings it could be useful to also know how uncertain the network is about a particular prediction. Several methods exist in order to give the network this ability. Some of them are *Bayesian Methods* that make each parameter of the model a random variable; in this way outputs are stochastic and based on that it is possible to compute the uncertainty of predictions [10]. This type of network is known as *Bayesian Neural Network* (BNN) and in general, it can not be trained directly because of *intractability* problems (see Section 2.3.1). A lot of strategies have been proposed for approximating a BNN using techniques like *Variational Inference* (VI) [10], *Monte Carlo* (MC) [11] sampling or *Monte Carlo Dropout* [12] (see Section 2.3.2). Furthermore, recently some methods have emerged to compute the uncertainty of a neural network in a *deterministic* way using a single forward pass and by exploiting techniques like, for instance, *Gaussian Processes* (GP) [13] (see Section 2.3.4). An example of this is the Deterministic Uncertainty Estimation (DUE) method [14].

Uncertainty estimation is still an *open problem* and a lot of metrics have been proposed for catching it [15][16][17], but in the general case two main categories of uncertainty are studied (see Section 2.3.3):

- *Epistemic*. This is the uncertainty present in the model and in theory can be explained given enough training samples.
- *Aleatoric*. Related to the noise inherent in data.

The goal of this project is to *merge* the two lines of work described above, *adversarial robustness* and *uncertainty estimation*. The primary question posed at the beginning of this thesis was: *Is there a correlation between samples that can be perturbed to successfully attack the model and on-average increased model's uncertainty estimates for those samples?* And if true, *would it be possible to perform an uncertainty-guided exploration to find adversarial samples?* To answer the first question some preliminary experiments were done on the MNIST dataset [18], using both the Monte Carlo Dropout [12] approach and the Variational BNN approach [10] (see Chapter 3). After having found that the model's estimated uncertainty is on-average *higher* when the Projected Gradient Descent (PGD) adversarial perturbation [19] is applied to the input data with respect to when no perturbation

is applied to the samples, this work focuses on designing some adversarial attacks that aim at maximizing the model's uncertainty. These Uncertainty Targeted Attacks (UTA) will be extensively analyzed in the rest of this work from different perspectives and using different methods and it will be shown how uncertainty-guided exploration has some advantages with respect to state of the art techniques (see Chapter 4).

To summarize, the main contributions given in this thesis are the following:

- Study of the connections between uncertainty estimation and adversarial training.
- Proposing a novel class of adversarial attacks that aim at maximizing the model's estimated uncertainty by modifying the input samples. These attacks are called *Uncertainty Targeted Attacks* (UTAs).
- Empirical evaluation of the effectiveness of these attacks when implemented either in the image space or in the latent space of a model and comparison with the most widely used attack techniques using different uncertainty estimation methods: BNNs, Monte Carlo Dropout, and DUE (based on Gaussian Processes).
- Study on the effect of adversarial training on methods based on kernel learning.

It will be shown how attacks that aim at maximizing uncertainty do not degrade the test accuracy on clean (nonperturbed) data, are robust to catastrophic overfitting and to state of the art attacks like Projected Gradient Descent.

Chapter 2

Related Work

2.1 Adversarial Attacks

As described in the previous chapter, Neural Networks (NNs), despite being successful in a lot of complex tasks, can be easily *fooled by an adversary* that modifies the input data in a way that maximizes the network loss given the modified samples [6]. A classifier $\mathcal{C}_{\boldsymbol{\omega}} : \mathcal{X} \mapsto \mathbb{R}^c$ can be defined as an hypothesis function that maps an input sample $\boldsymbol{x}$ in the input space $\mathcal{X}$ from a finite dataset $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$ drawn from the data distribution $\mathcal{P}_d$, to an output $\hat{\boldsymbol{y}} \in \mathbb{R}^c$ containing the prediction over c possible classes; $\boldsymbol{\omega} \in \Omega$ represent the model's parameters, what is usually optimized during training. The output of the classifier consists in a vector with the Softmax predictions of the network, i.e. for each class the probability that a particular element belongs to that class.

A loss function can be defined as $\mathcal{L} : \mathbb{R}^c \times \mathbb{Z}_+ \mapsto \mathbb{R}_+$ in the form of a mapping that takes the prediction of the network given a sample (in the form of the Softmax output) and the real class of the sample, expressed as a positive integer corresponding to the index of the class from 1 to c, and outputs a real positive number expressing *how far* the prediction of the network is from the actual class. Training a network means *minimizing* this loss function given some training data.

After having described this setting one can imagine an adversary that takes a *normal* (clean) sample $\boldsymbol{x}$ and adds to it a small *perturbation* $\boldsymbol{\delta}$ such that the resulting sample $\boldsymbol{x} + \boldsymbol{\delta}$ maximizes the loss of the classifier:

$$\max_{\boldsymbol{\delta} \in \Delta} \mathcal{L}(\mathcal{C}_{\boldsymbol{\omega}}(\boldsymbol{x} + \boldsymbol{\delta}), \boldsymbol{y}) \,, \tag{2.1}$$

where Δ is the set of the allowed perturbations. Defining this set is not so straightforward: the perturbations should be *small* in order to generate some adversarial samples that are *indistinguishable for a human* from the clean samples or at most are a little bit noisier. These perturbations on the other hand should

be able to *fool* the classifier by making it believe that the samples belong to a different class. Of course, it is not possible to define this set in an exact way, so most adversarial attacks techniques define a *subset* of this Δ.

A reasonable choice of Δ that is commonly used is the $\boldsymbol{l}_\infty$ ball defined as:

$$\Delta = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_\infty \leq \boldsymbol{\varepsilon}\} \tag{2.2}$$

where the $\boldsymbol{l}_\infty$ norm of a vector a is defined as $\|a\|_\infty = \max_i |a_i|$, so using this norm means *constraining the magnitude* of the perturbation to be inside the range $[-\varepsilon, \varepsilon]$. [20] argue that the $\boldsymbol{l}_\infty$ norm is the *optimal* distance metric to use to quantify the similarity of two samples in the adversarial setting. Other choices of Δ that are commonly used are the $\boldsymbol{l}_1$ or the $\boldsymbol{l}_2$ ball. This last metric was used especially in the beginnings of the adversarial attacks studies [6], because the $\boldsymbol{l}_2$ distance between two samples stays small when there are many small changes to many pixels.

As a technical detail, usually when performing an adversarial attack the perturbed sample should be a *valid* sample, e.g. if all the elements of the dataset are in the range $[0, 1]$ then a perturbed $\boldsymbol{x} + \boldsymbol{\delta}$ sample should lie in this range. This can be enforced by *clamping* the adversarial samples to the desired values.

The type of attack introduced in Equation 2.1 is called *untargeted*, because its goal is to fool the network by making it believe that a particular sample belongs to a different, non specified, class. A different approach consists in *targeted* attacks: in this case the attacker wants to fool the network by letting it believe a sample belongs to a specific class different than the actual one. Mathematically, given the real class index $\boldsymbol{y}$ and the target class index $\boldsymbol{y}_t$, targeted attacks can be defined as follows:

$$\max_{\boldsymbol{\delta} \in \Delta} \mathcal{L}(\mathcal{C}_{\boldsymbol{\omega}}(\boldsymbol{x} + \boldsymbol{\delta}), \boldsymbol{y}) - \mathcal{L}(\mathcal{C}_{\boldsymbol{\omega}}(\boldsymbol{x} + \boldsymbol{\delta}), \boldsymbol{y}_t)\,, \tag{2.3}$$

so the attacker will *maximise* the loss of the classifier with respect to the real class and *minimize* it with respect to the target class. In this work, only *untargeted* attacks will be analyzed. The main method proposed in this thesis is called: Uncertainty Targeted Attacks (UTAs). In this case the word *targeted* does not refer to the fact that the attack targets a particular class, but it is related to the fact that the target of optimization is uncertainty. The UTA method beside its naming is *non-supervised*.

2.1.1 Fast Gradient Sign Method

In literature several methods have been proposed to generate adversarial perturbations; one of the most famous is the Fast Gradient Sign Method (FGSM) [5] that can be defined for the $\boldsymbol{l}_\infty$ ball as:

$$\boldsymbol{\delta}_{FGSM} \triangleq \varepsilon \cdot \operatorname{Sign} \nabla_{\boldsymbol{x}} \mathcal{L}(\mathcal{C}_{\boldsymbol{\omega}}(\boldsymbol{x}), \boldsymbol{y})\Big)\,. \tag{2.4}$$

This method consists in performing one optimization step, by computing the gradient of the loss and then apply a perturbation of magnitude ε to each component of the sample with the *sign* of the computed gradient. The basic idea here comes from the intuition that for each component of the input data the best possible adversarial attack inside the $\boldsymbol{l}_\infty$ ball of dimension ε consists in taking the *biggest* allowed step for maximizing the loss in the direction given by the sign of the gradient. It could be shown that FGSM is the optimal attack under the $\boldsymbol{l}_\infty$ norm against a binary linear classifier [5]. However in practice, this is not the case for deep neural networks [19].

A variant of the standard FGSM attack consists in starting from a random point inside the ε ball and then adding the perturbation computed using the sign of the gradients. It will be shown in the next sections that this helps in fighting a common problem in adversarial training: catastrophic overfitting. This modified perturbation is known as Random start FGSM (R-FGSM) [8] and can be defined as:

$$\boldsymbol{\delta}_{R-FGSM} \triangleq \prod_{||\cdot||_\infty \leq \varepsilon} \Big(\xi + \alpha \cdot \mathrm{Sign}\Big(\nabla_{\boldsymbol{x}} \mathcal{L}(\mathcal{C}_{\boldsymbol{\omega}}(\boldsymbol{x}), \boldsymbol{y}) \Big) \Big), \tag{2.5}$$

where $\prod$ represent the projection into the $\boldsymbol{l}_\infty$ ball, $\xi \sim U([-\varepsilon, \varepsilon]^d)$ with d dimension of the input space and the hyper-parameter step size α is introduced to limit the magnitude of the step.

2.1.2 Projected Gradient Descent

A generalization of the FGSM methods consists in taking multiple optimization steps, instead of only one, to find the adversarial perturbation. This method is known as Projected Gradient Descent (PGD) [19]. Given $i = 1, \ldots, k$ the PGD perturbation at step i can be defined as:

$$\boldsymbol{\delta}^i_{PGD} \triangleq \prod_{||\cdot||_\infty \leq \varepsilon} \Big(\boldsymbol{\delta}^{i-1}_{\mathrm{PGD}} + \alpha \cdot \mathrm{Sign}\Big(\nabla_{\boldsymbol{x}} \mathcal{L}(\mathcal{C}_{\boldsymbol{\omega}}(\boldsymbol{x} + \boldsymbol{\delta}^{i-1}_{PGD}), \boldsymbol{y}) \Big) \Big). \tag{2.6}$$

PGD with k steps is often referred as PGD-k. Like for the FGSM case a modified version of PGD has been proposed by adding random restarts [8]. In this case at each restart the adversary starts from a random place in the ε-ball, then the perturbation that maximizes the loss among the various restarts is chosen. Given k steps and n restarts the corresponding attack will be defined as PGD-k-n. In Figure 2.1 are shown some adversarial samples coming from the CIFAR-10 dataset [21] using the FGSM ($\varepsilon = \frac{8}{255}$) and PGD-50-10 attack ($\varepsilon = \frac{8}{255}$, $\alpha = \frac{\varepsilon}{4}$). The magnitude of the perturbation ε used in Figure 2.1 is quite big for the CIFAR10 dataset and it is definitely possible to see the difference between clean and perturbed samples, nevertheless these additional perturbations do not modify the human ability of

classyfing the objects. Furthermore note how the FGSM and PGD-50-10 perturbed samples visually look almost the same, even if the PGD-50-10 is a much stronger attack.

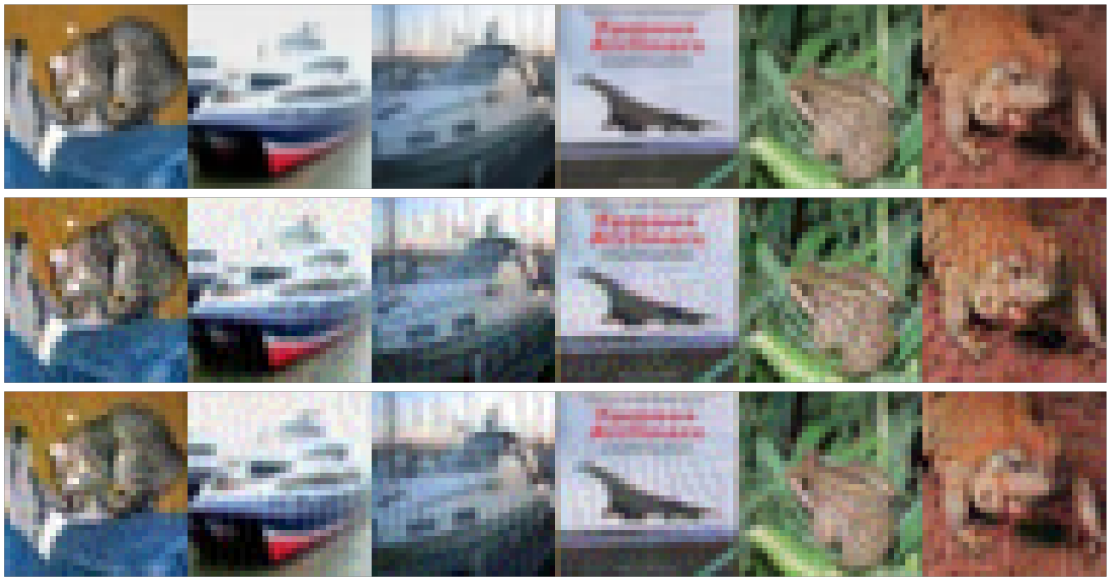


Figure 2.1: Comparison between clean images (*top row*), FGSM perturbed images (*middle row*) and PGD-50-10 perturbed ones (*bottom row*). Note that perturbed samples appear only slightly noisier, but it is still definitely possible for a human to recognize the correct label.

2.2 Adversarial Training

One of the most effective solution to defend the networks against adversarial attacks is to incorporate the adversarial samples during training, so that the network can be more robust to an adversary. Solving the adversarial training problem means solving the following min-max optimization:

$$\min_{\boldsymbol{\omega}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{P}_d}[\max_{\boldsymbol{\delta}\in\Delta} \mathcal{L}(\mathcal{C}_{\boldsymbol{\omega}}(\boldsymbol{x}+\boldsymbol{\delta}),\boldsymbol{y})]\,. \tag{2.7}$$

In general this optimization is *non-convex*. A proposed solution is to approximate the inner maximization problem with Taylor expansion and then use Lagrangian multipliers [22]. For the $\boldsymbol{l}_\infty$ bounded attacks this approximation yields to the FGSM attack. In the case of a linear binary classifier the Danskin's theorem holds [23]. This means that the gradient of the inner function in Equation 2.7 (the maximization term) is equal to the gradient of the function evaluated at its

maximum. Starting from this it is possible to show that in this specific case FGSM is the optimal attack [5][19].

However in practice in the case of DNNs most of the times AT is empirically solved by adding some perturbations that are just *good enough* and by minimizing the loss of the classifier using the perturbed data for training. This is equivalent to *lower bounding* the inner maximization problem.

Further studies on adversarial training showed that in general better performances in terms of adversarial accuracy come at the cost of reduced clean accuracy with respect to a model trained only with non perturbed data [24], indicating that the two objectives of robustness and clean accuracy may be *competing*.

Another general problem of Adversarial Training (AT) is *transferability*, i.e. models trained using samples perturbed with one particular type of attack usually are not robust to other attacks, e.g. a model trained with FGSM perturbed samples may not be robust to a PGD-10 adversary [8][6][5].

2.2.1 Latent Space Attacks

In their work [25] show how adversarial perturbations leave the data *manifold*. On the other hand, an on-manifold perturbation can be created by applying the perturbation in the *latent space* of a model rather than in the input space. They show that on-manifold perturbations used during AT boost generalization on synthetic datasets. For doing so they apply the adversarial attacks in the latent space of a VAE-GAN, i.e. a generative model that merges the approaches of Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) [26] [27]. Alternatively it is possible to split a traditional neural network into an *encoder* and a *classifier* part and apply the latent space attacks in the output of the encoder as shown in Figure 2.2.

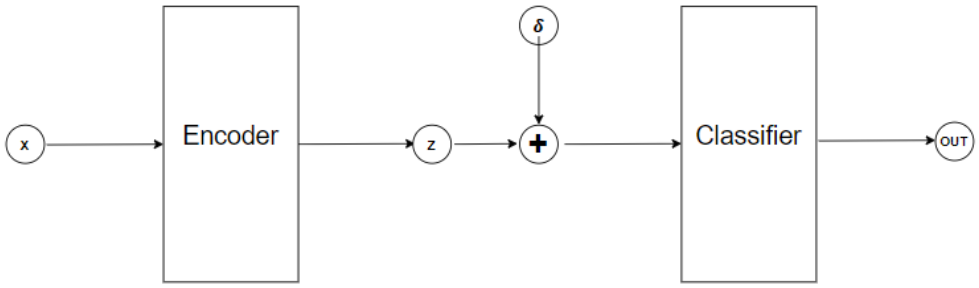


Figure 2.2: Diagram of a perturbation applied in the latent space of a classifier using a deep encoder.

2.2.2 Catastrophic Overfitting

Recent studies identified another problem of adversarial training called *Catastrophic Overfitting* (CO) [8]. This problem happens when a model is trained using modified samples coming from a particular attack, for example FGSM and is validated against another stronger attack, for example PGD-10. In particular during training it could happen that the model's robust accuracy against the stronger attack—be that measured on the training or on the validation set—drops to almost 0%, while the robust accuracy against the weaker attack with which the model is trained suddenly jumps to very high values, like 90% or 100%. This means that the model is *overfitting* the attack and can not generalize to others. Usually catastrophic overfitting happens during adversarial training with fast attacks, like FGSM or PGD-2 and testing against stronger versions of the same attack, like PGD-10 or PGD-50-10 (50 steps and 10 random restarts). The existence of catastrophic overfitting implies that stronger attacks are needed during adversarial training in order to be more robust at test time, but in general, since most attacks are defined in an iterative way, stronger attacks are more computationally expensive making the adversarial training of big networks almost unfeasible. In their work [8] introduce a new attack called R-FGSM (Equation 2.5) arguing that using this attack for adversarial training reduces the problem of catastrophic overfitting while still being fast. The authors of that paper show that this approach works, but they are not able to show why it works and in general why CO happens. Some recent works aims at understanding this problem [28][29].

2.3 Uncertainty Estimation

Another line of work in the machine learning field has the goal of increasing the *interpretability* of the model by associating an uncertainty estimate to the predictions of the network [15][16]. Two different kinds of uncertainty are considered in the context of machine learning: (i) *aleatoric*, describing the *noise* inherent in the observations, as well as (ii) *epistemic*, uncertainty originating *from the model*. Since aleatoric uncertainty depends on the data, it can not be explained given enough training samples and so it quantifies the noise present in the dataset. Epistemic uncertainty on the other hand can be completely explained given enough training samples. This type of uncertainty captures regions in which there are not so many training data points (*unexplored* regions) or overlapping regions between classes.

Several methods have been proposed for estimating uncertainty, the most important ones that will be used in this thesis are described in detail in the following sections.

2.3.1 Bayesian Neural Networks

While standard neural networks perform a Maximum Likelihood Estimation (MLE), so a *point estimation*, of the parameters of the network $\boldsymbol{\omega} \in \Omega$, the goal of Bayesian Neural Networks (BNNs) is to estimate also a posterior distribution of the weights given the data distribution, providing a framework for estimating uncertainty. Given N Training points from the dataset $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$ using the Bayes theorem and assuming i.i.d. samples (independent and identically distributed) it is possible to define:

$$\underbrace{p(\boldsymbol{\omega}|\mathcal{D})}_{posterior} = \frac{\overbrace{p(\mathcal{D}|\boldsymbol{\omega})}^{\text{likelihood}}\overbrace{p(\boldsymbol{\omega})}^{\text{prior}}}{\underbrace{\int_{\Omega} p(\mathcal{D}|\boldsymbol{\omega})p(\boldsymbol{\omega})d\boldsymbol{\omega}}_{\triangleq p(\mathcal{D})(\text{normalizing const})}} \propto p(\boldsymbol{\omega}) \prod_{i=1}^{N} p(y_i, \boldsymbol{x}_i, \boldsymbol{\omega}) \,, \tag{2.8}$$

where $p(\boldsymbol{\omega})$ is the prior distribution over the weights. So each weight in the BNNs setting is modeled as a Random Variable (RV) and at forward time it is possible to sample from those RVs to get a prediction. Given a new sample $\boldsymbol{x}^\star, y^\star$ the predictive distribution can be defined as:

$$p(y^\star|\boldsymbol{x}^\star, \mathcal{D}) = \int_{\Omega} p(y^\star|\boldsymbol{x}^\star, \boldsymbol{\omega})p(\boldsymbol{\omega}|\mathcal{D})d\boldsymbol{\omega} \,. \tag{2.9}$$

Because of the integration with respect to the whole parameter space Ω computing this posterior is an intractable problem for most Deep Neural Networks (DNNs). Although BNNs provide a preferred way to estimate uncertainty, being able to catch both aleatoric and epistemic, this approach is *infeasible* in most complex tasks, thus some techniques have been proposed to approximate a BNN. This means to approximate its distribution over the weights; two of the most used approximation methods are:

- **Markov Chain Monte Carlo (MC) approaches** - consists in *sampling* multiple times from the network and averaging the predictions.
- **Variational Inference (VI) approches** - consists in learning a *variational distribution* $q(\boldsymbol{\omega})$ to approximate the real posterior.

MCMC methods are more accurate than VI (a.k.a. Variational Bayes) ones, but in general they can not scale to large datasets because they need to consider the whole training set at each step [11], so in the next subsection only the VI approach will be considered. An exception to what said above is the the *Monte Carlo Dropout* [12] method, an uncertainty estimation technique based on Monte Carlo (MC) sampling that is gaining a lot of popularity and will be described more in-depth in the next sections.

Variational Bayes Approach

The Variational Bayes approach [10] allows to approximate the intractable posterior with a simpler variational distribution. Training is performed by optimizing a variational lower bound of the marginalized likelihood, i.e the distribution of all observed data with the weights marginalized out:

$$p(\mathcal{D}) = \int_{\Omega} p(\boldsymbol{\omega}) p(\mathcal{D}|\boldsymbol{\omega}) d\boldsymbol{\omega} \,. \tag{2.10}$$

The lower bound then can be defined in a similar way as is done for Variational Autoencoders (VAEs) [30]:

$$\log p(\mathcal{D}) \geq \mathcal{F}(q) \triangleq \underbrace{\mathbb{E}_{q(\boldsymbol{\omega})}[\log p(\mathcal{D}|\boldsymbol{\omega})]}_{\text{Likelihood term}} - \underbrace{\mathbb{D}_{KL}\Big(q(\boldsymbol{\omega})||p(\boldsymbol{\omega})\Big)}_{\text{KL term}} , \tag{2.11}$$

with $\mathbb{D}_{KL}$ the Kullback–Leibler (KL) divergence. The likelihood term can be for example the cross-entropy loss in a multi-class classification setting. The KL term on the other hand pushes q towards matching the prior on the weights, encouraging it to be more spread out and so to ensure more stochasticity in the weights instead of having them simply approximating the maximum likelihood like in traditional NNs. Differently from VAEs in this case $p(\mathcal{D})$ is fixed and only $\mathcal{F}(q)$ is being maximized with respect to the weights. This function $\mathcal{F}(q)$ is usually known as Evidential Lower Bound (ELBO) and training a BNNs with VI is equal to maximizing this lower bound.

This method with a scale mixture prior combined with a diagonal Gaussian posterior constitutes the Bayes-by-Backprop method introduced in [10].

2.3.2 Monte Carlo Dropout

The VI method analyzed in the previous section despite being able to approximate quite well a BNN, is still computationally expensive and can not scale well to large state of the art network architectures. A very popular and effective uncertainty estimation technique consists in the *deep ensembles* approach [31] that allows to compute uncertainty by taking the predictions of each network in an ensemble and by averaging them out. This method, despite being way simpler to implement and more parallelizable than Variational Bayes, has the problem that it needs a lot of computational power and hardware in order to train the needed ensemble of neural networks. A solution to this problem is the *Monte Carlo* (MC) Dropout technique [12] consisting in approximating an ensemble of neural network by using dropout even at test time, thus obtaining a *cheap ensemble*. In this setting, dropout is added after each layer and at test time is kept active, so that it is possible to sample from the network and compute uncertainty given multiple predictions. It

can be shown that this technique is an approximation of BNNs and it has gained a lot of popularity because of its simplicity, easiness to implement, and scalability.

2.3.3 Popular Uncertainty Estimation Measures

After having described the techniques used to compute uncertainty, it is natural do continue describing what are the most important and used uncertainty measures in deep learning. Estimating uncertainty is in general not a closed problem and a lot of metrics have been proposed to catch this concept.

Epistemic and Aleatoric

Given a BNN classifier $\mathcal{C}$ in a classification setting with weights $\hat{\boldsymbol{\omega}}$, let $\{\hat{\boldsymbol{\omega}}_t\}_{t=1}^T$ be T realization of the weights of the network given a test point $\boldsymbol{x}^*$, i.e. sample T times from the network. Let $\hat{p}_t = \mathcal{C}_{\boldsymbol{\omega}_t}(\boldsymbol{x}^*)$ be the vector with the Softmax probabilities for the particular realization t of the weights and let $\bar{p} = \sum_{t=1}^T \hat{p}_t$ be the vector with the mean of the predictions, then the uncertainty can be computed as:

$$U = \underbrace{\frac{1}{T}\sum_{t=1}^{T} \operatorname{diag}(\hat{p}_t) - \hat{p}_t\hat{p}_t^\top}_{\text{aleatoric}} + \underbrace{\frac{1}{T}\sum_{t=1}^{T}(\hat{p}_t - \bar{p})(\hat{p}_t - \bar{p})^\top}_{\text{epistemic}} . \tag{2.12}$$

This method is particularly used as it permits to separate aleatoric from epistemic uncertainty [17]. Furthermore, it is designed for BNNs, but can also be used with MC-Dropout or other techniques.

Entropy and Mutual Information

Another measure that has been proposed to estimate epistemic uncertainty is *entropy* [9]. Given the same setting as in the previous section, let $\bar{p} = \sum_{t=1}^T \hat{p}_t$ be the vector with the mean of the predictions, $\bar{p} \in \mathbb{R}^C$ with C number of classes, entropy can be computed as:

$$\mathcal{H} = \sum_{c=1}^{C} \bar{p}_c \log(\bar{p}_c) . \tag{2.13}$$

Additionally, sometimes the measure of Mutual Information (MI) is used to estimate epistemic uncertainty. This can be defined as:

$$MI = \mathcal{H} - \frac{1}{T}\sum_{t=1}^{T}\sum_{c=1}^{C} \mathcal{C}^c_{\boldsymbol{\omega}_t}(\boldsymbol{x}^*) \log \mathcal{C}^c_{\boldsymbol{\omega}_t}(\boldsymbol{x}^*) , \tag{2.14}$$

being $\mathcal{C}^c_{\boldsymbol{\omega}_t}(\boldsymbol{x}^*)$ the Softmax output of the network for class c given weights $\boldsymbol{\omega}_t$. MI is bounded by entropy and is used to measure the spread between the various samples of the model [32].

2.3.4 Estimating Uncertainty in a Deterministic Way

The methods seen so far for estimating uncertainty rely on stochastic networks, whether they are implemented using a VI approach, MC Dropout or other strategies. Recently some methods have emerged that aim at estimating epistemic uncertainty in a *deterministic* way, i.e. without considering the weights of the network as random variables. These techniques are often called Deterministic Uncertainty Methods (DUMs) [33] and are usually based on concepts like *Gaussian Processes* (GPs), Gaussian Mixtures Models [34] or Normalizing Flows [35]. In this work, one of these strategies to estimate the uncertainty of the model, based on Gaussian Processes, will be used. This will be described in the following subsections after an initial introduction on GPs, using the same notation of [13]. A more comprehensive discussion on Gaussian processes can be found in [36].

Gaussian Processes

A Gaussian Process can be seen as a collection of random variables having a joint Gaussian Distribution. The following formulation of GPs is the classical one and it is valid for solving regression tasks, later it will be explained how Gaussian Processes can be modified to solve classifications tasks. With GPs it is possible to model a *distribution over functions*, i.e. given a dataset $\mathcal{D}$ with N vectors $X = \boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ and a vector of targets $\boldsymbol{y} = (y(\boldsymbol{x}_1), \dots, y(\boldsymbol{x}_n))^T \in \mathbb{R}^{N\times 1}$, it is possible to define a function value $f(\boldsymbol{x}) \sim \mathcal{GP}(\boldsymbol{\mu}, K)$ and any collections of functions values $\mathbf{f}$ can be modeled as:

$$\mathbf{f} = f(X) = [f(\boldsymbol{x}_1), \dots, f(\boldsymbol{x}_n)]^T \sim \mathcal{N}(\boldsymbol{\mu}, K)\,. \tag{2.15}$$

The Gaussian distribution has a vector of means $\boldsymbol{\mu} \in \mathbb{R}^{N\times 1}$ such that $\boldsymbol{\mu}_i = \mu(\boldsymbol{x}_i)$ and a covariance matrix $K \in \mathbb{R}^{N\times N}$. To $\boldsymbol{\mu}$ is associated a user specified function $\mu(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$ (usually a vector of zeros is used to model the Gaussian Process mean) and to K a kernel function that models the covariance between two function values, i.e. $K_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = cov(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j))$. Many functions can be used as kernel in Gaussian processes, and choosing the right one is a crucial designing task. The kernel models the correlation between the points in the dataset and by acting on it is possible to modify the shape of the function $f(X)$. A reasonable and popular choice for the kernel is to use the Radial Basis Function (RBF) defined as:

$$k_{RBF} = (\boldsymbol{x}_i, \boldsymbol{x}_j) = \nu^2 e^{\frac{-\frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{l^2}}\,, \tag{2.16}$$

where ν^2 and l are the two hyperparameters of the kernel. ν^2 is known as signal variance and represent the square of the average distance from the function's mean. On the other hand, l is called the lengthscale parameter and represents the reach of influence of a point to its neighbors. If l is small it means that the functions vary more rapidly depending on the input $\boldsymbol{x}$.

The targets $y(\boldsymbol{x})$ can be modelled with a GP and assuming additive gaussian noise σ^2: $y(\boldsymbol{x})|f(\boldsymbol{x}) \sim \mathcal{N}(y(\boldsymbol{x}); f(\boldsymbol{x}), \sigma^2)$ (likelihood over data); then given N_* test points X_* the predictive distribution is given by:

$$\begin{aligned} &\mathbf{f}_*|X_*, X, \boldsymbol{y}, \boldsymbol{\theta}, \sigma^2 \sim \mathcal{N}(\hat{\mathbf{f}}_*, cov(\mathbf{f}_*)), \\ &\hat{\mathbf{f}}_* = \boldsymbol{\mu}_{X_*} + K_{X_*,X}[K_{X,X} + \sigma^2 I]^{-1}\boldsymbol{y}, \\ &cov(\mathbf{f}_*) = K_{X_*,X_*} - K_{X_*,X}[K_{X,X} + \sigma^2 I]^{-1}K_{X,X_*}\,, \end{aligned} \tag{2.17}$$

where $\boldsymbol{\theta}$ represents the kernel hyperparameters, all the covariance matrices, depend on them. The predictive posterior in Equation 2.17 can be computed starting from the prior $f(\boldsymbol{x})$ and the likelihood $y(\boldsymbol{x})|f(\boldsymbol{x})$ and using the Bayes theorem.

It is possible then to optimize the Gaussian Process, i.e. finding the best kernel hyperparameters $\boldsymbol{\theta}$ that maximize the marginal likelihood. In Gaussian Processes the optimization is not done by maximizing the likelihood because this is a function of $\boldsymbol{\theta}$ and $f(\boldsymbol{x})$ and the objective function should depend only on $\boldsymbol{\theta}$. The solution is then to optimize the *marginal* likelihood $p(\boldsymbol{y}|\boldsymbol{\theta})$, the denominator of the bayes formula. This is equivalent to maximizing:

$$\log p(\boldsymbol{y}|\boldsymbol{\theta}) \propto -[\underbrace{\boldsymbol{y}^T(K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1}\boldsymbol{y}}_{\text{model fit}} + \underbrace{\log|K_{\boldsymbol{\theta}} + \sigma^2 I|}_{\text{complexity penalty}}]\,. \tag{2.18}$$

Note how Gaussian Processes are a non parametric model, this means that after the learning of the parameters it is necessary to keep all training data X and $\boldsymbol{y}$ because they are also present in the posterior. This limits the scalability of Gaussian Processes to big datasets.

Gaussian Processes suffer from two *computational bottlenecks*: the first consists in solving the linear system $(K + \sigma^2 I)^{-1}\boldsymbol{y}$, while the second in computing the determinant $\log|K + \sigma^2 I|$. The standard way of solving this is by computing the Cholesky decomposition of K and this procedure is $\mathcal{O}(N^3)$ in time complexity and $\mathcal{O}(N^2)$ in storage. After learning, computing the mean and variance of a single test point respectively cost $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$.

In order to solve these computational issues, many approaches focus in estimating not only the kernel hyperparameters, but also some *inducing points*, i.e. points that *summarize* the data and then use these points to compute an *approximated* version of the kernel $\tilde{K}$ [37].

Furthermore, the described Gaussian Processes work only in the case of regression tasks, in the case of classification the likelihood of the data can no longer be modeled using a Gaussian distribution and as a consequence all the properties of random Gaussian vectors, that allow an analytical derivation of the posterior, can no longer be used. If for example a *categorical distribution* is used for the likelihood (like in the case of multi-class classification), the Gaussian properties of the posterior do not hold anymore and the computation of a posterior becomes intractable. In

order to adapt Gaussian Processes to perform classification tasks then a *variational inference* approach is needed, i.e. instead of computing the exact posterior, use a variational approximation of it.

Deep Kernel Learning

In their work [14] introduce a new method for estimating uncertainty based on the concept of Gaussian Processes: *Deterministic Uncertainty Estimation* (DUE) that is built on the basis of another popular similar method: *Deep Kernel Learning (DKL)* [38]. Starting from DKL the main idea of this method is to consider a simple neural network as a feature extractor (be that fully connected or convolutional) and then to add as final hidden layer of the network a Gaussian Process. Depending on the kernel, e.g. RBF, this would be like adding a new layer with *infinite neurons*. Then by maximizing the marginal likelihood of the Gaussian Process it is possible to jointly learn the network weights $\boldsymbol{\omega}$ and the kernel hyperparameters $\boldsymbol{\theta}$. The kernel used is not the exact one but is an approximation created using the *inducing points* technique for faster computation, using the same method proposed in [13]. With this method, given N points, is possible to obtain a computational complexity for inference and learning of $\mathcal{O}(N)$.

Deterministic Uncertainty Estimation

In the DUE [14] paper on the other hand the authors argue that DKL has the problem of suffering from what they call *feature collapse* of the latent space, i.e. points that are far away in the image space can become near in the latent space and this is the input of the Gaussian Process. As a consequence, the uncertainty estimation of methods such as DKL can be unreliable because points that are out of distribution or outliers may end up in high density regions in the latent space and thus being associated with a low epistemic uncertainty estimate. In DUE the good properties of the latent space desired (basically this means to approximately preserve the distances among points) are ensured by using as feature extractor a network that is *bi-Lipschitz*. It has been shown that this property brings a lot of advantages in many scenarios [39]. The bi-Lipschitz property can be obtained by using a network with *residual connections* and *spectral normalization*, this approach is also used in another GP based method: Spectral-normalized Neural Gaussian Process (SNGP) [40]. Ensuring good properties of the latent space allows to exploit the *clustering capabilities* of it. The authors of DUE suggest that for the classification tasks is enough to use as many inducing points as the number of classes. The idea is that each of these points will be representative of a particular class, acting in a similar way as *class centroids*. In fact the initial inducing points, before training the GP, in DUE are learned by using k-means on the feature representation of the points in the dataset. It will be the goal of the GP process

then to find the best inducing points, the parameters of the network and the kernel hyperparameters using a variational approach. Thanks to DUE is possible to use a very small number of inducing points for classification (not dependent on the size of the dataset), thus ensuring quite fast training (almost as fast as normal NNs). The uncertainty is then estimated by sampling from the learned variational posterior and using the *entropy* measure.

Chapter 3

Connections between uncertainty and adversarial training

The first step of this work consists in *understanding* whether there are any *connections* between the model's estimated uncertainty and adversarial training. What happens to the model's uncertainty during adversarial training? Will adversarial training have an influence or not? All of these questions will be answered in this chapter. As a framework for computing uncertainty two different techniques have been used: initially are presented the experiments done using BNNs approximated via the Bayes-By-Backprop method, introduced in Section 2.3.1. Then the same experiments are be repeated using the MC Dropout technique (Section 2.3.2). This choice comes from the fact that the Variational Bayes method offers a *more theoretically grounded* approach to uncertainty estimation. On the other hand this technique is not so scalable to big networks. So the very first preliminary experiments have been done on small Variational BNNs to investigate the connections between AT and uncertainty estimation, then the experiments have been repeated with MC Dropout to see if the same conclusions found using BNNs hold using this *more scalable* but less exact method.

Regarding the uncertainty estimation methods, in this chapter will be presented the results using 4 different measures (depending on the section): epistemic and aleatoric uncertainty as defined in Section 2.3.3; entropy and mutual information as defined in Section 2.3.3.

All the experiments in this chapter have been done using the MNIST dataset [18], containing 60000 training and 10000 test images of labeled handwritten digits from 0 to 9 (10 classes classification problem). The training set has been further divided in training and validation set with respectively 50000 and 10000 samples.

3.1 BNNs experiments

For these BNN experiments the Blitz PyTorch library has been used [41], containing the implementations of the most used NN layers in a Variational Bayesian setting according to the original Bayes-By-Backprop paper [10]. The network used in this section is a Bayesian version of a very simple Multi-Layer Perceptron (MLP) architecture consisting in three Bayesian fully connected layers: the first with 512 neurons, the second with 128 and the final output layer with 10 neurons. The activation function used after each layer is the Rectified Linear Unit (ReLu) [42] for the hidden layers and the Softmax [43] for the output layer in order to provide the class probabilities. Uncertainty has been estimated by sampling from the network $T = 20$ times and by using these samples for computing the desired uncertainty measure. For these Variational BNNs experiments only the epistemic and aleatoric uncertainties measures are considered here. The experiments related to entropy and mutual information are shown in the Appendix A.

3.1.1 Clean training

Training has been done initially in a *clean* setting, i.e. without using any adversarial sample at all. The resulting model had been tested against the clean validation set and the PGD-10 perturbed validation set, using a perturbation magnitude $\varepsilon = 0.1$ and a step size $\alpha = 0.01$. In Figure 3.1 some samples perturbed with this attack are shown; note how the perturbation applied is quite small for the MNIST dataset and it is definitely possible for the human eye to recognize the correct label.

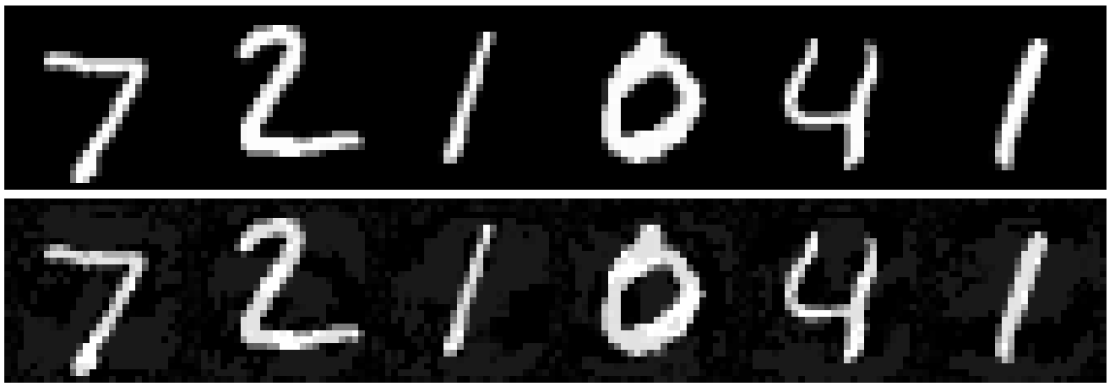


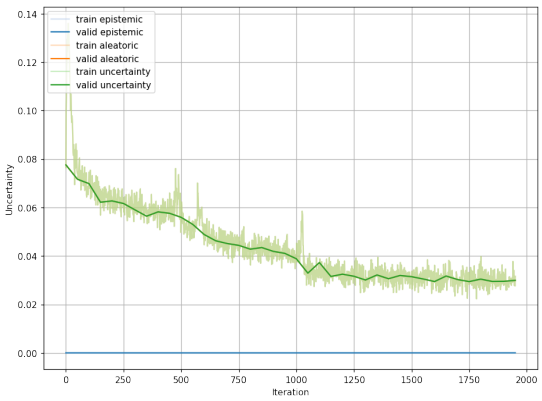
Figure 3.1: Comparison between clean images (*top row*) and PGD-10 perturbed images (*middle row*) on the MNIST dataset.

The network has been trained for 20 epochs with a batch size of 512 using the Adam optimizer [44] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ as suggested in the original paper

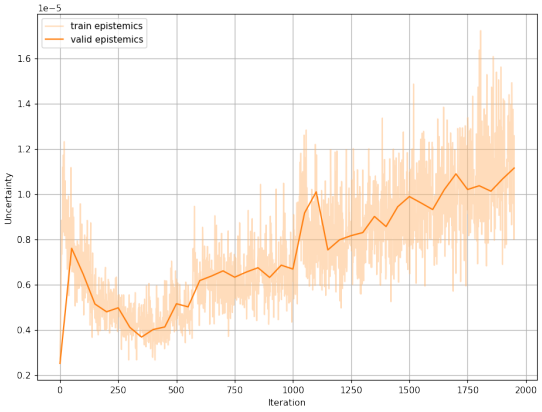
and an initial learning rate of 0.001. Results are shown in Figure 3.2. In particular, in subfigure 3.2a are shown both the epistemic and aleatoric uncertainty during the various training iterations, the total uncertainty is the *sum* of the aleatoric and epistemic terms. The former is greater in magnitude than the latter, so in subfigure 3.2b a *zoom* on epistemic uncertainty only during training is shown. The epistemic uncertainty seems to be increasing during training, this can be seen as something quite counterintuitive because in theory the epistemic uncertainty should be explained if the model sees enough data. A possible explanation of what is seen in subfigure 3.2b is that the model is becoming *less overconfindent* as the training goes on. In theory given a very big dataset and an enough number of epochs the epistemic uncertainty should decrease, but this does not appear to be the case with such a small dataset like MNIST. Furthermore, note how the models considered in this chapter have very *low uncertainty*, indeed there are many oscillations during training around very small epistemic values (on the order of 10^{-5}). In subfigure 3.2c the negative ELBO (Evidential Lower Bound) during training is shown, while in subfigure 3.2d results relative to the accuracy obtained are reported, in terms of clean (valid) and adversarial accuracy. Note how the latter is smaller than the former that is because the model is not being trained adversarially and the PGD-10 attack is quite effective as is able to bring the accuracy of the network from something like 0.98 down to less than 0.50.

3.1.2 Adversarial training

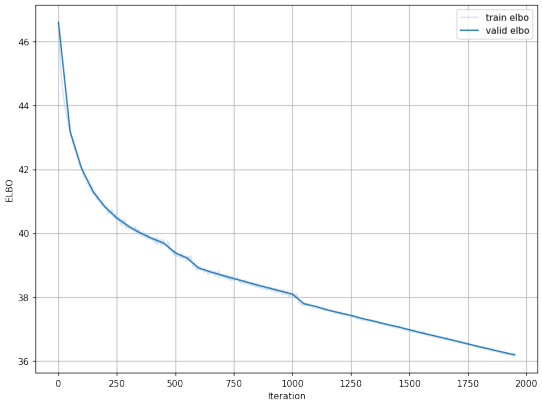
The same experiments are repeated using the modified PGD-10 samples during training instead of the clean ones with the same hyperparameters described in Section 3.1.1, with the only difference that the network is now trained for 40 epochs instead of 20. Results are shown in Figure 3.3. In particular in subfigure 3.3a and 3.3b it is possible to see how the uncertainty in the training set is definitely higher than the uncertainty measured on the validation set. In this case, the training set contains the adversarial images, while the validation set contains only the clean samples. So a connection between adversarial training and uncertainty estimation *exists* and in general it seems, from these preliminary experiments, that the model is more uncertain when it receives as input some adversarial samples compared to when it receives only the clean ones. In this case, as shown in subfigure 3.3d the adversarial accuracy is closer to the clean accuracy because of adversarial training, hence leading to a more robust network. Furthermore note how for the first almost 2000 iterations the model's uncertainty and the accuracy stays *flat*, then after this initial phase the model starts learning how to classify the dataset. This may be due to the inherent optimization procedure present in the Variational BNNs loss consisting in a *likelihood* term and in a *KL* term. At the beginning of the training the KL term is *greater* and then the network focuses on optimizing this, then after



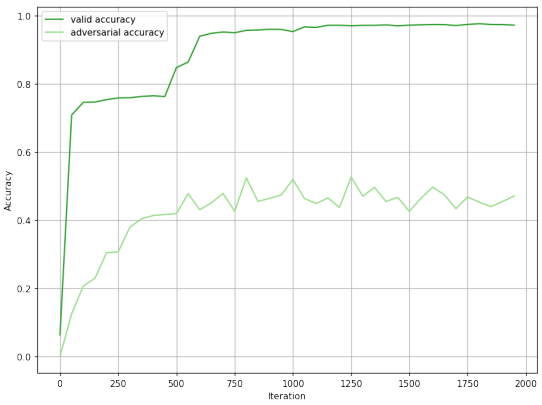
(a) train/validation uncertainty



(b) train/validation epistemic uncertainty



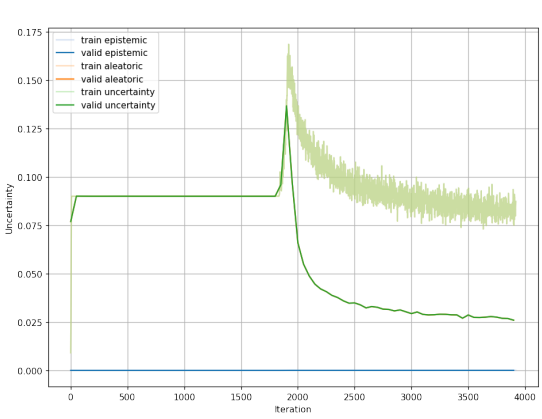
(c) train/validation ELBO



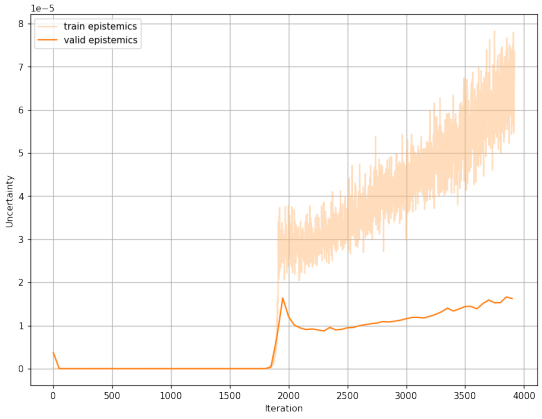
(d) clean (validation) and PGD-10 (adversarial) accuracy

Figure 3.2: Uncertainty estimation and accuracy for a **Variational BNN** trained only with clean data on the **MNIST** dataset. (a): aleatoric, epistemic and total uncertainty both on the training and validation set, note how aleatoric uncertainty is definitely higher than epistemic. (b): zoom on epistemic uncertainty only during training and validation. This term appears to have an upwards trend around very small values. (c): train and validation ELBO. (d): accuracies shown both with clean data (valid accuracy) and adversarial data using the PGD-10 attack. Note how the adversarial accuracy is definitely lower than the clean one.

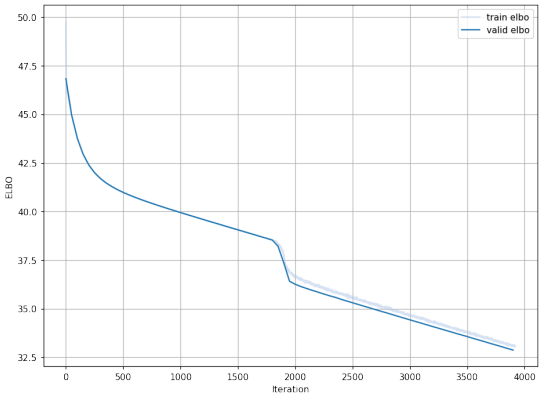
a certain amount of steps the likelihood term becomes important and the network starts learning the data. It is interesting how this behaviour happens in AT but not in clean training, this it could be related to the *min-max nature* of the AT problem that makes the variational approximation of the data prior more difficult for the network.



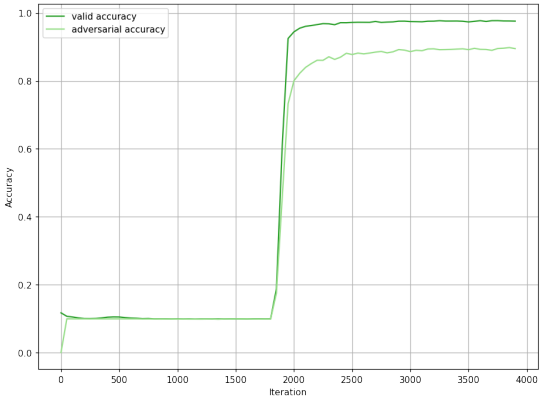
(a) train/validation uncertainty



(b) train/validation epistemic uncertainty



(c) train/validation ELBO



(d) clean (valid) and PGD-10 (adversarial) accuracy

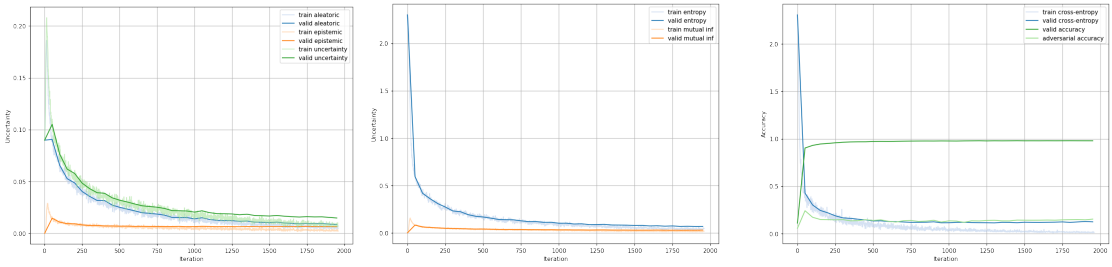
Figure 3.3: Uncertainty estimation and accuracy for a **Variational BNN** adversarially trained with PGD-10 data on the **MNIST** dataset. (a): aleatoric, epistemic and total uncertainty. In this case, the uncertainty on the training set (with perturbed samples) is definitely higher than the one measured on the validation set (with clean data only). (b): zoom on epistemic uncertainty only during training and validation. Also, the value measured on the perturbed training set is definitely higher and with a faster upward trend than the uncertainty measured on the clean validation set. (c): train and validation ELBO. (d): accuracies shown both on clean data (denoted as "valid" accuracy) and on adversarial data using the PGD-10 attack. Note how in this case the adversarial accuracy is quite high and the gap with respect to the clean one is smaller than the one shown in subfigure d.

3.2 MC Dropout experiments

The same experiments presented in Section 3.1 are now repeated using as uncertainty estimation method the MC Dropout technique to see if the previous findings can be generalized also to this *less exact* method. The network used has the same structure as the BNN of the previous section, but it uses simple Fully Connected (FC) layers instead of the Bayesian ones. In addition dropout is used after each layer (except the output one) with a rate $p_{drop} = 0.5$. In this section also the entropy metric has been considered.

3.2.1 Clean training

In Figure 3.4 the results obtained with clean training are presented. As shown in subfigures 3.4a and 3.4b entropy and mutual information seem to follow a similar behavior to the aleatoric and epistemic in quantifying uncertainty (entropy and mutual information are designed to be measures of epistemic uncertainty). In general the uncertainty estimation behaves a bit differently with respect to the BNNs case. In particular the oscillating pattern of epistemic uncertainty seen in the BNNs experiments seems *not detected* here, and after a while the epistemic value becomes almost constant.



(a) epistemic and aleatoric

(b) entropy and MI

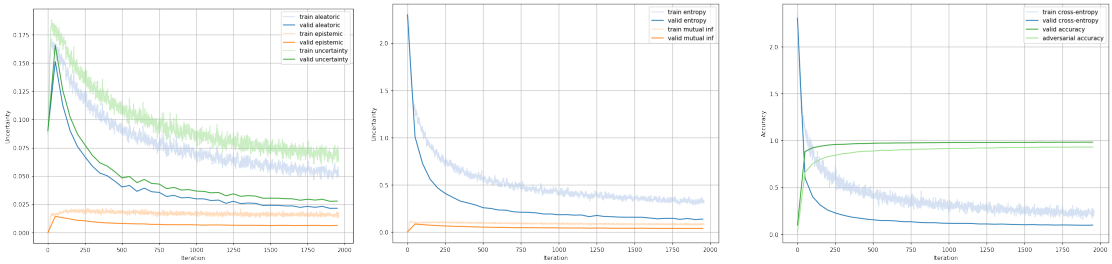
(c) clean (valid) and PGD-10 (adversarial) accuracy

Figure 3.4: Uncertainty estimation and accuracy for a **MCD** network trained only with clean data on the **MNIST** dataset. (a): aleatoric, Epistemic and sum of the twos both for the training and validation set. (b): train and validation Entropy and MI. (c): cross entropy and accuracy using clean (valid) and perturbed samples.

3.2.2 Adversarial training

In Figure 3.5 are shown the results obtained by training adversarially an MCD network and estimating the uncertainty. As in the previous case the train uncertainties reported in subfigures 3.5a and 3.5b are measured with the perturbed training set,

while the valid uncertainties with the clean set. It is possible to see also in this case how the uncertainty for the perturbed set is higher than the one measured in the clean set and the same behaviour can be observed also for the measures of entropy and mutual information.



(a) epistemic and aleatoric

(b) entropy and MI

(c) clean (valid) and PGD-10 (adversarial) accuracy

Figure 3.5: Uncertainty estimation and accuracy for a **MCD** network adversarially trained with PGD-10 on the **MNIST** dataset. (a): aleatoric, epistemic and sum of the twos both for the training and validation set, note that also in this MCD case the training uncertainty is higher than the validation one (measured on clean data only). (b): train and validation entropy and MI. (c): cross entropy and accuracy using clean (valid) and perturbed samples.

3.3 Preliminary experiments with uncertainty based adversarial training

Since adversarial training seems to affect the model's estimated uncertainty and since all the uncertainty measures presented are differentiable, it is possible to define some adversarial attacks that instead of maximizing the loss of the network aim at *maximizing its uncertainty*. The main intuition behind such attacks is that by performing AT with these uncertainty maximization perturbations the network will learn how to become more robust to the *most confusing* samples, and this will act as a regularization factor that will help the network to be more certain when faced with adversarial inputs, thus being robust to such attacks. These type of attacks are introduced here as **Uncertainty Targeted Attacks** (UTAs) and can be defined given an attack step i as:

$$\boldsymbol{\delta}^{i}_{UTA} \triangleq \prod_{||\cdot||_\infty \leq \varepsilon} \left(\boldsymbol{\delta}^{i-1}_{\text{UTA}} + \alpha \cdot \text{Sign}\Big(\nabla_{\boldsymbol{x}} \mathcal{U}(\mathcal{C}_{\boldsymbol{\omega}}(\boldsymbol{x} + \boldsymbol{\delta}^{i-1}_{UTA}), \boldsymbol{y}) \Big) \right), \tag{3.1}$$

where $\mathcal{U}$ is the function used to compute the uncertainty of the classifier $\mathcal{C}_{\boldsymbol{\omega}}$ given the data. These uncertainty maximization attacks are defined in an iterative

fashion, like PGD. This definition of UTAs makes it possible to create various versions, by changing the particular uncertainty measure used. Since the only type of uncertainty that is dependent on the model is the epistemic one, it is natural to constrain the metrics used to those capable of capturing the epistemic term. In the following sections are reported the experiments done with epistemic maximization (as defined in Equation 2.12) and entropy maximization (as defined in Equation 2.13) attacks using both Variational BNNs and MC Dropout networks.

3.3.1 Maximum epistemic adversarial training

With a slight modification of the PGD attack it is possible to craft a maximum uncertainty attack that uses *epistemic uncertainty*. In the following sections the values of $\varepsilon = 0.1$ and $\alpha = 0.01$ have been used for the maximum epistemic attack. Another important hyperparameter, affecting the attack speed, is the number of samples used for estimating uncertainty. In fact uncertainty maximization attacks require T forward passes per backward pass more than PGD, with T number of samples from the network. A value of $T = 10$ has been used. Regarding the network the training has been done with the exact same hyperparameters described in Section 3.1 and Section 3.2 respectively for the training of BNNs and MCD networks. In Figure 3.6 are shown the results of the maximum epistemic attack on BNNs. Despite the fact that the uncertainty of the input data is maximized during training the network shows lower values of epistemic uncertainty in the clean validation set compared to those obtained with AT using PGD, while keeping a certain degree of robustness to PGD-10 attacks. Furthermore note how the flat pattern seen for the first 2000 iterations during adversarial training with PGD in Figure 3.3 is not seen here.

In Figure 3.8 are shown the results of the same type of adversarial training, but using an MCD network. The same considerations still apply in this case and even with the less exact MCD method the *uncertainty-guided* exploration of the attack space is able to guarantee good robustness levels and generalization to PGD-10 attacks.

3.3.2 Maximum entropy adversarial training

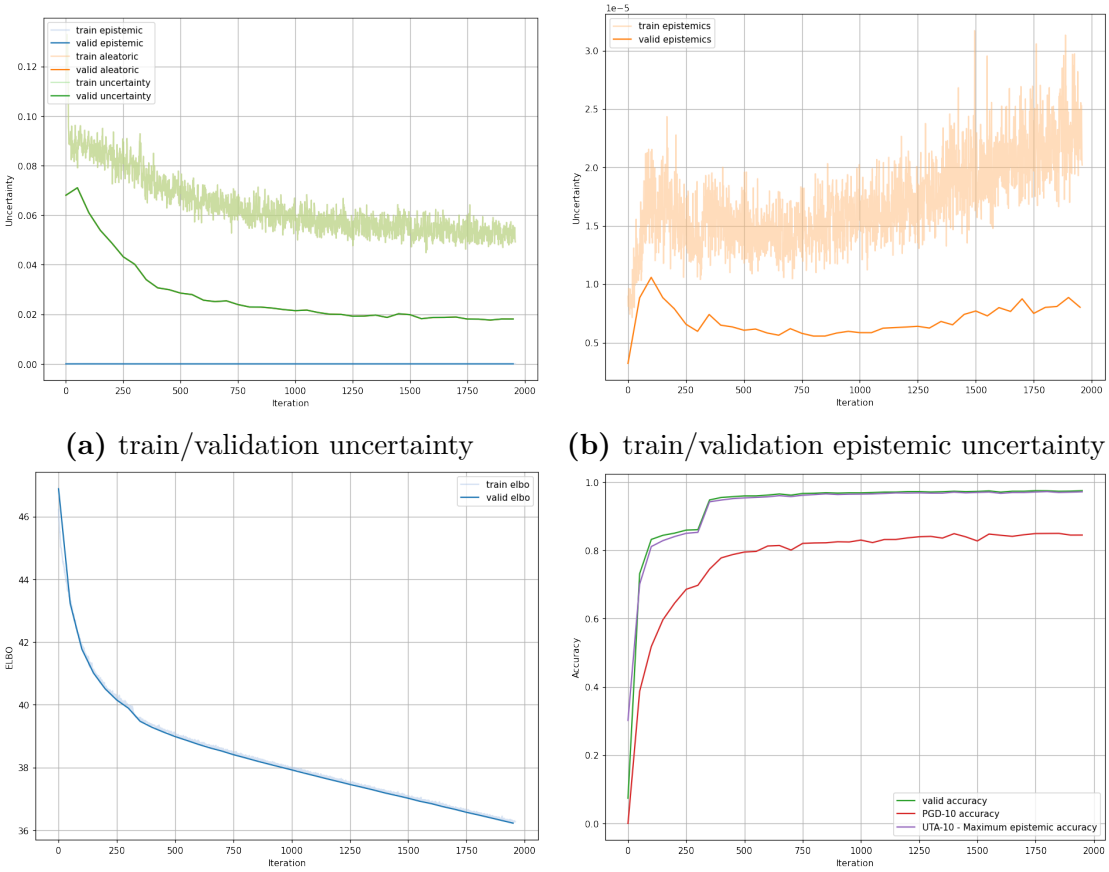
Here are reported the results obtained by using exactly the same setting as in the previous section (3.3.1), but maximizing entropy instead of epistemic uncertainty. In Figure 3.7 are presented the results obtained by training a Variational BNN on MNIST using adversarial samples crafted with the UTA attack using *entropy maximization*. As it is possible to see the results are pretty much in line to those obtained with maximum epistemic AT as shown in Figure 3.6 indicating that the two measures of uncertainty are able to catch the same variability in data. Note

how also in this case the model is able to generalize to the PGD-10 attack and is quite robust to it.

In Figure 3.9 are shown the results obtained using a MCD network. As it is possible to see the results are very similar to those in Figure 3.8.

3.4 Discussion

In this chapter the connections between adversarial training and uncertainty estimation where analyzed in the settings of Variational BNNs and MC Dropout methods. It was shown how the two concepts are *linked* and how the model outputs higher uncertainty estimates when adversarial data is given in input to it. Then some attacks that aim at *maximizing the uncertainty* of the model were introduced and the same adversarial training experiments where repeated in this setting. These attacks do not have the goal of fooling the network, but they aim at *confusing* it. A key element in defining such Uncertainty Targeted Attacks (UTAs) is the uncertainty measure used for maximization. After some experiments using both epistemic uncertainty as introduced in Equation 2.12 and entropy as introduced in Equation 2.13 it has been decided to continue the study of UTA attacks by using only the **entropy** version. As shown in [45], [46] the maximum entropy principle can be used as a way to *regularize* the predictions of the network and *reduce overfitting*. In the context of adversarial examples, [47] highlight how using an entropy regularizer during training improves the network's robustness. Moreover, entropy is an *unsupervised* measure as the labels of the samples used are not required for computing it. These elements motivated the choice of using entropy as a way to perform Uncertainty Targeted Attacks (UTAs) with respect to the other measures. These particular type of attacks involving entropy maximization will be studied extensively in the following chapter under different settings and approaches.



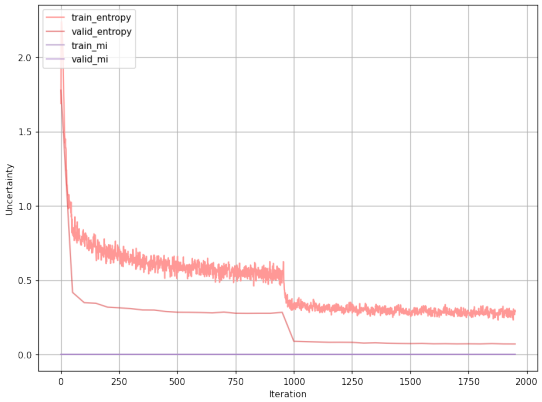
(a) train/validation uncertainty

(b) train/validation epistemic uncertainty

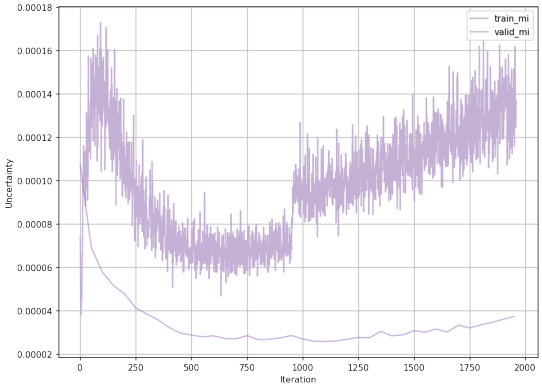
(c) train/validation ELBO

(d) clean (valid), PGD-10 and UTA-10 accuracy

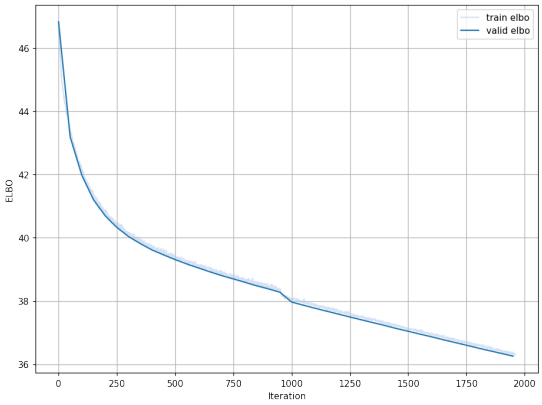
Figure 3.6: Uncertainty estimation and accuracy for a **Variational BNN** adversarially trained with a max. epistemic attack one the **MNIST** dataset. (a): aleatoric, epistemic and total uncertainty. Also in this case as in subfigure 3.3a the training uncertainty (perturbed data) is definitely higher than the validation one (clean data). Despite the fact that training is done with maximum uncertainty attack the value of total uncertainty on the perturbed dataset in general is lower than the one obtained by training with PGD. This is due to a particular property of UTA perturbations of not crossing the decision boundary of the sample's class that will be extensively analyzed in the next chapter. (b): zoom on epistemic uncertainty only during training and validation. In this particular case note how the epistemic uncertainty on the clean validation set is lower than the one obtained with PGD adversarial training shown in subfigure 3.3b. The uncertainty maximization during training makes the network more confident at prediction time. (c): train and validation ELBO. (d): clean and robust accuracy against different attacks. Note how accuracy against the attack used during training (UTA-10) is quite high, but the network is able to generalize also against PGD-10.



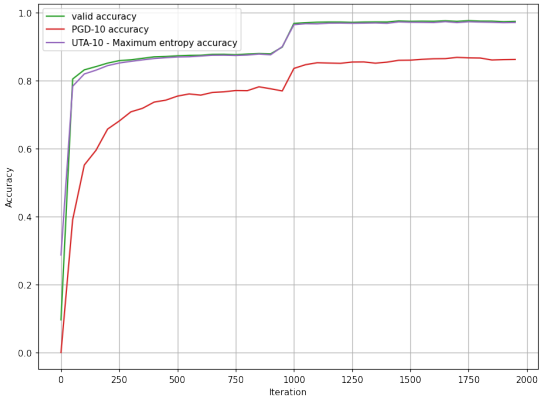
(a) train/validation uncertainty



(b) train/validation mutual information

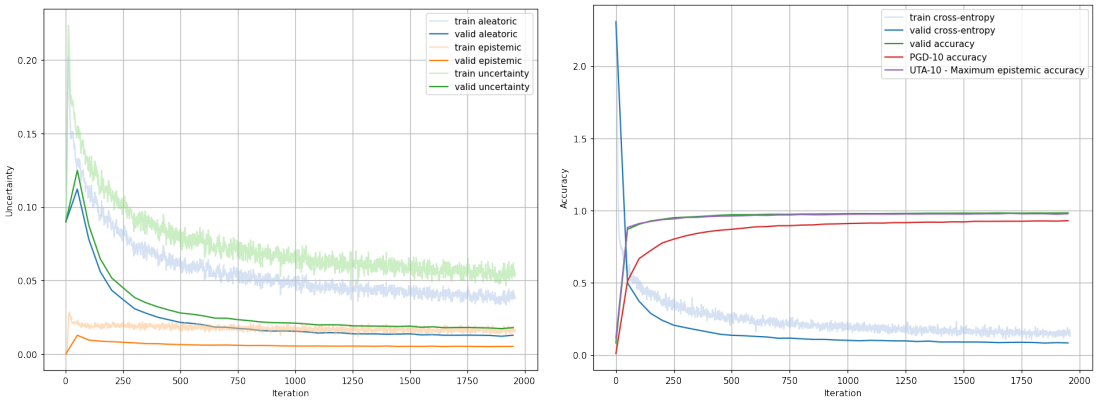


(c) train/validation ELBO



(d) clean (valid), PGD-10 and UTA-10 accuracy

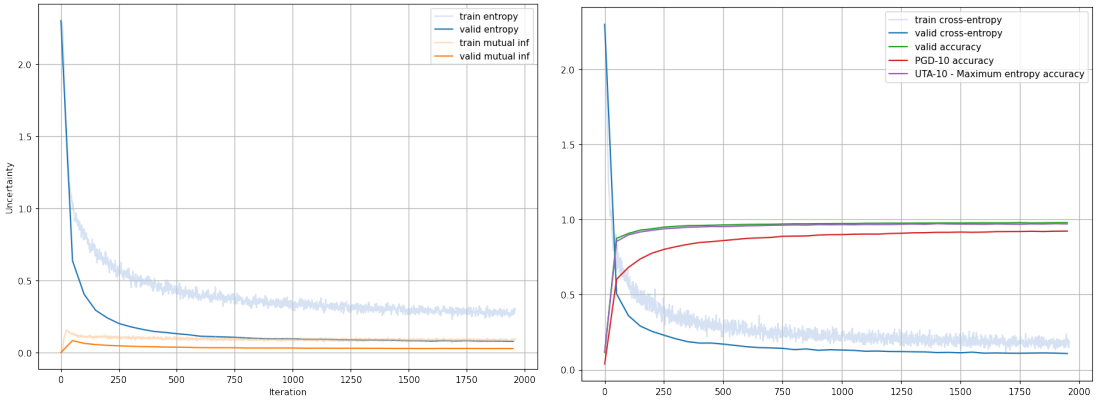
Figure 3.7: Uncertainty estimation and accuracy for a **Variational BNN** adversarially trained with a maximum entropy attack one the **MNIST** dataset. (a): aleatoric, epistemic and total uncertainty. also in this case as in subfigure 3.3a the uncertainty measured on the perturbed training set is definitely higher than the one measured on the validation set. (b): zoom on epistemic uncertainty only during training and validation. (c): train and validation ELBO. (d): clean and robust accuracy during testing under different attacks. Note how accuracy against the attack used during training (UTA-10) is quite high, but the network is able to generalize also against a different testing attack, PGD-10.



(a) train/validation epistemic and aleatoric uncertainty

(b) cross-entropy, clean (valid), PGD-10 and UTA-10 accuracy

Figure 3.8: Uncertainty estimation and accuracy for a **MCD** network adversarially trained with a maximum epistemic attack on the **MNIST** dataset. (a): aleatoric, epistemic and sum of the twos both for the perturbed training and the clean validation set. Note how also in this case the training curves are definitely higher than the validation ones evaluated with clean data only (b): cross entropy and accuracy using clean samples (valid accuracy) and samples perturbed with the PGD-10 and UTA-10 dataset.



(a) train/validation entropy and mi

(b) cross-entropy, clean (valid), PGD-10 and UTA-10 accuracy

Figure 3.9: Uncertainty estimation and accuracy for a **MCD** network adversarially trained with a maximum entropy attack on the **MNIST** dataset. (a): entropy and mutual information both for the perturbed training and the clean validation set. (b): cross entropy and accuracy using clean (valid accuracy) and perturbed samples with PGD-10 and UTA-10. Results seems pretty similar to those obtained using epistemic uncertainty maximization shown in Figure 3.8.

Chapter 4

In-Depth Study of Uncertainty Targeted Attacks

Uncertainty Targeted Attacks (UTAs) can be defined as a family of adversarial attacks that aim at maximizing the model's uncertainty during AT. In the following UTA experiments **entropy** (as defined in Section 2.3.3) is used to estimate the uncertainty of the model [48]. In this chapter this type of attack will be deeply analyzed both in the context of *input space* attacks as seen in the previous chapter, i.e. the perturbations are applied to the input images, and in the context of attacks applied to the *latent space of* an encoder network as introduced in Section 2.2.1.

Let $\mathcal{H}$ be the entropy of the model, then adversarial training with UTA can be defined as the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{\omega}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim p_d}[\mathcal{L}(\mathcal{C}_{\boldsymbol{\omega}}(\mathcal{E}(\boldsymbol{x})+\boldsymbol{\delta}_u),\boldsymbol{y})] \\ \text{s.t.} \quad &\boldsymbol{\delta}_u = \arg\max_{\boldsymbol{\delta}\in\Delta} \mathcal{H}(\mathcal{E}(\boldsymbol{x})+\boldsymbol{\delta},\boldsymbol{\omega})\,, \end{aligned} \tag{4.1}$$

where $\mathcal{E}$ is a generic encoder that brings data from the input space to a generic latent space and $\mathcal{C}_{\boldsymbol{\omega}}$ is a classifier that accepts inputs in the latent space. In the case of perturbations applied in the image space $\mathcal{E}$ is simply the identity mapping. With this formulation is possible to define both attacks applied in the input space and in the latent space. Regarding the particular iterative implementation of UTA this is defined more generally in Equation 3.1 introduced in Section 3.3 in the particular case in which *entropy* is used as uncertainty estimation metric.

An important parameter in the context of UTA perturbations is the *number of samples* done in order to estimate the uncertainty. In the rest of this chapter the notation UTA-k-t will be used to identify a perturbation obtained using t samples

for estimating uncertainty and k steps. In general for $t = 1$ the UTA perturbation would be approximately as fast as its PGD counterpart with the same number of steps, but the uncertainty estimation would not been so reliable; this setting is also defined here as *single model UTA*. In the following sections a modified version of UTA attacks that includes *random restarts* will be considered. Random restarts are added like in the PGD case. In this setting the notation used will be UTA-k-t-n with n number of random restarts. In the specific case in which attacks are applied in the input space using a single model ($t = 1$) the UTA algorithm is presented in Algorithm 1.

Algorithm 1 Pseudocode of Uncertainty Targeted Attacks in input space using a single model ($t = 1$).

Input: Classifier $C_{\boldsymbol{\omega}_0}$ with *logits* output and initial weights $\boldsymbol{\omega}_0$, stopping time T, data distribution p_d, learning rate γ, its loss $\mathcal{L}$, L_∞ ball radius ε, perturbation step size α, and number of attack iterations k.
for $t \in 0, \dots, T-1$ **do**
  Sample $\boldsymbol{x}, \boldsymbol{y} \sim p_d$
  $\boldsymbol{\delta}_u^0 \leftarrow \mathbf{0}$
  for $i \in 0, \dots, k-1$ **do**
    $\boldsymbol{p} \leftarrow \mathrm{Softmax}(\mathcal{C}_{\boldsymbol{\omega}_t}(\boldsymbol{x} + \boldsymbol{\delta}_u^i))$
    $\mathcal{H} \leftarrow -\sum_c \boldsymbol{p}_c \log(\boldsymbol{p}_c)$ *(Compute entropy)*
    $\boldsymbol{\delta}_u^{i+1} = \boldsymbol{\delta}_u^i + \alpha\, \mathrm{sign}(\nabla_{\boldsymbol{\delta}_u^i} \mathcal{H})$ *(Update the perturbations $\boldsymbol{\delta}_u$)*
    Projection $\boldsymbol{\delta}_u^{i+1} \leftarrow \prod_{\|\cdot\|_\infty \leq \varepsilon} \boldsymbol{\delta}_u^{i+1}$ *(Ensure $\|\boldsymbol{\delta}_u^{i+1}\|_\infty \leq \varepsilon$)*
  end for
  $\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \gamma \nabla_{\boldsymbol{\omega}} \mathcal{L}(\mathcal{C}_{\boldsymbol{\omega}_t}(\boldsymbol{x} + \boldsymbol{\delta}_u^K), y)$ *(Update $\boldsymbol{\omega}_t$ using $\tilde{\boldsymbol{x}} \triangleq \boldsymbol{x} + \boldsymbol{\delta}_u^K$)*
end for
Output: $\boldsymbol{\omega}_T$

4.1 Motivating example

In this section an intuitive example showing the advantages of UTA with respect to PGD will be presented using a 2-dimensional *toy dataset*, composed by dimension $X1$ and dimension $X2$ and whose samples belong to two classes, class 0 and class 1. This dataset is shown in Figure 4.1; as it is possible to see from the figure, the *average distance* between the samples of the two classes is different for dimension $X1$ and for dimension $X2$, this means that distances are *non-isotropic* between the samples of opposite classes in $\mathbb{R}^2$. It is possible to imagine this dataset as a representation of the latent space of a deep model.

In Figure 4.2 the PGD and UTA attacks are compared. In particular in subfigure 4.2a is shown the decision boundary of a classifier trained with clean data only. The

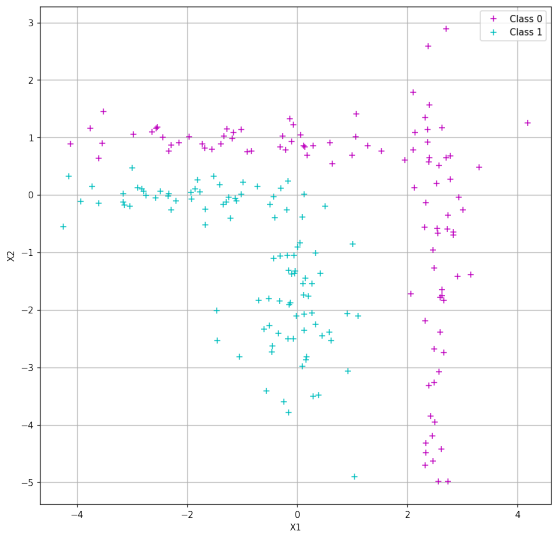
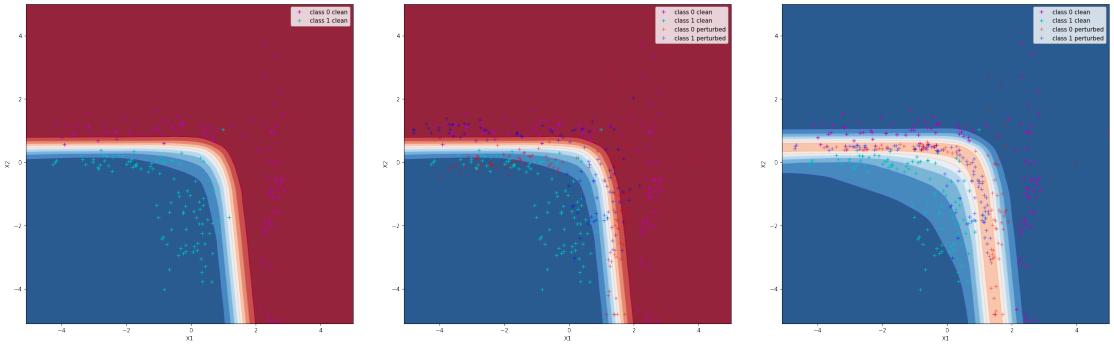


Figure 4.1: Non-isotropic distances 2-D dataset.

used classifier is a very simple NN made by one hidden layer with 50 neurons, an output layer with 2 neurons and a dropout layer between the twos with probability $p_d = 0.5$. The classifier is trained for 500 iterations on the full dataset using the MC Dropout approach. In subfigure 4.2b the adversarial samples obtained using the PGD-15 perturbation are shown (the decision boundary in the background is the one obtained with the *cleanly* trained model). The applied perturbation is computed using as hyperparameters $\varepsilon = 1$ and $\alpha = 0.1$, so quite a *big* ε for this dataset. It is possible to see in subfigure 4.2b that the PGD adversarial samples are able to c*ross the decision boundary*, thus being misclassified by the algorithm trained only with clean data. On the other hand in subfigure 4.2c are shown the samples perturbed with a UTA-15-1 perturbation, i.e. 1 sample and 15 steps. In the background instead of the decision boundary is shown the entropy of the model obtained by sampling 10 times from the MCD network. As it is possible to see entropy is high near the decision boundary, so an attack that wants to maximize entropy will move its samples to be *close to the boundary*, without crossing it. This means that even if the attack as has very big ε the samples will not go far away in the other class' region.

In Figure 4.3 are shown the decision boundaries of the model adversarially trained with PGD and UTA and the clean dataset. In subfigure 4.3a the PGD-AT decision boundary is shown (training with the same perturbation hyperparameters as the perturbations in subfigure 4.2b). As it is possible to see this decision boundary is *quite different* than the one obtained with clean training shown in Figure 4.2. On the other hand the decision boundary obtained with UTA-AT is



(a) decision boundary for a cleanly trained classifier

(b) decision boundary and PGD-15 attack

(c) entropy and UTA-15-1 attack

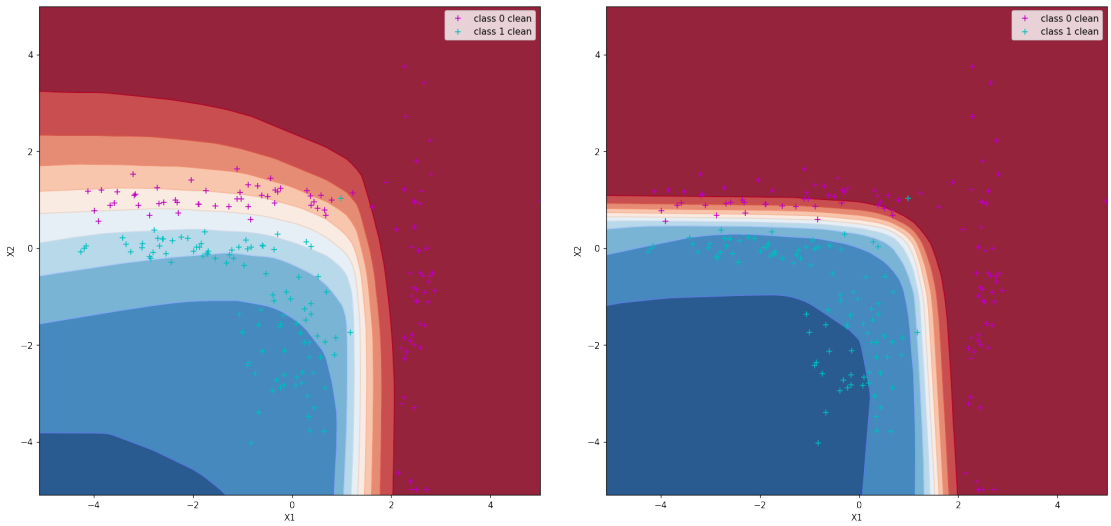
Figure 4.2: Impact of different attacks on the dataset. (a): decision boundary of a model trained with clean data only in a 2-D toy dataset. (b): decision boundary of the clean trained classifer and PGD-15 perturbation plot on top. Note how the PGD-15 samples are moved to the other side of the decision boundary. (c): entropy of the clean model and UTA-15 perturbation plotted on top. Note how entropy is high in zones near the decision boundary, thus UTA adversarial samples, differently than PGD ones, will go towards the decision boundary without crossing it.

much more similar to the clean one indicating that UTA in this case *does not degrade the clean accuracy* of the model.

4.1.1 Advantages of UTA

As shown in Figure 4.2 PGD-AT is much more *sensitive* to the choice of ε especially in non-isotropic cases. If adversarial training is done with a too small ε then the samples will not be able to move towards the boundary, thus not achieving high values of robustness. On the other hand if ε is too big the samples will *cross the boundary* and go very far away in the other class' region. As a consequence the decision boundary obtained with adversarial training will be very *different* than the one obtained with clean training because the algorithm is trying to correctly classify the adversarial samples, but this will make the accuracy of robust models measured on clean samples *lower* than that obtained with clean training.

Imagining that the toy examples in Figure 4.3 represent the latent space of a deep model it is argued here that PGD prevents the network from learning *good* latent spaces. So in the case of PGD-AT there is a *tradeoff* between robust and clean accuracy and this depends much on the choice of ε. In the case of non-isotropic dataset the distances between classes can *vary* across dimensions, thus making the tuning of a globally good ε for AT difficult (a particular value



(a) decision boundary for a PGD-15 adversarially trained classifier

(b) decision boundary for a UTA-15 adversarially trained classifier

Figure 4.3: Decision boundaries obtained with AT using PGD and UTA. (a): decision boundary of a model trained with PGD data. As it is possible to see this decision boundary is definitely different than that obtained with clean training shown in Figure 4.2 and the network will surely lose accuracy in classifying clean samples. (b): decision boundary obtained by training with UTA adversarial samples. This is definitely better than the PGD decision boundary and is much more similar to the clean training one. These experiments highlight the advantage of UTA compared to PGD of preserving the properties of the decision boundary. Imagining that this dataset represent the latent space of a classifier it is argued here that PGD may limits the network from learning *good* latent spaces.

may be too big or too small depending on the distance across dimensions). For this reason adversarial evaluation is usually done only on *few* datasets, e.g. MNIST or CIFAR-10, as the same common values of ε and α are used for *benchmarking* newer attacks against the most famous ones, like PGD.

On the other hand UTA is less sensitive to the choice ε and even if $\varepsilon \mapsto \infty$ the obtained samples will not cross the decision boundary, thus ensuring an higher clean accuracy. This is shown in Figure 4.4: on the *top row* are shown some clean images taken from the CIFAR-10 dataset [21], on the *middle row* some UTA perturbed images and in the *bottom row* some PGD perturbed images. The perturbations have the same parameters: 1000 steps, $\varepsilon = \infty$ and $\alpha = 0.1$. The network used for generating the perturbations is a ResNet18 [49] trained on the clean CIFAR-10 dataset. It is possible to see how, even for an infinite ε, the correct classes of

UTA perturbed samples are still *recognizable* by a human observer. This is not the case for the PGD perturbed images which are almost unrecognizable. UTA, being *adaptive* to the choice of ε, can be used for AT with a wide variety of datasets in which the best values of ε are not known. In an *online learning* setup, where it is not possible to know a priori the samples that will be presented to the model, this adaptability of UTA to the choice of ε can become very useful.

Standard loss-based methods for adversarial training (defined in Equation 2.1) in the context of image datasets select values for ε which are *very small*, almost imperceptible to the human eye, thus ensuring that there cannot be two data points of different classes for which their ε-balls *intersect*. This is something that is quite in line with the original goal of adversarial attacks: generating samples that can fool a model but not a human. Hence, for some datasets, an ε that globally satisfies this assumption may be very *small*.

Robustness to *large* ε indicates how far the data points are from the decision boundary, and larger distance *increases* the guarantees on the model's robustness and thus generalization. But on the other hand being able to train a model with larger ε allows for a larger *exploration* of the input space. UTA attacks have exactly this goal, while ensuring that the network learns "good" properties of the latent space and decision boundary.

This is in sharp contrast to PGD, which is unable to learn a meaningful decision boundary for high-ε regimes, due to the non-isotropic margins which can be more likely to occur on real-world datasets.

Furthermore the UTA attack is based on entropy rather than cross-entropy, so it *unsupervised* as it does not require the ground truth labels. Finally as a note the UTA samples will not always go towards the decision boundary, but in general they will go in areas of high entropy, and depending on the uncertainty estimation method used this means in *unexplored* regions of the training set.

4.2 Image space experiments

In this section are reported some experiments done with UTA applied in the input space of the MNIST [18], the Fashion-MNIST [50] and the CIFAR-10 [21] datasets. The Fashion-MNIST dataset contains 60000 grayscale images of size 28×28 belonging to 10 different classes each one representing a particular type of clothing. The dataset has been split in a training set of 45000 samples, a validation set of 5000 samples and a test set of 10000 samples. On the other hand the CIFAR-10 dataset contains 60000 color images of size 32×32 (times 3 RGB channels) belonging to 10 different classes representing different objects, vehicles or animals. Also in this case the dataset has been split in a training set with 45000 samples, a validation set with 5000 samples and a test set with 10000 samples.

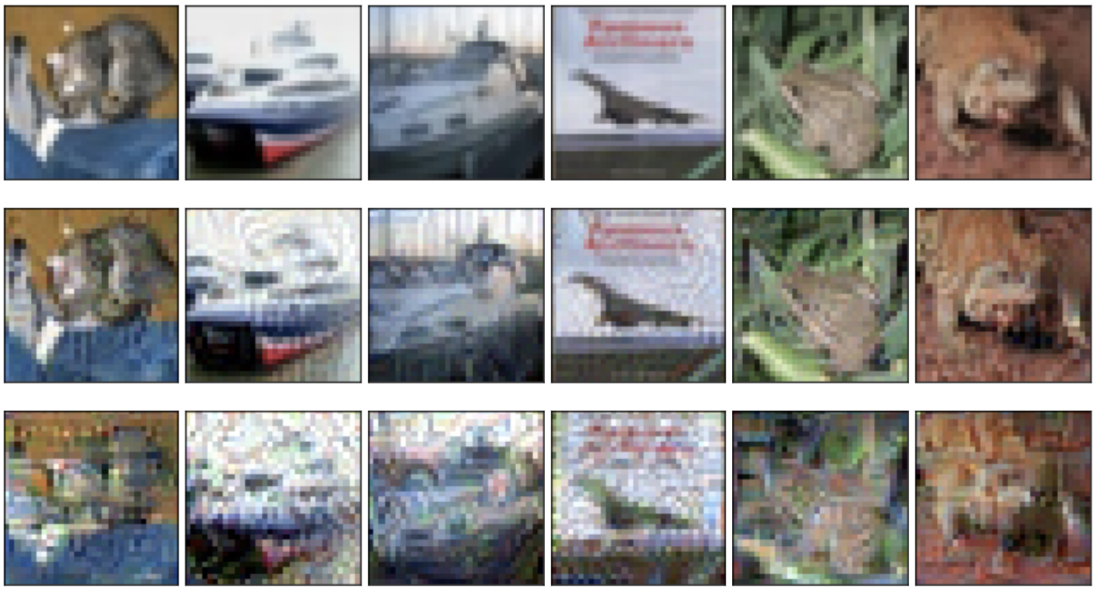


Figure 4.4: Comparison between clean images (*top row*), UTA perturbed images (*middle row*) and PGD perturbed ones (*bottom row*) on the **CIFAR-10** dataset. For UTA and PGD the same setup is used (1000 steps, $\alpha = 0.001$, $\varepsilon = \infty$). This large number of steps is used to verify empirically if the difference between UTA and PGD depicted in Figure 4.2 holds on real-world datasets as well. Contrary to the PGD-perturbed samples, the correct class of the UTA-perturbed ones remains perceptible.

Regarding the split for the MNIST dataset the same described in Chapter 3 has been used. The goal of these experiments is to test the performances of UTA attacks under different datasets and situations. For all the experiments of this section the MC Dropout method is used.

For the MNIST and Fashion-MNIST experiments a LeNet5 model is used [51] with some modifications: (i) the weights of the convolutional and linear layers are initialized with a truncated normal distribution with standard deviation $\sigma = 0.1$ (ii) dropout is added after each layer except the output one with dropout probability $p = 0.2$.

For the CIFAR-10 experiments a ResNet18 network is used [49] modified in the folllowing way[1]: (i) The first 7×7 convolutional layer with stride 2 and padding 3

[1]These modifications (except the dropout additions) follow the ResNet implementation present in `https://github.com/kuangliu/pytorch-cifar` and perform better on the CIFAR-10 dataset.

is changed to a 3×3 convolution with stride 1 and padding 1 (ii) the following max-pooling operation is removed (iii) batch normalization layers are added after convolutional layers (iv) dropout is added after each layer except the output one with dropout probability $p = 0.2$.

As noted in [52] adding dropout after each layer helps in the *stabilization* of adversarial training, especially with FGSM, so the same dropout of $p = 0.2$ is kept during training also for evaluating methods different from UTA. See Appendix B for more details on the architectures used. All results are reported as a *t-confidence interval* over 3 runs.

4.2.1 Robustness on MNIST and Fashion-MNIST experiments

Here are presented the results relative to the robustness of some models to perturbations on the MNIST and Fashion-MNIST datasets. For the former a LeNet5 model is trained for 20 epochs, with a batch size of 512 and using the Adam optimizer with initial learning rate of 1×10^{-3} and weight decay of 1×10^{-3}. In table 4.1 are shown the results obtained by training the model described above with **clean data** only and tested with perturbed data on **MNIST**. As it is possible to see the PGD-50-10 perturbation brings the accuracy of the model to 0 for all the runs, and it is the most powerful attack. FGSM, despite being very fast, is able to bring the accuracy to the network to 0 too. On the other hand UTA attacks, even in the case random restarts are added, are not as powerful in bringing the accuracy of the model down. As it has been shown in the toy example in Section 4.1 UTA samples, unlike PGD ones, do not cross the decision boundary, but lie on areas of *high uncertainty*, where the network is most unsure. It may be questioned whether or not UTA attacks are really *attacks*, in the sense that they are not designed to fool a network, but rather to increase its robustness during AT while keeping good decision boundary properties. In the next sections the advantages of this approach will be shown in more detail in the context of catastrophic overfitting.

In table 4.2 are shown the robustness accuracy results of various models trained with adversarial data against different perturbations on MNIST. The hyperparameters ε and α of the attacks used during training are the same as those used during the robustness evaluation, so $\varepsilon = 0.3$ for all the AT attacks and $\alpha = 0.01$ for PGD-50 and UTA-50.

The clean accuracy of the AT models seems to be quite similar to that obtained with clean training (Table 4.1) even in the case of PGD-50 training that is quite a strong attack. This is not a general phenomenon but it is an *artifact* of the MNIST

Dataset	Perturbation	Test Acc.
MNIST	None	0.990 ± 0.003
	FGSM ($\varepsilon = 0.3$)	0.064 ± 0.015
	PGD-50-10 ($\varepsilon = 0.3, \alpha = 0.01$)	0.000 ± 0.000
	UTA-10-5 ($\varepsilon = 0.3, \alpha = 0.01$)	0.430 ± 0.036
	UTA-50-5-10 ($\varepsilon = 0.3, \alpha = 0.01$)	0.345 ± 0.074

Table 4.1: Adversarial accuracy of attacks performed on **MNIST** at test time to a model trained only with clean data.

dataset [2]. Nevertheless the UTA training seems to lead to a slightly better clean accuracy on average.

All of the AT methods show robustness to FGSM attacks, this is not something unexpected for the PGD-50 AT because this attack is just an *iterative* version of FGSM, but can be quite interesting for the UTA attack. A similar behaviour can be seen also in terms of robustness to PGD-50-10: in this case the FGSM AT performs poorly: this is a sign of catastrophic overfitting happening. Quite surprisingly the UTA attack generalizes well on this much stronger attack and yields to a robustness comparable to that of PGD-50 AT.

Finally, regarding the UTA attacks, they seems to be able to fool FGSM AT by lowering its accuracy to around 50%; FGSM AT is known for not generalizing well to other attacks, as it tend to *overfit* around the specific samples used during training [29]. On the other hand PGD seems quite robust to this particular kind of attack and this is in line with what shown on the toy examples (Section 4.1).

Perturbations / Models	PGD-50	FGSM	UTA-50-5
None	0.97 ± 0.01	0.96 ± 0.02	0.98 ± 0.01
FGSM ($\varepsilon = 0.3$)	0.89 ± 0.02	0.91 ± 0.04	0.86 ± 0.01
PGD-50-10 ($\varepsilon = 0.3, \alpha = 0.01$)	0.75 ± 0.24	0.01 ± 0.00	0.68 ± 0.18
UTA-10-5 ($\varepsilon = 0.3, \alpha = 0.01$)	0.94 ± 0.04	0.54 ± 0.13	0.92 ± 0.07
UTA-50-5-10 ($\varepsilon = 0.3, \alpha = 0.01$)	0.94 ± 0.05	0.50 ± 0.32	0.92 ± 0.08

Table 4.2: Robust accuracy of different models trained with perturbed data against various perturbations on the **MNIST** dataset. UTA-50-5-10 means UTA with 50 steps, 5 model samples and 10 restarts.

[2] more on this here: `https://adversarial-ml-tutorial.org/adversarial_training/`

Robustness in high-ε regime

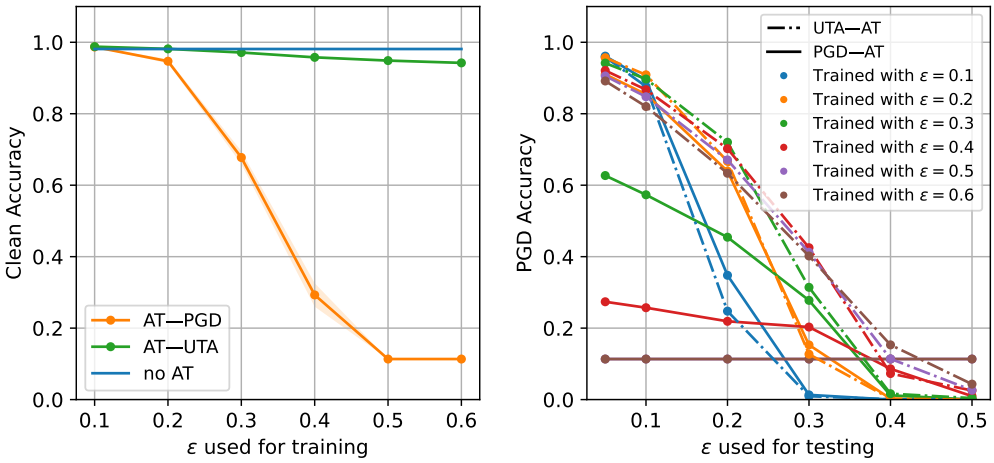
Furthermore an analysis of the robustness of PGD vs. *single model UTA* (only one sample for estimating uncertainty) in a high-ε regime is performed. As discussed in Section 4.1.1 it is not possible to do AT with big ε using standard loss based attacks as this leads to *poor* decision boundaries. UTA on the other hand allows an *uncertainty guided exploration* of the space that can lead to more robust models, while preserving *high* clean accuracy. This is clearly shown in Figure 4.5 where a comparison between UTA and PGD while varying ε is shown.

In particular the training is done for $\varepsilon_{\text{train}} \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ while the robustness testing is carried out for $\varepsilon_{\text{test}} \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$. The step size α of the attack for training is **always fixed** to 0.01 and the number of iterations is set to $k = \frac{\varepsilon}{\alpha}$. This is done to ensure that *enough steps* are done to explore the ε-ball. For testing the number of iterations is always fixed to $k = \frac{\varepsilon}{0.01}$, while α is computed using a *triangular scheduler* as in [53]. For steps between 0 and $k/2$, alpha is increased linearly from 0.01 to $\max(0.01, \varepsilon_{\text{test}}/5)$, for steps between $k/2$ and K, alpha is decreased linearly to 0.01. The step-size scheduler during testing is to prevent PGD from *getting stuck* in a local minimum close to the original data point. The network used is the same LeNet5 used in the previous section, but this time training is done with a batch size of 128 and for 100 epochs.

As it is possible to see from subfigure 4.5a the clean accuracy of the models trained with UTA is definitely higher than the PGD-AT ones and decreases slower as ε increases. In subfigure 4.5b is shown the PGD accuracy of the models trained with PGD and *single model UTA* (dashed lines) samples. Is it possible to note that as ε_{train} increases, UTA models keep a certain level of robustness to PGD, while PGD-AT model fail in this. Importantly, when training using UTA with a larger ε, it is possible to see that the resulting robustness is competitive also when evaluated on *smaller* ε. Hence, even if targeting solely robustness on *small* ε, training with UTA and large ε can provide better small-ε robustness as well.

Quite interestingly the clean accuracy shown for PGD-AT varying the ε used for training degrades quite *fast* and for $\varepsilon = 0.3$ there is quite a substantial drop, reaching a value definitely lower than the one seen in Table 4.2. This may be related to the fact that in these experiments a *larger* number of epochs is used, thus the model is in certain sense *overfitting* more the training attack and this leads to poor decision boundaries. Furthermore note how in the high-ε regime case even for $\varepsilon = 0.3$ UTA-AT seems *more robust* to PGD testing than PGD-AT. This may be related to the fact that in these experiments random restarts are not used and experimentally throughout this thesis it has been noted that UTA is slightly *less* robust to PGD when random restarts are added during testing.

The same analysis has been repeated with the **Fashion-MNIST** dataset to see if the advantages of UTA in high-ε regime still holds in a more challenging



(a) Clean accuracy on test dataset **(b)** PGD robustness on test dataset

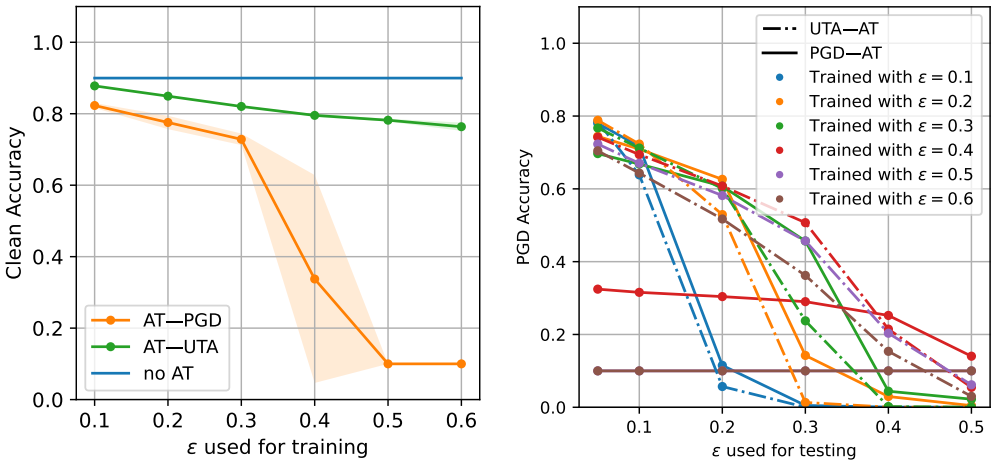
Figure 4.5: Comparison between PGD and *single model UTA* on **MNIST**, results are averaged over 3 runs. (a): accuracy on the unperturbed test dataset, for varying $\varepsilon_{\text{train}}$. (b): PGD robustness, for varying $\varepsilon_{\text{test}}$ (x-axis), where dashed-dotted curves are UTA, and solid curves are PGD.

setup. For these experiments the setting used is the same presented for the MNIST experiments in high-ε regime. Results are shown in Figure 4.6 for $\varepsilon_{\text{train}} \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$ and $\varepsilon_{\text{test}} \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$. Also in this case UTA attacks lead to better clean accuracy and robustness for high-ε training.

4.2.2 Robustness on CIFAR-10 experiments

The same experiments on the robustness of various models against the FGSM, PGD and UTA perturbations are repeated using the CIFAR-10 dataset. The ResNet18 model with dropout is trained for 90 epochs using *cyclic learning rate* schedule as in [53] with a maximum learning rate of 0.2 for the FGSM and PGD-10 attacks and 0.1 for the UTA-10-5 attacks. The value of ε used is of $\frac{8}{255}$ while α for PGD and UTA is set to $\alpha = \frac{\varepsilon}{4}$ (both for training and testing) following [8].

In table 4.3 are reported the results relative to the clean and robust accuracy of a model trained only with clean data against various attacks. As it is possible to see all the attacks are quite good in reducing the accuracy of the network and even the UTA attack succeded in obtaining 0 adversarial robustness. This is something quite *different* than what seen for MNIST in the previous section. Classifying MNIST digits is quite a simple problem for modern convolutional networks and, as pointed out in Section 3.1.1, models trained on the MNIST dataset tend to have *low uncertainty*, compared to CIFAR-10 models, hence the UTA attack at test



(a) Clean test accuracy **(b)** PGD robustness

Figure 4.6: Comparison between PGD and single *single model UTA* on **Fashion-MNIST**, results are averaged over 3 runs. (a): accuracy on the unperturbed test dataset, for varying $\varepsilon_{\text{train}}$. (b): PGD robustness on the test data points, for varying $\varepsilon_{\text{test}}$ (x-axis), where dashed-dotted curves are UTA, and solid curves are PGD.

Dataset	Perturbation	Test Acc.
CIFAR-10	None	0.943 ± 0.019
	FGSM ($\varepsilon = \frac{8}{255}$)	0.147 ± 0.037
	PGD-50-10 ($\varepsilon = \frac{8}{255} \alpha = \frac{2}{255}$)	0.000 ± 0.000
	UTA-50-5-10 ($\varepsilon = \frac{8}{255}, \alpha = \frac{2}{255}$)	0.000 ± 0.000

Table 4.3: Adversarial accuracy of attacks performed on **CIFAR-10** at test time to a model trained only with clean data. UTA-50-5-10 means UTA with 50 steps, 5 model samples and 10 restarts.

time may have a weaker effect on MNIST in attacking a clean model compared to CIFAR-10.

In table 4.4 are shown the robustness results under several attacks of different adversarially trained models. As it is possible to see, also in this case FGSM AT suffers from *catastrophic overfitting*: the robust PGD-50-10 and UTA-50-10 accuracy is 0 while the FGSM accuracy (the same attack used for AT) is very high, higher than the clean one, so the model is overfitting the attack. The PGD-10 AT model on the other hand seems to be quite robust to all the other attacks, while having a lower clean accuracy compared to the other AT methods. Finally

the UTA-10-5 AT while being slightly less robust to PGD-50-10 and *FGSM* than PGD-10 AT has higher clean accuracy values.

Quite interestingly the UTA-10-5 AT model is definitely less robust to the UTA-50-10 attack compared to the PGD-10 AT model (around 16% accuracy drop). As explained in the previous sections UTA-AT can be seen as a *less disruptive* version of PGD-AT in which ε is automatically adapted to not alter too much the decision boundary properties. So testing a UTA-AT model against PGD with the same ε used both for training and testing would be like testing against an attack that has a **maximum** magnitude equal to the ε used during AT and this could explain the higher robustness of PGD-10 AT compared to UTA-10-5 AT. This actually is not a big problem, because as shown in the previous section UTA attacks do not have the goal of *fooling* a network, but of helping the network to be robust and, at the same time, to generalize to other attacks (without overfitting) while preserving higher clean accuracy.

Perturbations / Models	PGD-10	FGSM	UTA-10-5
None	0.82 ± 0.06	0.84 ± 0.07	0.88 ± 0.04
FGSM ($\varepsilon = \frac{8}{255}$)	0.54 ± 0.04	0.91 ± 0.04	0.50 ± 0.07
PGD-50-10 ($\varepsilon = \frac{8}{255}, \alpha = \frac{2}{255}$)	0.49 ± 0.06	0.00 ± 0.00	0.46 ± 0.06
UTA-50-5-10 ($\varepsilon = \frac{8}{255}, \alpha = \frac{2}{255}$)	0.76 ± 0.01	0.02 ± 0.02	0.60 ± 0.02

Table 4.4: Robust accuracy of different models trained with perturbed data against various perturbations on the **CIFAR-10** dataset.

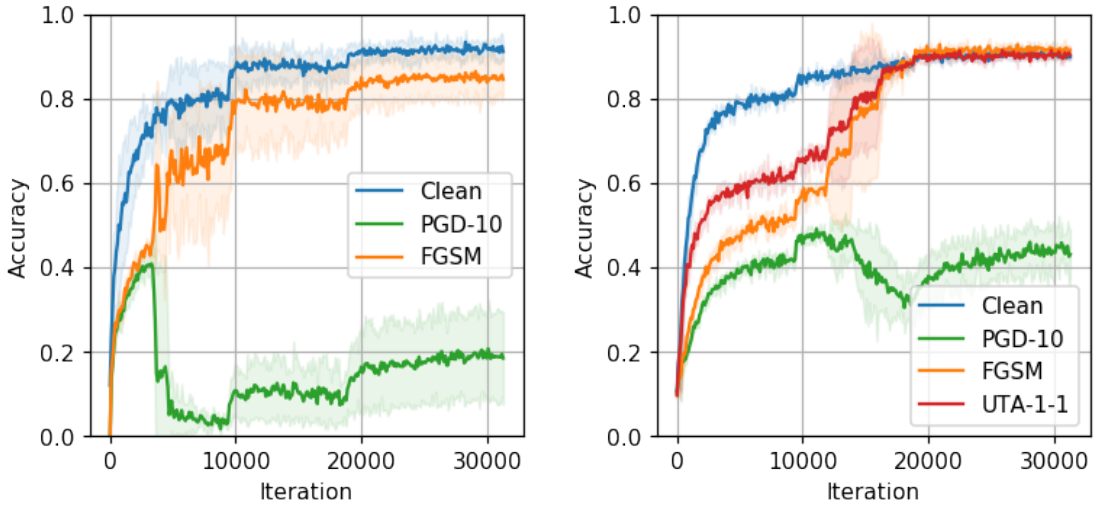
4.2.3 Catastrophic Overfitting experiments

As described in Section 2.2.2 *catastrophic overfitting* is a phenomenon that happens when adversarial training is performed using a certain attack, but the testing is done against a stronger adversarial attack. In this setting it could happen after a certain number of iterations that the robust accuracy evaluated against the AT attack jumps to a very high value, while the robust accuracy against the stronger attack drops to almost zero. This is visible in subfigure 4.7a. In this case the model is adversarially trained with FGSM and tested against PGD-10.

The model used is the ResNet18 described in the previous section and has been adversarially trained for 200 epochs with a batch size of 256 using the Stochastic Gradient Descent (SGD) optimizer with Nesterov momentum. The initial learning rate was of 0.1 decayed by a factor of $\frac{1}{5}$ after $60, 120, 160$ epochs. The FGSM attack used in Figure 4.7a is done with $\varepsilon = \frac{8}{255}$; for the testing PGD-10 attack the hyperparameters $\varepsilon = \frac{8}{255}$ and $\alpha = \frac{\varepsilon}{4}$ are used. As it is possible to note

Catastrophic Overfitting (CO) happens at around iteration 4700. Before CO the PGD-10 accuracy follows the FGSM one, but after CO the first goes to almost zero, while the second *increases drastically*.

On the other hand in subfigure 4.7b is shown adversarial training done with the UTA-1-1 (*single model*) attack using $\varepsilon = \frac{8}{255}$ and $\alpha = \varepsilon$. This is a very *fast* version of UTA that is comparable to FGSM in terms of speed. The training is done in the exact same setting as in subfigure 4.7a, with the difference that now UTA is used for AT. Even in this setting there is a drop in PGD-10 accuracy at a certain point, but this drop happens *later* during training with respect to FGSM AT and is not as catastrophic.



(a) FGSM adversarial training **(b)** UTA-1-1 adversarial training

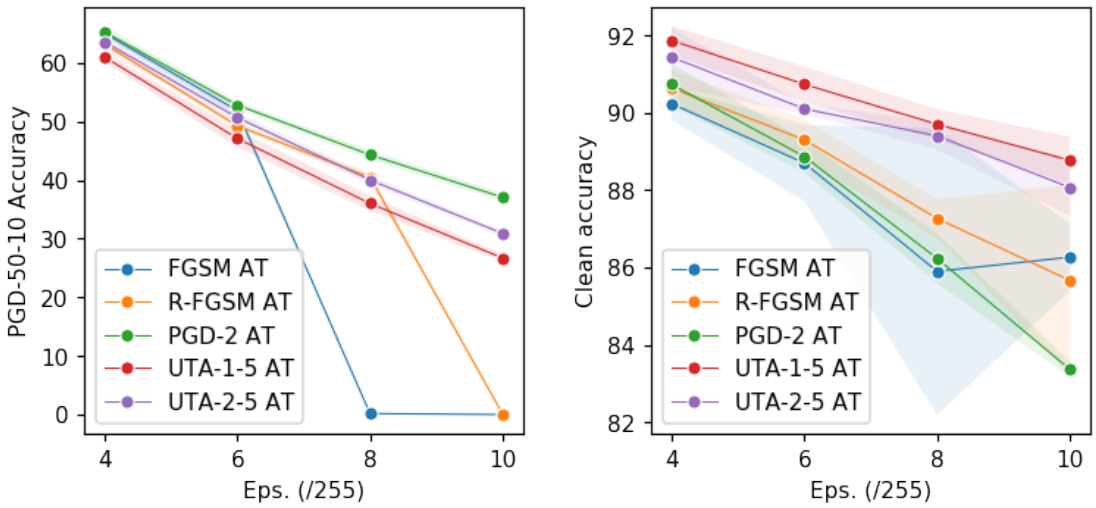
Figure 4.7: Catastrophic overfitting experiments with FGSM and UTA on the **CIFAR-10** dataset. Results are averaged over 3 runs and accuracies are calculated on the evaluation set. (a): training with FGSM–with $\varepsilon = 8/255$; and testing against PGD–10 with $\varepsilon = 8/255$ and $\alpha = \varepsilon/4$. CO occurs at around iteration 4700. (b): training with UTA with 1 step (fast version), 1 sampled model and $\alpha = \varepsilon = 8/255$; testing against PGD-10 ($\varepsilon = 8/255$, $\alpha = \varepsilon/4$) and FGSM ($\varepsilon = \alpha = 8/255$). It is possible to observe that UTA is more robust to CO relative to 4.7a.

In Figure 4.8 the robustness of different AT methods are shown. In order to reach faster convergence in these experiments a Resnet18 model with dropout has been trained with the same setting used in Section 4.2.2, but with a maximum learning rate of 0.2 for the FGSM, R-FGSM and PGD-2 attacks and 0.1 for the UTA-1-5 and UTA-2-5 attacks.

As in [28] the evaluation of these AT methods is done against the PGD-50-10

attack, i.e. with 50 steps and 10 random restarts. Evaluation is done on the test set at the end of the training. Results are shown for an increasing magnitude of the attack $\varepsilon \in \{\frac{4}{255}, \frac{6}{255}, \frac{8}{255}, \frac{10}{255}\}$; differently with respect to what presented in Figure 4.5 the same ε is used for training and testing. The α used is $\alpha = \frac{\varepsilon}{4}$ for PGD and UTA and $\alpha = 1.25\varepsilon$ for R-FGSM as suggested in [8]. As it is possible to note, FGSM and R-FGSM AT *suffer* from catastrophic overfitting as ε increases, even if R-FGSM is slightly more robust and suffers this problem for a larger ε. On the other hand UTA attacks are *more robust* and for the values of ε analyzed they do not suffer from CO.

In subfigure 4.8b on the other hand, instead of the robust is shown the clean accuracy at the end of the training. Note how UTA-AT yields to *higher clean accuracy* than FGSM, R-FGSM and PGD-AT.

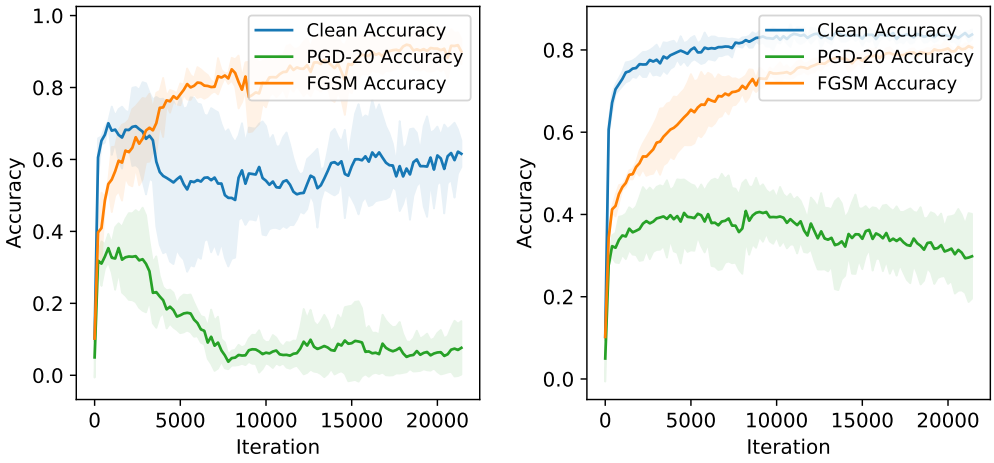


(a) Robust accuracy **(b)** Clean accuracy

Figure 4.8: Robust evaluation of different AT methods against PGD-50-10 after 90 epochs varying ε on the **CIFAR-10** dataset. Results are averaged over 3 runs. (a): PGD-50-10 comparison of different AT methods for different values of the perturbation radius ε. UTA methods, albeit being marginally less robust to PGD-50-10, do not suffer from CO even for large values of ε. (b): clean accuracy comparison of different AT methods with different values of the perturbation radius ε. UTA methods lead to higher clean accuracy than PGD methods.

For confirming what seen in figure 4.7 the same experiments have been repeated with the Fashion-MNIST dataset, so using a LeNet model trained with the hyper-parameters described in Section 4.2.1. Results are shown in Figure 4.9: as it is possible to see, FGSM AT leads to catastrophic overfitting soon after the training

starts, while this does not happens for UTA-1-1 AT: there is, after a certain while, a *decreasing trend* in PGD-20 accuracy, but definitely not a drop. Furthermore note how in case of FGSM AT there is also a drop in clean accuracy too, while this is not the case for UTA-1-1 AT. In this case for FGSM and UTA-1-1 AT $\varepsilon = 0.2$ is used, while for testing the PGD-20 attack with $\varepsilon = 0.2$ and $\alpha = 0.01$ is applied.



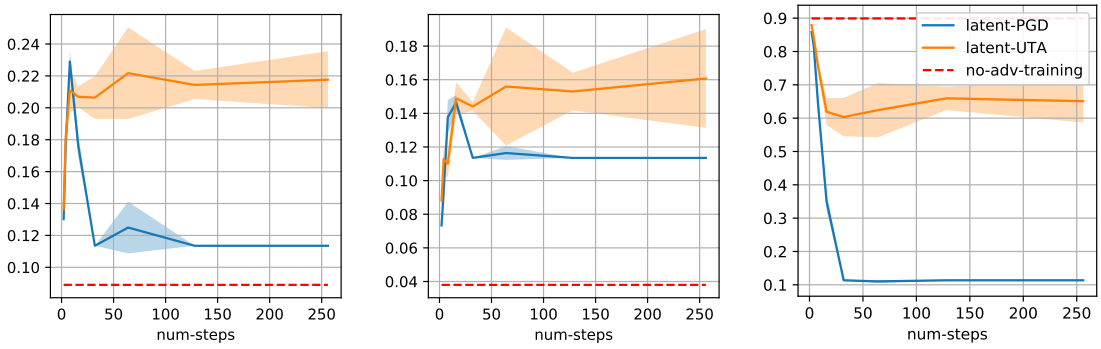
(a) FGSM training **(b)** UTA-1-1 training

Figure 4.9: Catastrophic Overfitting (CO) on the **FashionMNIST** dataset using a LeNet model; results are averaged over 3 runs. (a): training with FGSM using a step size $\alpha = 0.2$, $\varepsilon = \alpha$; and testing against PGD-20 with $\varepsilon = 0.2$ and $\alpha = 0.01$. Soon after the beginning of the training the FGSM accuracy suddenly jumps to very high values while the PGD-20 accuracy approaches 0. Note also how the clean accuracy decreases and becomes lower than the FGSM one after CO. (b): training with UTA-1-1 (single model) using a step size $\alpha = 0.2$, $\varepsilon = \alpha$; and testing against PGD-20 with $\varepsilon = 0.2$ and $\alpha = 0.01$. While the PGD robustness for UTA decreases to some extent, the drop is not as large as for FGSM AT. Note also how UTA-1-1 AT leads to higher clean accuracy.

4.3 Latent Space Experiments

Up until now only perturbations carried out in the image space have been considered. In this section will be reported the experiments related to attacks applied in the *latent space* of a model. Let $\mathcal{E}(\boldsymbol{x})$ be a *pre-trained* encoder that converts a generic input image to its latent space representation $\boldsymbol{z}$. In the following experiments this encoder is pre-trained on the *clean* dataset and kept frozen during adversarial training. Let then $\mathcal{C}_\omega$ be a classifier that takes as input the latent space representation

$\boldsymbol{z}$ of the data. Perturbations are applied to the latent space representation $\boldsymbol{z}$ rather than on the input $\boldsymbol{x}$ in order to obtain *on-manifold* attacks. Experiments are done both one the MNIST dataset and on CIFAR-10. For both the settings the encoder is a two layer convolutional neural network while the classifier is a simple two layers fully connected neural network. In the latent space the choice of the magnitude of the perturbation ε is *arbitrary* since the distances are arbitrary too. In the reported experiments for the latent space attacks a $\varepsilon = 5$ and a $\alpha = 0.05$ have been used. In Figure 4.10 are shown the results in terms of robust accuracy obtained by evaluating some models trained with latent space PGD-k and UTA-k-10 for $k \in \{2,4,8,16,32,64,128,256\}$ and tested against latent PGD-10 (subfigure 4.10a) and latent PGD-50-10 (subfigure 4.10b) [48]. Both the testing attacks were applied in the latent space with $\varepsilon = 5$ and $\alpha = 0.01$. Uncertainty is computed with the MC Dropout method. The clean accuracy of these robust models is also shown in subfigure 4.10c. As it is possible to see the models trained using latent PGD lose their robustness as the number of steps increases. For these models also the clean accuracy degrades with increasing number of steps AT. UTA-AT on the other hand in the latent space is able to preserve its *robustness* levels as the number of steps increases and the clean accuracy stays high. Note that adversarial training using latent attacks leads to less robustness to latent space space attack testing compared to the robustness obtained by doing AT on image space attacks. Despite that, they provide a certain degree of robustness with respect to non adversarial training (red dotted line in Figure 4.10).



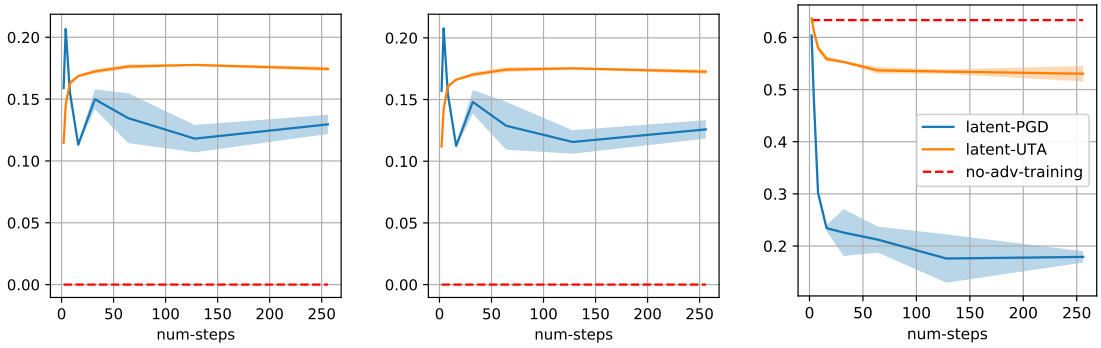
(a) PGD-10 accuracy **(b)** PGD-50-10 accuracy **(c)** Clean accuracy

Figure 4.10: Robustness of latent space UTA attacks compared to latent space PGD attacks on the **MNIST** dataset. (a): robust accuracy to PGD-10 when the number of steps used for AT increases. (b): robust accuracy to PGD-50-10 when the number of steps used for AT increases. (c): clean accuracy of AT models as the number of steps increases. Note how UTA-AT preserves higher values of clean accuracy compared to PGD-AT and is more robust to latent space attacks.

In Figure 4.11 the same experiments are repeated using the CIFAR-10 dataset and testing against PGD-10 and PGD-50-10. As it is possible to see also in this case the UTA latent attacks lead to more robust models with higher clean accuracy compared to latent PGD attacks and latent UTA-AT models preserve their robustness for an increasing number of training steps confirming the intution in Section 4.1 that UTA attacks acts like a more adaptive version of PGD and so they are less dependent on the particular hyperparameters chosen.

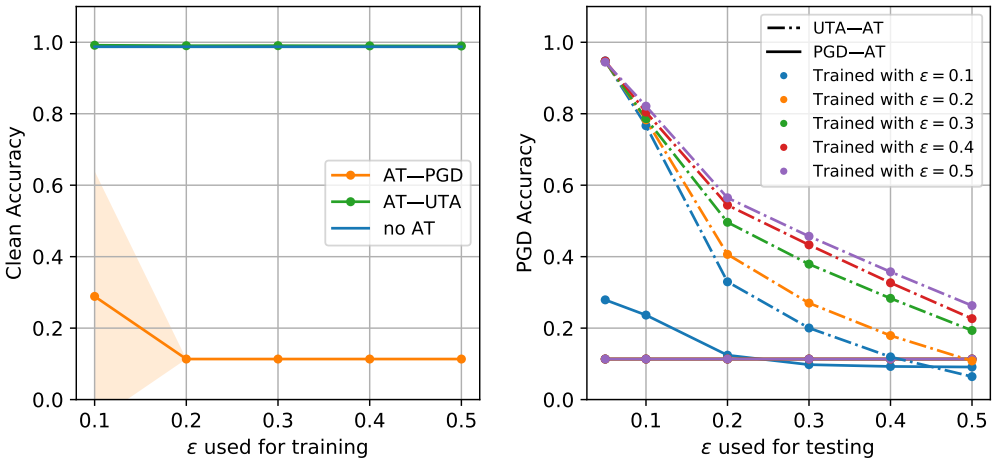
4.3.1 Latent robustness in the high-ε regime

Furthermore an evaluation of how latent UTA attacks behave compared to latent PGD in an high-ε regime is carried out on the **MNIST** dataset. For this specific case the network used is the same LeNet introduced in Section 4.2.1 trained with the same setting used for the MNIST dataset in high-ε regime. In this specific case the Monte Carlo Dropout method is *not* used. Uncertainty is on the other hand estimated using the *deep ensemble* approach, with 5 models. This method works in a very similar way as Monte Carlo Dropout, with the only difference that the various networks of the ensemble are *independent* and not sampled randomly at forward time. The LeNet network has been split in an *encoder* and a *classifier* part. The encoder contains the convolutional layers of the network and the first fully connected, while the classifier the last two fully connected layers. Everything is



(a) PGD-10 accuracy **(b)** PGD-50-10 accuracy **(c)** Clean accuracy

Figure 4.11: Robustness of latent space UTA attacks compared to latent space PGD attacks on the **CIFAR-10** dataset. (a): robust accuracy to PGD-10 when the number of steps used for AT increases. (b): robust accuracy to PGD-50-10 when the number of steps used for AT increases. (c): clean accuracy of AT models as the number of steps increases. Note how also for CIFAR-10 UTA-AT preserves higher values of clean accuracy compared to PGD-AT and is more robust to latent space attacks.



(a) Clean test accuracy **(b)** PGD robustness

Figure 4.12: Comparison between latent PGD and latent *UTA-1* with a 5 models ensemble on **MNIST**, results are averaged over 3 runs and testing is done in the image space. (a): accuracy on the unperturbed test dataset, for varying $\varepsilon_{\text{train}}$. b image space PGD robustness on the test data points, for varying $\varepsilon_{\text{test}}$ (x-axis), where dashed-dotted curves are UTA, and solid curves are PGD.

being trained together *without* pre-training the encoder first. The training is done for $\varepsilon_{\text{train}} \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ with $\alpha = 0.01$ and the number of steps adjusted to be $\frac{\varepsilon}{\alpha}$. Since attacks are done in the latent space the values of ε are arbitrary and depend on the architecture chosen; *by chance* in this case it appeared that the good values for ε in this latent space are the same as the ones used in image space. The testing is done for $\varepsilon_{\text{test}} \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ a number of steps $\frac{\varepsilon}{0.01}$ and α scheduled in a triangular way as described in Section 4.2.1. The evaluation is done against **image space attacks**. Results both in terms of clean and robust accuracy are shown in Figure 4.12. As it is possible to see UTA attacks when applied in the latent space lead to higher clean accuracy and are definitely more robust to image space attacks. This is in line with the argumentation that PGD-AT leads to poor properties of the latent space as in the specific case in which attacks are applied in this latent representation is not able to generalize to *image space robustness*.

4.4 Discussion

In this chapter an extensive analysis on how UTA attacks differ with respect to standard loss-based methods has been done under several settings and conditions. Quite interestingly it has been found that the proposed method (UTA) leads to

an improved *generalization-robustness* tradeoff with respect to state of the art adversarial methods. UTA definitely improves the *state of the art* in terms of fast adversarial training and allows for a better exploration *guided by uncertainty* of the perturbation space. This allows to obtain better robustness without losing generalization, thus being able to *mitigate* the common adversarial problems that affected the deployment of deep learning models to real world applications. In this chapter a general formulation of UTA has been provided that permits to extend the experiments done here to several different uncertainty estimation methods and metrics. That is exactly the goal of the next chapter: provide an extensive analysis of UTA under a *completely different* uncertainty estimation technique, in order to see if UTA brings to some advantages also in this setting.

Chapter 5

Adversarial Evaluation of Deterministic Uncertainty Estimation

The results presented in Chapter 4 showed how UTA-AT can be an advantageous method to train a network that is robust to different state of the art adversarial attacks and to catastrophic overfitting, while keeping high clean accuracy. However the method used for estimating uncertainty, MC Dropout, is a quite *approximate* method and recent solutions have been proposed to compute uncertainty in a different, *deterministic*, way [14]. The goal of this chapter is to investigate how UTA attacks and adversarial training in general behave when the uncertainty estimation method used is the Deterministic Uncertainty Estimation (DUE) introduced in Section 2.3.4. The formulation of adversarial training using UTA is the same as that introduced in chapter 4. In DUE the number of samples t used for computing uncertainty and prediction is an *hyperparameter* of the network, rather than of the attack, and is always kept the same ($t = 32$ as suggested in the paper), so in this chapter the UTA nomenclature *loses* the information about t, keeping only the information about the number of steps k and the number of restarts n when expressed, e.g. UTA-5 means UTA with $k = 5$ steps, exactly as in the PGD nomenclature. The sampling procedure for making predictions, as implemented in the original DUE paper [14], is done regardless of adversarial training, therefore the computational complexity of PGD and UTA in the case they are applied on a DUE model are exactly the same.

Furthermore, towards the end of this chapter some experiments will be made also using a modified version of DUE in which spectral normalization is turned off. This modified method falls in the more general family of *Deep Kernel Learning* (DKL) methods [38].

5.1 An introductory toy example

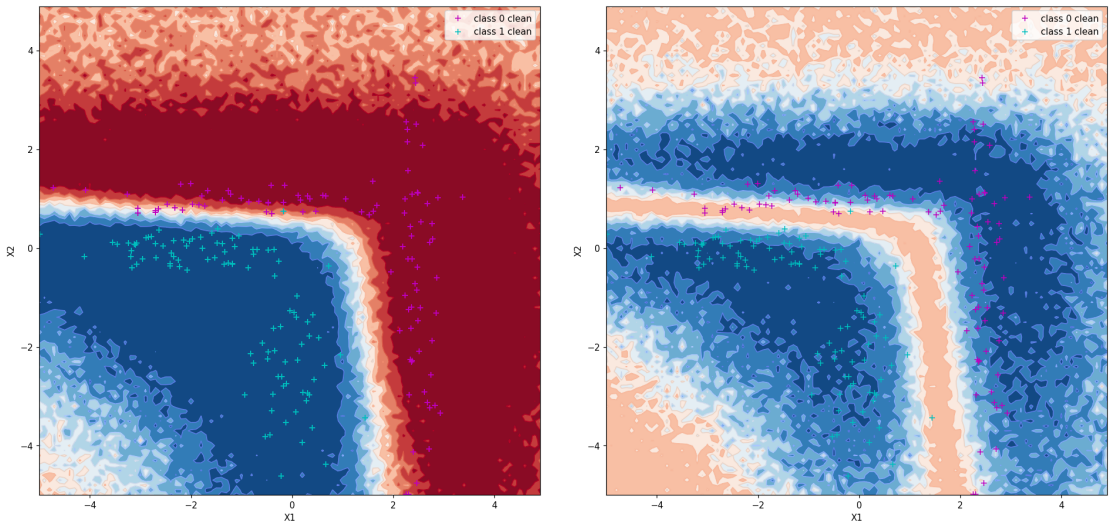
In this section an intuitive visualization of the decision boundaries and uncertainty, generated by training with the DUE method will be presented using the same non-isotropic toy dataset introduced in Section 4.1. In Figure 5.1 are shown the decision and entropy boundary of a model trained with DUE on the 2-dimensional toy dataset. The network used as a feature extractor is a fully connected residual network with spectral normalization, implemented like in [14] and similar to the one used in [54]. The first FC layer maps the input to an initial latent space, without activation function. Then there are 4 additional residual blocks composed by a fully connected plus the residual such that the output of the block is $\boldsymbol{x}' = \boldsymbol{x} + f(\boldsymbol{x})$, with f representing a linear mapping followed by a ReLU activation function. Spectral normalization is used in the linear mapping implemented as in [55]. The SGD optimizer is used with an initial learning rate of 1×10^{-1} with momentum 0.9 and weight decay equal to 4×10^{-4}. Each layer used has 128 neurons, this is the dimensionality of the latent space; 2 inducing points are used. Regarding the Gaussian Process an additive Gaussian noise $\sigma = 0.1$ has been used to model the variance of the data likelihood, with an RBF kernel. The uncertainty is estimated using entropy as in the DUE paper [14].

In Figure 5.1 it is possible to see the decision boundary obtained with the DUE method; this is considerably *different* with respect to that obtained using an MCD network (see Figure 4.2). A similar pattern can be seen in the uncertainty estimation: in the DUE case it seems like the network is not only unsure in regions that are near the decision boundary, but also in regions in which there are not many training points. This is particularly interesting because an adversarial attack targeted at maximizing the uncertainty of the model will not only move the samples to be closer to the decision boundary, but also in areas in which there is low density of training points, thus generating some *outlier examples*.

This intuition is confirmed by the results presented in Figure 5.2, where are shown the PGD (subfigure 5.2a) and UTA (subfigure 5.2b) attacks applied to a model trained with clean data only. Both attacks use 50 steps and 10 random restarts, with an $\varepsilon = 1$ and $\alpha = 0.1$. As it is possible to see from the figure some of the UTA samples are moved towards the decision boundary, but do not cross it. On the other hand some other samples are pushed towards unexplored region of the dataset, because they have high uncertainty. This lastly described behaviour can be noted to a minor extent also by looking at the PGD perturbed samples.

PGD Adversarial Training

In Figure 5.3 are shown the decision boundary and the entropy of a model trained adversarially with PGD-10 ($\varepsilon = 1$, $\alpha = 0.1$). This decision boundary is definitely



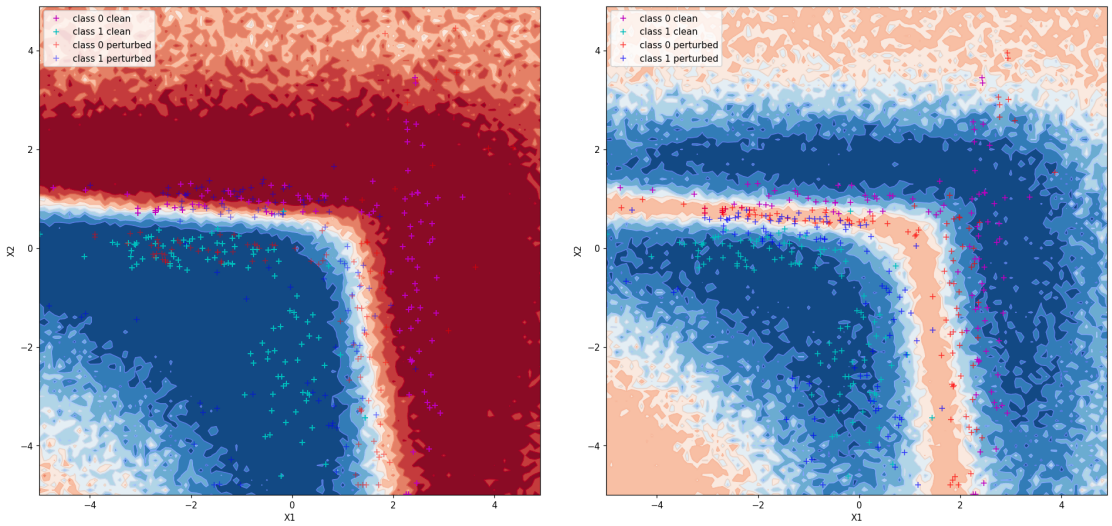
(a) DUE clean training decision boundary **(b)** DUE clean training entropy

Figure 5.1: Decision boundary and entropy of a classifier trained with clean data using the DUE method on a toy dataset. (a): decision boundary of the model. (b): entropy of the model. Note how entropy is high also unexplored areas and not only near the decision boundary.

different than the one shown in Figure 5.1 and the model is not able to learn the dataset properly, a lot of clean samples end up being misclassified. Furthermore by looking at subfigure 5.3b it is possible to see how the model, because of adversarial training, loses the ability to estimate in a correct way the entropy, that is high everywhere except in a small region with an high density of points belonging to class 0.

UTA Adversarial Training

The same experiments shown in Section 5.1 are repeated here using UTA-10-AT and are shown in Figure 5.4. In this case the model is able to learn the dataset better and the decision boundary is much more similar to that obtained by training with clean data, compared to PGD. The same can be said regarding the uncertainty estimation of the model shown in subfigure 5.4b. Interestingly the model seems to have lower values of uncertainty in unexplored areas of the dataset, compared to the clean model in Figure 5.1. This may be due to the fact that the training samples used are perturbed with UTA that, as shown in Figure 5.2, tends to move the samples towards unexplored areas, hence if a models sees these samples during training these will be able to explain the uncertainty around those regions (remember that entropy is a measure of epistemic uncertainty).



(a) PGD-50-10 attack on DUE

(b) UTA-50-10 attack on DUE

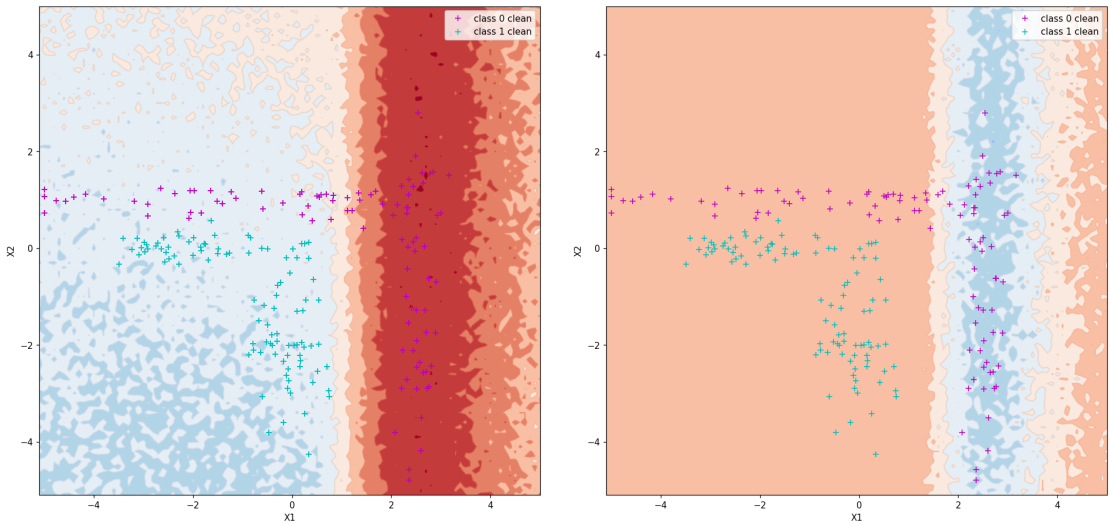
Figure 5.2: PGD and UTA attack on a DUE model trained with clean data on the toy dataset. No adversarial training at all is done in this figure. (a): PGD samples, crafted to fool the clean model, plotted on top of the decision boundary. Note how these adversarial samples cross the boundary and go to the other side. (b): UTA samples plotted on top of the entropy of the model. Not how the UTA samples, having the goal of maximizing entropy do not cross the boundary, but go towards it. Furthermore some samples are pushed to unexplored areas of the space.

5.2 CIFAR-10 experiments

The robustness experiments described in the previous sections are done here using the DUE method on the CIFAR-10 dataset. In this setting the network used as feature extractor is a Wide Res-Net (WRN) [56] up to the last linear layer. The version used has a depth of 40 and a widening factor 4, following one of the suggested implementations in the original paper [56]. Training has been done for 100 epochs with a batch size of 128 using SGD with momentum 0.9 and an initial learning rate of 0.1 multiplied by 0.2 at 40, 60, 80 epochs. Regarding the spectral normalization, this was implemented as in [14]. All results are reported as a *t-confidence interval* over 3 runs.

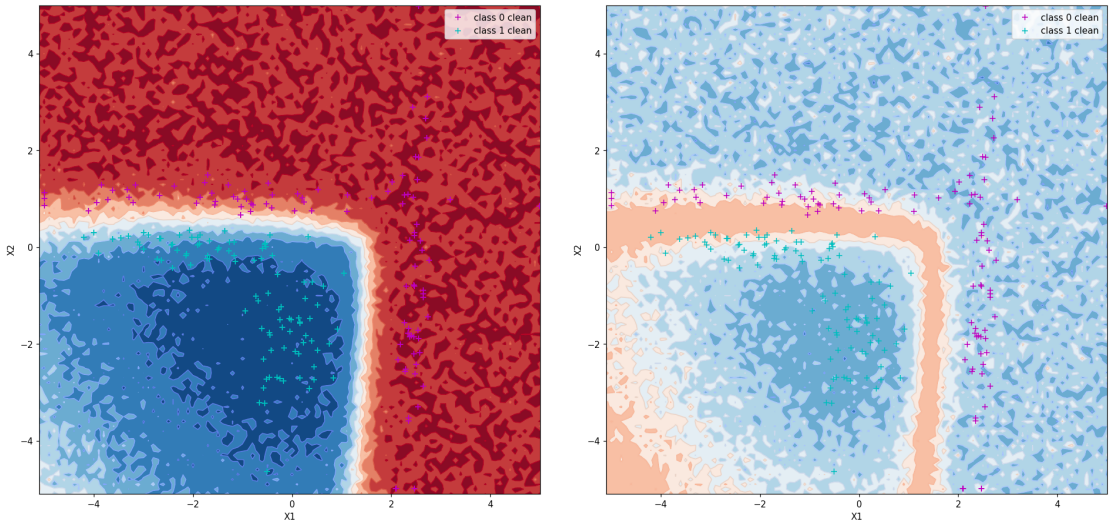
5.2.1 Robustness of PGD against UTA

The experiments were done in order to compare UTA-AT and PGD-AT under the same setting. For all the training and testing attacks the values of $\varepsilon = \frac{8}{255}$ and $\alpha = \frac{\varepsilon}{4}$ have been used. The robust evaluation has been done against PGD-50-10.



(a) Decision boundary of a PGD-AT model. **(b)** Entropy boundary of a PGD-AT model.

Figure 5.3: Decision boundary (a) and entropy (b) for a PGD-10 AT model on the toy dataset. Note how PGD-AT leads to a very bad decision boundary and the model is not able to learn the dataset correctly.



(a) Decision boundary of a UTA-AT model. **(b)** Entropy boundary of a UTA-AT model.

Figure 5.4: Decision boundary (a) and entropy (b) for a UTA-10 AT model. Note how in this case UTA-AT preserves a good decision boundary and the model is able to learn the dataset.

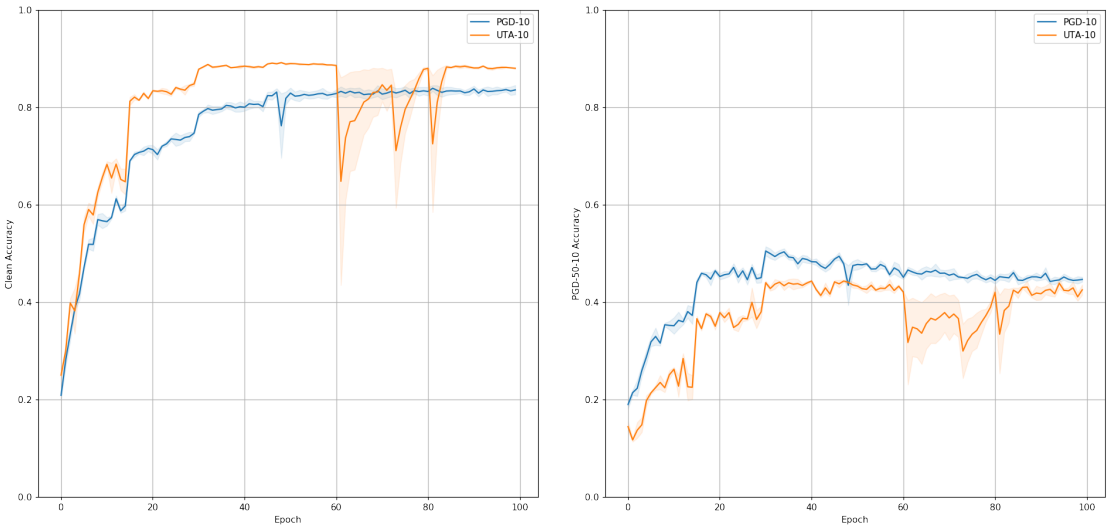
In Figure 5.5 are shown the results both in terms of clean and adversarial accuracy of some models trained with PGD-10 vs UTA-10 using the standard DUE method as described in the original paper [14]. Results are averaged over three runs. As it is possible to see PGD-10 AT seems to lead to better robustness than UTA-AT that, on the other hand, leads to a better clean accuracy. Interestingly there are some very sharp spikes during UTA-AT in which there is an accuracy drop, both in the clean and adversarial case, making the training with UTA samples less stable compared to PGD-AT. These results are actually quite in contrast with what shown in the toy example: in that case PGD-AT led to a very poor decision boundary that basically prevented the model to learn correctly the dataset (Figure 5.3).

In the previous discussions (Chapter 4) it was argued that PGD-AT may prevent the network from learning *good* latent spaces representations and decision boundaries. But in these DUE experiments (Figure 5.5) on the other hand it seems like there is a *constraint* that is ensuring PGD-AT to lead to good latent spaces such that the network can become more robust. It is argued here that the *spectral normalization* applied by the DUE method is responsible for this as the goal of this technique applied here is exactly to ensure good latent spaces properties. This in a certain way can be a factor that makes PGD-AT more stable.

On the other hand, looking at UTA-AT it seems like this spectral normalization factor is stopping the network from exploring in a complete way the space, thus making it a lot more *constrained*.

Adversarial training without spectral normalization

To confirm the intuition expressed in the previous section the same experiments were repeated keeping the same setting, but this time turning off spectral normalization. In this way the uncertainty estimation method used is much more similar to a general Deep Kernel Learning technique [38]. In Figure 5.6 are shown the results of the experiments done in this setting. First of all UTA-AT appears more stable when spectral normalization is *not* active and the accuracy drops seen in Figure 5.5 are not seen here. Moreover UTA-AT seems to be also more robust to PGD-50-10 attacks when spectral normalization is not active. PGD-AT on the other hand seems to lose part of its *robustness* when spectral normalization is off. The adversarial accuracy reaches a maximum that is on average higher than that reached in subfigure 5.5b, but then it experience a *downward* trend indicating that the model is overfitting the attack. At around epoch 60 the PGD-AT models become *less* robust than its UTA counterpart. Also in this case regarding the clean accuracy, UTA is definitely better than PGD.



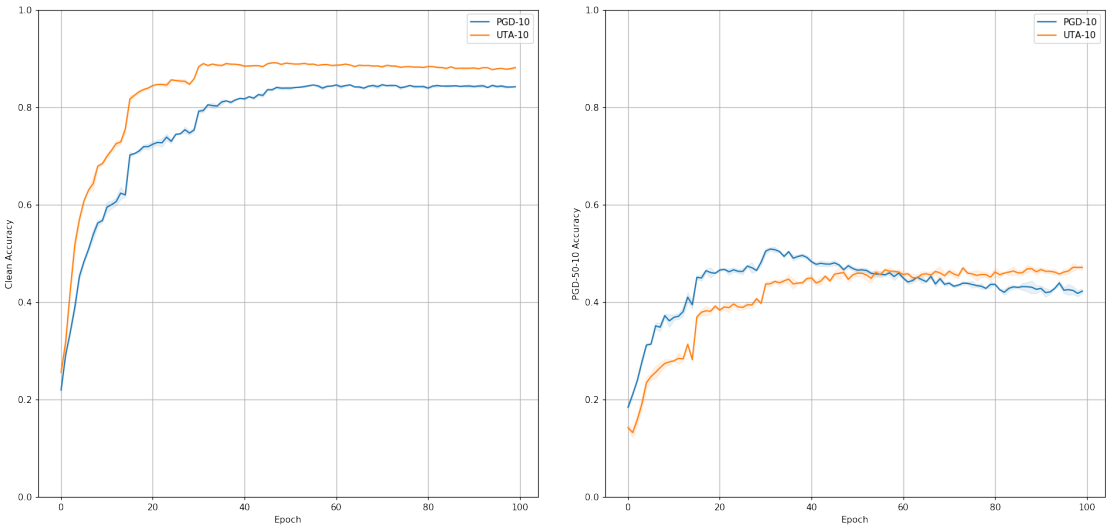
(a) Clean accuracy **(b)** Adversarial accuracy

Figure 5.5: Adversarial training with PGD-10 vs. UTA-10 using the standard DUE method on **CIFAR-10**. (a): comparison ofr the clean accuracy obtained by doing AT with the different attacks. (b): adversarial evaluation against PGD–50-10 of PGD vs. UTA-AT. Note how PGD in general seems to be slightly more robust than UTA, that on the other hand has an unstable training pattern with some sharp performance drops that last only fer iterations. It is argued here that this may be due to the effect of spectral normalization of adversarial training. Results are averaged over three runs.

5.2.2 Robustness of fast adversarial training

In order to give more credit to the hypothesis that spectral normalization may have a certain influence when doing adversarial training some experiments were done comparing two fast adversarial training methods: FGSM and UTA-1. The setting is the same as the one described in Section 5.2.1 but using FGSM with $\varepsilon = \frac{8}{255}$ and UTA-1 with the same ε and $\alpha = \varepsilon$. Also in this case are presented the results obtained *with* and *without* spectral normalization.

Regarding the former, results are shown in Figure 5.7. Quite interestingly it is possible to see that the robust accuracy obtained when doing fast AT with DUE shows an *oscillating behavior* for both methods: during training there are periods in which UTA-1-AT is more robust and periods in which FGSM-AT is better. Training is quite unstable and there are some *drops* both in clean and adversarial accuracy. Intuitively it is like the attacks are trying to make some modifications to train the model to be more robust, but spectral normalization is not allowing those modifications and forces the network to have a more *standard* behaviour.

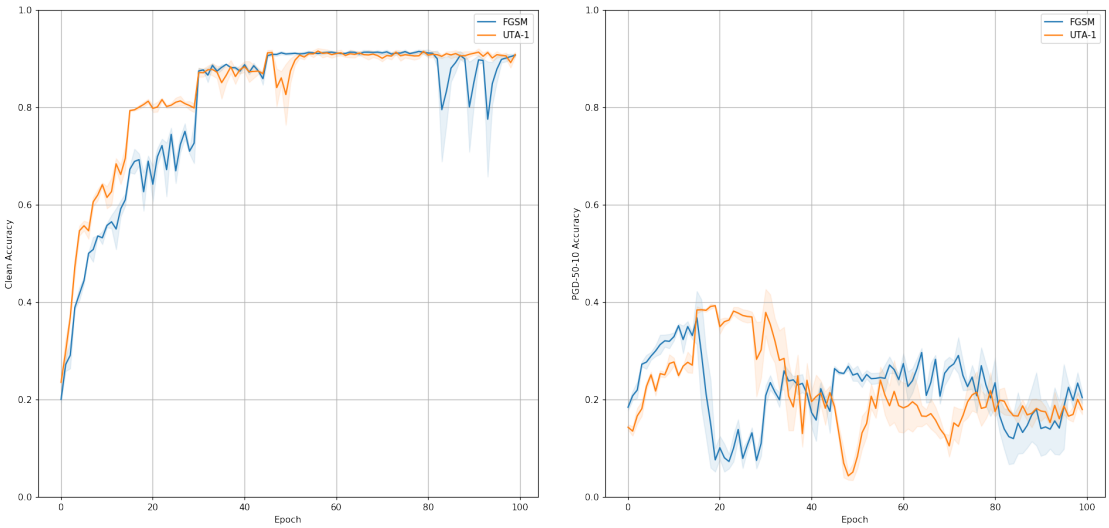


(a) Clean accuracy **(b)** Adversarial accuracy

Figure 5.6: Adversarial training with PGD-10 vs. UTA-10 using a DKL adaptation of the DUE method: **spectral normalization** is turned **off** on **CIFAR-10**. (a): comparison of the clean accuracy obtained by doing AT with the different attacks. (b): adversarial evaluation against PGD-50-10 of PGD vs. UTA-AT. Note that when spectral normalization is not present in the model the PGD adversarial accuracy reaches an higher adversarial accuracy compared to when spectral normalization is active during training, but then it experiences a downwards robustness trend indicating that the network is overfitting the attack. UTA-AT on the other hand seems to be more robust and stable when spectral normalization is turned off, compared to the case in which is active.

Exactly the same experiments are now repeated using the DKL method modification of DUE that simply consists in *not* doing spectral normalization in the feature extractor. Results are shown in Figure 5.8. In this case the oscillating adversarial accuracy pattern seen in the previous experiments is not detected, but both methods seem to have a more stable training. Regarding FGSM-AT it is possible to see a behavior similar to *catastrophic overfitting* at around epoch 15: sudden robust accuracy drop and the network can not recover during the training. For UTA-1-AT on the other hand the situation is different as there is a certain robustness drop, but it happens later and the network is able to recover from this during the training. This is a pattern very similar to the one seen in the catastrophic overfitting experiments presented in Section 4.2.3 with the dropout methods.

Also in this case it is possible to say here that also in the case of *fast* adversarial



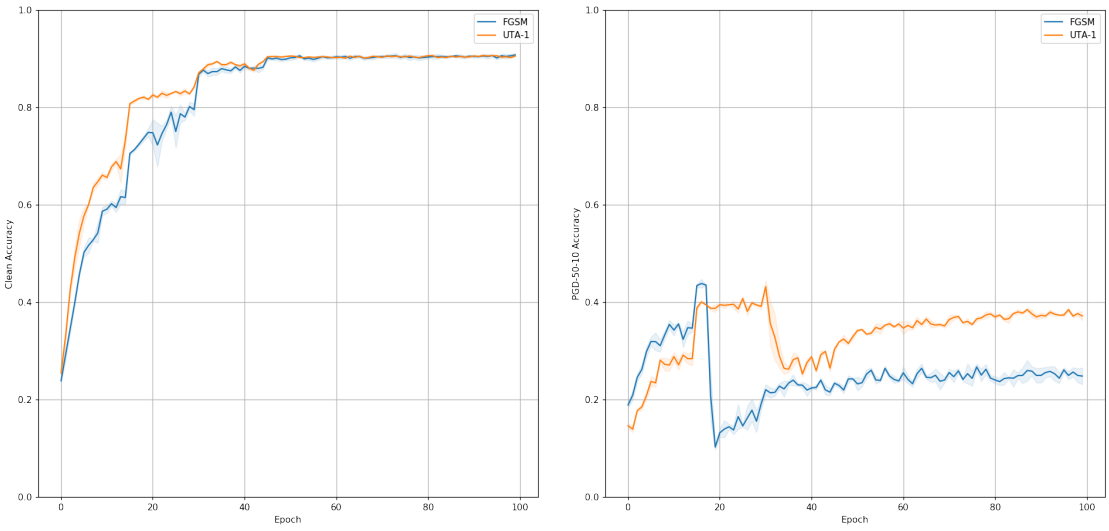
(a) Clean accuracy **(b)** Adversarial accuracy

Figure 5.7: Adversarial training with FGSM vs. UTA-1 using the standard DUE method on **CIFAR-110**. (a): comparison of the clean accuracy obtained by doing AT with the different attacks. (b): adversarial evaluation against PGD-50-10 of FGSM vs. UTA-AT. Note how in this case the adversarial accuracy of both FGSM and UTA-AT has a sort oscillating pattern and it is not possible to determine which method is better. Furthermore there are some sharp accuracy drops both in terms of clean ad adversarial accuracy for both methods during training.

training the intuition that spectral normalization may have an impact on the robustness of the model seems to be reasonable.

5.3 Discussion

In this chapter the DUE method has been studied from an *adversarial perspective*, comparing both standard loss based attacks and the uncertainty guided method proposed in this thesis. After a motivating toy example highlighting the effects of adversarial training to the DUE method in a 2-dimensional settings, some experiments where made using a deeper network and a more challenging dataset. In this last setting during the experiments it was *hypothesized* that adversarial training may be influenced by the *spectral normalization* applied in the DUE method studied. Further experiment gave credit to this intuition. Actually the connection between AT and spectral norm have been already studied in literature, for example in [57] it is shown how adversarial training with projected gradient descent based attacks is equivalent to a *data-dependent* operator norm regularization. In particular the



(a) Clean accuracy **(b)** Adversarial accuracy

Figure 5.8: Adversarial training with FGSM vs. UTA-1 using the modified DKL method when **spectral normalization** is turned **off** on **CIFAR-**10. (a): comparison of the clean accuracy obtained by doing AT with the different attacks. (b): adversarial evaluation against PGD-50-10 of FGSM vs. UTA. In this case there is no more the oscillating pattern seen in Figure 5.7. The FGSM-AT models seems to suffer from catastrophic overfitting at around epoch 15, while the UTA-1-AT models seems to be definitely more robust in this setting. There is a certain drop in robust accuracy but it happens later and is not as catastrophic. These results are pretty much in line with what seen with MC Dropout in Section 4.2.3.

authors confirms the intuition that the network spectral properties are linked to its adversarial robustness. The experiments carried out in this chapter with PGD-AT are in line with the results of the paper. Furthermore the UTA experiments highlighted how spectral normalization may *limit* the efficacy of uncertainty guided adversarial training. Actually these results emerged almost as a *side effect* during the experiments, but despite that they are actually very interesting and it is left here to future work to understand in a more *formal* way what are the connection betweens UTA and spectral norm and why this behavior happens.

Chapter 6

Conclusions and Next Steps

The main goal of this work was to introduce some techniques that allow an *uncertainty guided* exploration of the input / latent space of a model. Everything started by looking at some *connections* between the lines of work of adversarial training and uncertainty estimation, then, after having found those, the focus has been shifted on designing some methods that can provide a way to craft some adversarial attack by exploring a particular space in an *uncertainty based* way. The UTA attacks have been introduced and their various a*dvantages* have been highlighted in several different ways. To summarize, in this work it was shown how uncertainty based attacks compared to standard techniques lead to:

- **Higher clean accuracy**. This is because UTA methods do not alter too much the properties of the decision boundaries compared to standard clean training and lead to the learning of *better* latent spaces.
- **More generalization**. UTA based adversarial training seems to suffer less from overfitting, even in the fast variants, with respect to most state of the art loss based techniques.
- **Wider exploration of the latent space**. UTA attacks allow to train effectively using *larger* ε-balls, thus giving the opportunity to the network to find *better* perturbations during AT and to be *more robust* to different situations.

In some experiments, results suggested that the main disadvantage of UTA-AT is that it leads to *lower* levels of PGD robustness compared to loss based methods. This is actually not completely true, as in many situations, especially when training *fast* version of the attacks, UTA-AT methods lead to more robust models with respect to FGSM-AT. Moreover, it has been shown how the high-ε regime allows UTA to train models robust to *smaller* test-time perturbations, opening to a

different way of doing adversarial training in which a sort of robustness to the change of ε is guaranteed.

Furthermore, UTAs have been analyzed under several uncertainty estimation methods. These, being a key element in the attack definition, played a very important role in the performances of the AT techniques proposed here. After having evaluated UTA with Variational BNNs and with the MC Dropout approach a shift of focus was made in order to consider a new, but very promising, way of computing uncertainty: *deterministic methods*. Under these circumstances, an analysis of adversarial training using deep Gaussian Processes techniques was done and it was highlighted how *spectral normalization* can have an impact on the goal of robustness.

Several potential directions can be done as a follow-up of this project. For example, an interesting direction could be to study *why* spectral normalization impacts PGD and UTA adversarial training. Furthermore the study of UTA under an *online learning* perspective would be very interesting. Adversarial training can be seen as a form of online learning: given a sample it is possible to find *infinite* perturbations and the model will not know what perturbation will the adversary use. In this setting, an *uncertainty randomized* approach could be applied in order to provide some theoretical guarantees that can be used to build more robust and able to generalize models.

Appendix A

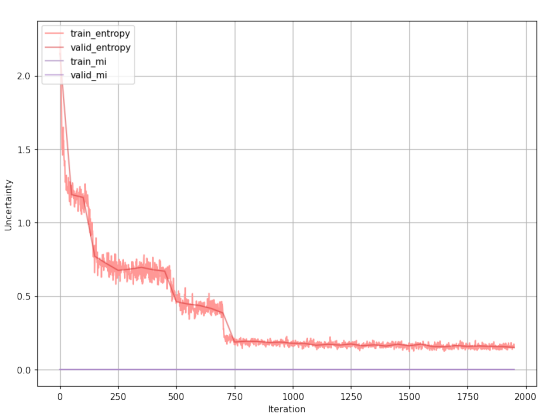
Additional results with Variational BNNs

In this appendix are shown some additional results related to the experiments presented in Chapter 3 using the Variational BNNs approach on MNIST.

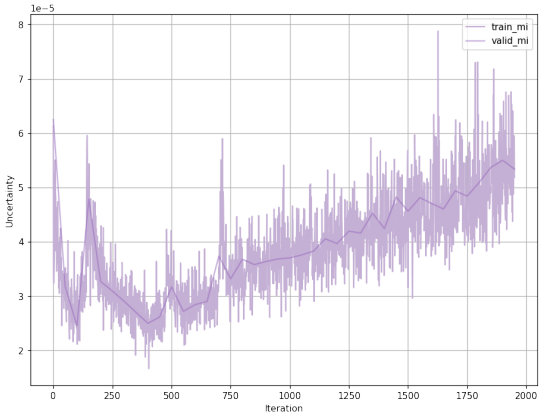
A.1 Entropy and Variational BNNs

In Figure A.1 are shown the results in terms of entropy and mutual information related to the clean training of a Variational BNN on MNIST. The conclusions made in Section 3.1 still hold here. In particular in subfigure A.1a it is possible to see the entropy trend that resembles a bit those of the aleatoric uncertainty shown in subfigure 3.2a even if entropy is a measure of epistemic uncertainty, in the sense that it quantifies the uncertainty present in the model. Regarding the mutual information, these is definitely lower than entropy in absolute value and has a patter similar to the one of epistemic uncertainty as estimated in Figure 3.2b. It is important to note that these measure of uncertainty differ in their implementation and so they are able to catch different component describing how uncertain the model is. This means that defining the correct measure of uncertainty is definitely not a trivial task and is dependent on the particular problem at hand.

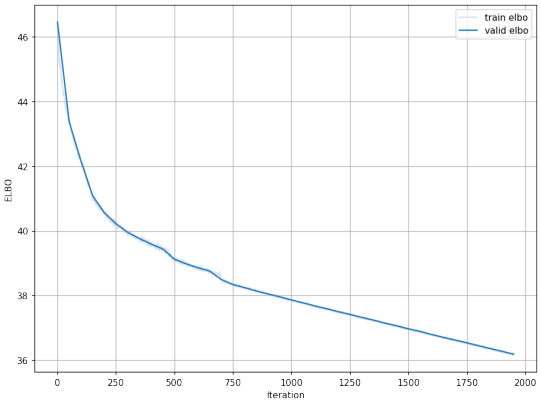
In Figure A.2 on the other hand are shown the results obtained by adversarially training with PGD. Also in this case, similarly to what observed in Figure 3.3, it is possible to see a flat behaviour both for uncertainty and accuracy for approximately the first 1500 iterations. For the rest of the training the pattern of mutual information shown in subfigure A.2b is very similar to the pattern of epistemic uncertainty seen in subfigure 3.3b showing that the this measure capture pretty much the same uncertainty as the epistemic uncertainty as defined in Equation 2.12.



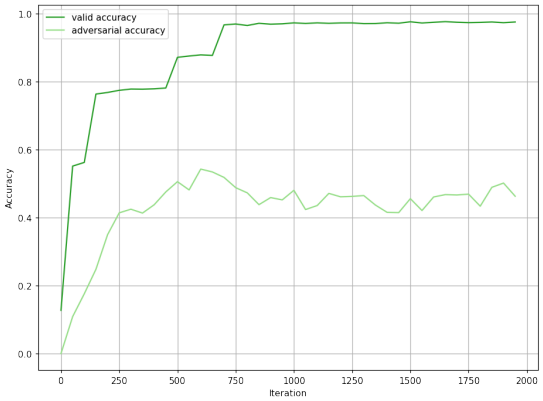
(a) train/validation uncertainty



(b) train/validation mutual information

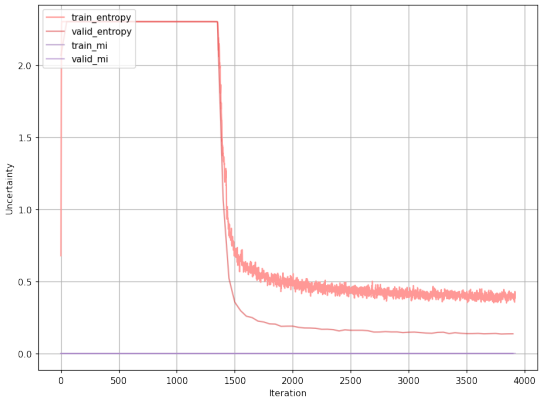


(c) train/validation ELBO

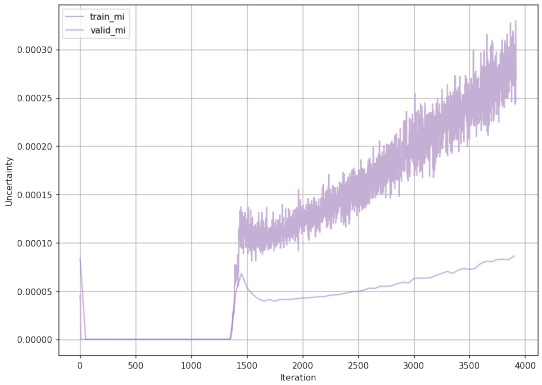


(d) clean (valid) and PGD-10 (adversarial) accuracy

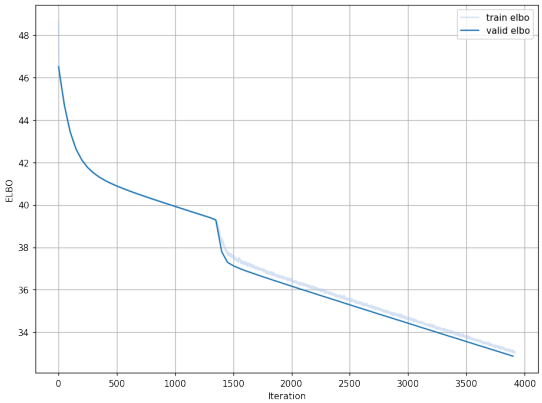
Figure A.1: Entropy, MI estimation and accuracy for a Variational BNN trained only with clean data on the MNIST dataset. (a) entropy and mutual information both on the training and validation set, note how entropy is definitely higher than mutual information. (b): zoom on mutual information uncertainty only during training and validation.(c): train and validation ELBO. (d): accuracies shown both with clean testing and adversarial testing using the PGD-10 attack. Note how the adversarial accuracy is definitely lower than the clean one.



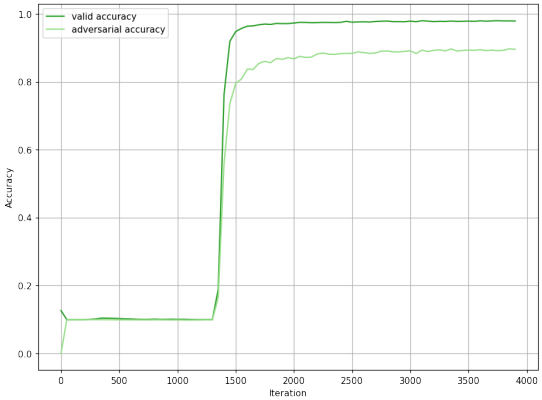
(a) train/validation uncertainty



(b) train/validation mutual information



(c) train/validation ELBO



(d) clean (valid) and PGD-10 (adversarial) accuracy

Figure A.2: Entropy, MI estimation and accuracy for a Variational BNN trained with PGD-10 data on the MNIST dataset. (a) entropy and mutual information both on the perturbed training and on the clea validation set. (b): zoom on mutual information only during training and validation.(c): train and validation ELBO. (d): accuracies shown both with clean testing and adversarial testing using the PGD-10 attack.

Appendix B

Architectures Used

In this section are described in detail the convolutional architectures used for the experiments with the various datasets. In Table B.1 is shown the LeNet5 architecture used for the MNIST and FashionMNIST MC Dropout experiments, while in Table B.2 the architecture used for the CIFAR-10 MC Dropout experiments.

LeNet
Input: $x \in \mathbb{R}^{28\times28}$
conv. (kernel: 5×5, $1 \rightarrow 6$; padding: 2; stride: 1)
ReLU
max pooling (kernel: 2×2; stride: 2)
convolutional (kernel: 5×5, $6 \rightarrow 16$; stride: 1)
ReLU
max pooling (kernel: 2×2; stride: 2)
Flattening
fully connected ($16 \times 5 \times 5 \rightarrow 120$)
ReLU
fully connected ($120 \rightarrow 84$)
ReLU
fully connected ($84 \rightarrow 10$)
$Softmax(\cdot)$

Table B.1: LeNet architecture used for experiments on MNIST. With $h\times w$ is denoted the kernel size. With $c_{in} \rightarrow y_{out}$ are denoted the number of channels of the input and output, for the convolution layers, and the number of input and output units for fully connected layers.

ResBlock (ℓ–th block)
Bypass:
conv. (ker: 1×1, pln/2 → pln; str: str; pad: 1), if str $\neq 1, \ell$ is odd
Batch Normalization, if str $\neq 1, \ell$ is odd
Feedforward:
conv. (ker: 3×3, pln/2 → pln, if str $\neq 1, \ell$ is odd, else pln → pln; str: str; pad: 1)
Batch Normalization
ReLU
MCD ($p = 0.2$)
conv. (ker: 3×3, $64 \times \ell \rightarrow 64 \times \ell$; str: 1; pad: 1)
Batch Normalization
Feedforward + Bypass
ReLU
MCD ($p = 0.2$)

ResNet Classifier
Input: $x \in \mathbb{R}^{3\times32\times32}$
conv. (ker: 3×3; $3 \rightarrow 64$; str: 1; pad: 1)
Batch Normalization
ReLU
MCD ($p = 0.2$)
2×ResBlock ($\ell \in [1, 2]$, str=1, pln=64)
2×ResBlock ($\ell \in [3, 4]$, str=2, pln=128)
2×ResBlock ($\ell \in [4, 5]$, str=2, pln=256)
2×ResBlock ($\ell \in [6, 7]$, str=2, pln=512)
AvgPool (ker:4×4)
Linear($512 \rightarrow 10$)

Table B.2: ResNet architectures for the experiments on CIFAR-10. Each ResNet block contains skip connection (bypass), and a sequence of convolutional layers, normalization, and the ReLU non–linearity. For clarity are listed the layers sequentially, however, note that the bypass layers operate in parallel with the layers denoted as "feedforward" [49]. The ResNet block for the model (right) differs if it is the first block in the network (following the input to the model).

Bibliography

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «Imagenet classification with deep convolutional neural networks». In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105 (cit. on p. 1).

[2] Jürgen Schmidhuber. «Deep learning in neural networks: An overview». In: *Neural networks* 61 (2015), pp. 85–117 (cit. on p. 1).

[3] Xiaolong Liu, Zhidong Deng, and Yuhan Yang. «Recent progress in semantic image segmentation». In: *Artificial Intelligence Review* 52.2 (2019), pp. 1089–1106 (cit. on p. 1).

[4] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. «Data clustering: a review». In: *ACM computing surveys (CSUR)* 31.3 (1999), pp. 264–323 (cit. on p. 1).

[5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: `1412.6572 [stat.ML]` (cit. on pp. 1, 5, 6, 8).

[6] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. *Intriguing properties of neural networks*. 2014. arXiv: `1312.6199 [cs.CV]` (cit. on pp. 1, 4, 5, 8).

[7] Alex Serban, Erik Poll, and Joost Visser. «Adversarial examples on object recognition: A comprehensive survey». In: *ACM Computing Surveys (CSUR)* 53.3 (2020), pp. 1–38 (cit. on p. 1).

[8] Eric Wong, Leslie Rice, and J. Zico Kolter. «Fast is better than free: Revisiting adversarial training». In: *International Conference on Learning Representations*. 2020. URL: `https://openreview.net/forum?id=BJx040EFvH` (cit. on pp. 1, 6, 8, 9, 39, 43).

[9] Yarin Gal. «Uncertainty in deep learning». In: (2016) (cit. on pp. 2, 12).

[10] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. «Weight uncertainty in neural network». In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1613–1622 (cit. on pp. 2, 11, 18).

[11] Laurent Valentin Jospin, Wray Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. «Hands-on Bayesian Neural Networks–a Tutorial for Deep Learning Users». In: *arXiv preprint arXiv:2007.06823* (2020) (cit. on pp. 2, 10).

[12] Yarin Gal and Zoubin Ghahramani. «Dropout as a bayesian approximation: Representing model uncertainty in deep learning». In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059 (cit. on pp. 2, 10, 11).

[13] Andrew Wilson and Hannes Nickisch. «Kernel interpolation for scalable structured Gaussian processes (KISS-GP)». In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1775–1784 (cit. on pp. 2, 13, 15).

[14] Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal. «On Feature Collapse and Deep Kernel Learning for Single Forward Pass Uncertainty». In: *arXiv preprint arXiv:2102.11409* (2021) (cit. on pp. 2, 15, 49, 50, 52, 54).

[15] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. «Examples are not enough, learn to criticize! Criticism for Interpretability». In: *Advances in Neural Information Processing Systems*. Vol. 29. 2016 (cit. on pp. 2, 9).

[16] Finale Doshi-Velez and Been Kim. «Towards A Rigorous Science of Interpretable Machine Learning». In: *arXiv:1702.08608* (2017) (cit. on pp. 2, 9).

[17] Yongchan Kwon, Joong-Ho Won, Beom Joon Kim, and Myunghee Cho Paik. «Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation». In: *Computational Statistics & Data Analysis* 142 (2020), p. 106816 (cit. on pp. 2, 12).

[18] Yann LeCun and Corinna Cortes. «MNIST handwritten digit database». In: (2010). URL: `http://yann.lecun.com/exdb/mnist/` (cit. on pp. 2, 17, 34).

[19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2019. arXiv: `1706.06083 [stat.ML]` (cit. on pp. 2, 6, 8).

[20] David Warde-Farley and Ian Goodfellow. «11 adversarial perturbations of deep neural networks». In: *Perturbations, Optimization, and Statistics* 311 (2016) (cit. on p. 5).

[21] Alex Krizhevsky. «Learning Multiple Layers of Features from Tiny Images». MA thesis. 2009 (cit. on pp. 6, 33, 34).

[22] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. «A unified gradient regularization family for adversarial examples». In: *2015 IEEE international conference on data mining*. IEEE. 2015, pp. 301–309 (cit. on p. 7).

[23] John M Danskin. «The theory of max-min, with applications». In: *SIAM Journal on Applied Mathematics* 14.4 (1966), pp. 641–664 (cit. on p. 7).

[24] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. «Robustness May Be at Odds with Accuracy». In: *arXiv preprint arXiv:1805.12152* (2019). eprint: 1805.12152 (cit. on p. 8).

[25] David Stutz, Matthias Hein, and Bernt Schiele. «Disentangling adversarial robustness and generalization». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 6976–6987 (cit. on p. 8).

[26] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. «Autoencoding beyond pixels using a learned similarity metric». In: *ICML*. 2016 (cit. on p. 8).

[27] Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. *Variational Approaches for Auto-Encoding Generative Adversarial Networks*. 2017. arXiv: 1706.04987 (cit. on p. 8).

[28] Maksym Andriushchenko and Nicolas Flammarion. «Understanding and Improving Fast Adversarial Training». In: *NeurIPS*. 2020 (cit. on pp. 9, 42).

[29] Peilin Kang and Seyed-Mohsen Moosavi-Dezfooli. *Understanding Catastrophic Overfitting in Adversarial Training*. 2021. arXiv: 2105.02942 [cs.LG] (cit. on pp. 9, 37).

[30] Diederik P Kingma and Max Welling. «Auto-encoding variational bayes». In: *arXiv preprint arXiv:1312.6114* (2013) (cit. on p. 11).

[31] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. «Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles». In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6405–6416 (cit. on p. 11).

[32] Andrey Malinin and Mark Gales. «Predictive uncertainty estimation via prior networks». In: *arXiv preprint arXiv:1802.10501* (2018) (cit. on p. 12).

[33] Janis Postels, Mattia Segu, Tao Sun, Luc Van Gool, Fisher Yu, and Federico Tombari. *On the Practicality of Deterministic Epistemic Uncertainty*. 2021. arXiv: 2107.00649 [cs.CV] (cit. on p. 13).

[34] Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip H. S. Torr, and Yarin Gal. *Deterministic Neural Networks with Inductive Biases Capture Epistemic and Aleatoric Uncertainty*. 2021. arXiv: 2102.11582 [cs.LG] (cit. on p. 13).

[35] Lynton Ardizzone, Radek Mackowiak, Carsten Rother, and Ullrich Köthe. «Training Normalizing Flows with the Information Bottleneck for Competitive Generative Classification». In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 7828–7840. URL: https://proceedings.neurips.cc/paper/2020/file/593906af0d138e69f49d251d3e7cbed0-Paper.pdf (cit. on p. 13).

[36] Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006 (cit. on p. 13).

[37] Joaquin Quinonero-Candela and Carl Edward Rasmussen. «A unifying view of sparse approximate Gaussian process regression». In: *The Journal of Machine Learning Research* 6 (2005), pp. 1939–1959 (cit. on p. 14).

[38] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. *Deep Kernel Learning*. 2015. arXiv: 1511.02222 [cs.LG] (cit. on pp. 15, 49, 54).

[39] Mihaela Rosca, Theophane Weber, Arthur Gretton, and Shakir Mohamed. «A case for new neural network smoothness constraints». In: (2020) (cit. on p. 15).

[40] Jeremiah Zhe Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. *Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness*. 2020 (cit. on p. 15).

[41] Piero Esposito. *BLiTZ - Bayesian Layers in Torch Zoo (a Bayesian Deep Learing library for Torch)*. https://github.com/piEsposito/blitz-bayesian-deep-learning/. 2020 (cit. on p. 18).

[42] Kunihiko Fukushima and Sei Miyake. «Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition». In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285 (cit. on p. 18).

[43] John S Bridle. «Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition». In: *Neurocomputing*. Springer, 1990, pp. 227–236 (cit. on p. 18).

[44] Diederik P Kingma and Jimmy Ba. «Adam: A method for stochastic optimization». In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 18).

[45] Abhimanyu Dubey, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. «Maximum-entropy fine-grained classification». In: *arXiv preprint arXiv:1809.05934* (2018) (cit. on p. 25).

[46] Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey Hinton. «Regularizing neural networks by penalizing confident output distributions». In: *arXiv preprint arXiv:1701.06548* (2017) (cit. on p. 25).

[47] Ekin D Cubuk, Barret Zoph, Samuel S Schoenholz, and Quoc V Le. «Intriguing properties of adversarial examples». In: *arXiv preprint arXiv:1711.02846* (2017) (cit. on p. 25).

[48] Gilberto Manunza, Matteo Pagliardini, Martin Jaggi, and Tatjana Chavdarova. «Improved Adversarial Robustness via Uncertainty Targeted Attacks». In: *ICML UDL* (2021) (cit. on pp. 29, 45).

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep Residual Learning for Image Recognition». In: *CVPR*. 2016 (cit. on pp. 33, 35, 65).

[50] Han Xiao, Kashif Rasul, and Roland Vollgraf. «Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms». In: *arXiv:1708.07747* (2017) (cit. on p. 34).

[51] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 35).

[52] R Venkatesh Babu. «Single-step adversarial training with dropout scheduling». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 950–959 (cit. on p. 36).

[53] Leslie N. Smith. «Cyclical Learning Rates for Training Neural Networks». In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017, pp. 464–472. DOI: 10.1109/WACV.2017.58 (cit. on pp. 38, 39).

[54] John Bradshaw, Alexander G de G Matthews, and Zoubin Ghahramani. «Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks». In: *arXiv preprint arXiv:1707.02476* (2017) (cit. on p. 50).

[55] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. «Invertible residual networks». In: *International Conference on Machine Learning*. PMLR. 2019, pp. 573–582 (cit. on p. 50).

[56] Sergey Zagoruyko and Nikos Komodakis. «Wide Residual Networks». In: *Proceedings of the British Machine Vision Conference (BMVC)*. Ed. by Edwin R. Hancock Richard C. Wilson and William A. P. Smith. BMVA Press, Sept. 2016, pp. 87.1–87.12. ISBN: 1-901725-59-6. DOI: 10.5244/C.30.87. URL: https://dx.doi.org/10.5244/C.30.87 (cit. on p. 52).

[57] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. «Adversarial training is a form of data-dependent operator norm regularization». In: *arXiv preprint arXiv:1906.01527* (2019) (cit. on p. 57).