IMPROVED GENERALIZATION-ROBUSTNESS TRADE-OFF VIA UNCERTAINTY TARGETED ATTACKS

Anonymous authors
Paper under double-blind review

ABSTRACT

The deep learning models' sensitivity to small input perturbations raises security concerns and limits their use for applications where reliability is critical. While adversarial training methods aim at training more robust models, these techniques often result in a lower unperturbed (clean) test accuracy, including the most widely used Projected Gradient Descent (PGD) method.

In this work, we propose *uncertainty-targeted attacks* (UTA), where the perturbations are obtained by maximizing the model's estimated uncertainty. We demonstrate on MNIST, Fashion-MNIST and CIFAR-10 that this approach does not drastically deteriorate the clean test accuracy relative to PGD whilst it is robust to PGD attacks. In particular, uncertainty-based attacks allow for using larger L_∞-balls around the training data points, are less prone to overfitting the attack, and yield improved generalization-robustness trade-off. Our source code is available at *anonymous.4open.science/r/Uncertainty-Targeted-Attacks-5BD3*.

1 INTRODUCTION

It has been shown that small perturbations added to the input can easily "fool" well-performing deep neural networks (DNNs) into making wrong predictions (Biggio et al., 2013; Szegedy et al., 2014), which limits their application in many real-world tasks due to security risks. The goal of *adversarial training* (AT) methods is improving the robustness to small perturbations of a trained classifier $\mathcal{C}_{\boldsymbol{\omega}} : \boldsymbol{x} \mapsto \hat{\boldsymbol{y}}$, with $\boldsymbol{x} \in \mathbb{R}^d$ denoting a data sample of finite dataset $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$ drawn from the training data distribution p_d, $\hat{\boldsymbol{y}} \in \mathbb{R}^c$ its prediction for c possible classes, and $\boldsymbol{\omega} \in \mathbb{R}^m$ the parameters of the model. For this purpose, AT methods make the assumption that all data points $\tilde{\boldsymbol{x}}$ within a small region around a training data point $\boldsymbol{x}_i, i \in [1, N]$ belong to the same class:

$$\|\tilde{\boldsymbol{x}} - \boldsymbol{x}_i\| \leq \varepsilon \Rightarrow \tilde{\boldsymbol{y}} = \boldsymbol{y}_i, \quad \text{and} \quad \nexists j \in [1, N], \text{s.t. } j \neq i, \boldsymbol{y}_j \neq \boldsymbol{y}_i, \|\tilde{\boldsymbol{x}} - \boldsymbol{x}_j\| \leq \varepsilon\,. \tag{AT-Asm}$$

As sampling *all* the points within the small regions around the training data points is infeasible, common AT methods *greedily* target this weakness of DNNs by training the model with *modified training samples* as follows:

$$\min_{\boldsymbol{\omega}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim p_d}\big[\max_{\boldsymbol{\delta}\in\Delta} \mathcal{L}(\mathcal{C}_{\boldsymbol{\omega}}(\boldsymbol{x}+\boldsymbol{\delta}), \boldsymbol{y})\big]\,, \tag{AT}$$

where $\mathcal{L}$ denotes the loss function, and the added *worst-case* perturbation $\boldsymbol{\delta} \in \mathbb{R}^d$ is constrained to a small region around the sample $\boldsymbol{x}$, typically a ball $\Delta \triangleq [-\varepsilon, \varepsilon]^d$ with $\varepsilon > 0$. Popular implementations of the (AT) objective are the *projected gradient descent* (PGD, Madry et al., 2018)—where the inner maximization step is implemented with k gradient ascent steps, and its "fast variants"—which take only one step to compute the perturbation $\boldsymbol{\delta}$, e.g., *fast gradient sign method* (FGSM, Goodfellow et al., 2015), see § 3. However, further empirical studies showed that AT methods often reduce the average accuracy on "clean" unperturbed test samples, indicating the two objectives—robustness and clean accuracy—might be competing (Tsipras et al., 2019; Su et al., 2018). Moreover, Wong et al. (2020) further pointed out a phenomenon referred to as *catastrophic overfitting* where the robustness of fast AT methods rapidly drops to almost zero, within a *single* training epoch, see § 5.

A separate line of work aims at increasing the interpretability of the model by associating to each of its predictions an *uncertainty estimate* (Kim et al., 2016; Doshi-Velez & Kim, 2017). Two main uncertainty types in machine learning are considered: (i) *aleatoric*–describing the *noise* inherent in the observations, as well as (ii) *epistemic*–uncertainty originating *from the model*. While the

former *cannot* be reduced, the latter arises due to insufficient data to train the model, and it *can* be explained away given enough data. In the context of classification, apart from capturing high-uncertainty due to overlapping regions of different classes, the epistemic uncertainty also captures which regions of the data space are not "visited" by the training samples.

In this work we consider "uncertainty targeted attacks" (UTA), motivated by the insights that (i) as standard AT methods find a perturbation $\boldsymbol{\delta}$ which maximizies the loss, the perturbed sample $\tilde{\boldsymbol{x}} \triangleq \boldsymbol{x} + \boldsymbol{\delta}$ is moved toward the decision boundary; and (ii) an uncertainty maximizing perturbation would move $\tilde{\boldsymbol{x}}$ either towards the decision boundary or toward non-visitied regions in data space, depending on the proximity. Furthermore, in this paper we investigate if finding a perturbation $\boldsymbol{\delta}$ which *maximizes the model's estimated uncertainty* can provide a better trade-off between generalization and robustness, as well as improve the reported problem of catastrophic overfitting.

Contributions. Our contributions can be summarized as follows:

- Primarily, we point out that standard loss-based attacks, and in particular PGD requires careful tuning of the size of the ε-ball, as large ε values *can* break the AT-Asm assumption, and in turn cause oscillations throughout training, see § 4.1.
- We propose *uncertainty-targeted attacks* (UTA), where the perturbations are obtained by maximizing the model's estimated uncertainty. Moreover, when approximating the uncertainty using a single model—corresponding in computational cost to that of PGD—our approach reduces to maximizing the entropy of the classifier, see § 4.2. We demonstrate on our 2D illustrative example that such uncertainty-based perturbed samples have the advantage of: *(i)* reduced "crossing" of the decision boundary—see § 4.3, allowing for using larger ε-balls, and in turn, *(ii)* reduced oscillations throughout training.
- Finally, our extensive empirical evaluation on Fashion-MNIST, MNIST, SVHN and CIFAR-10 shows that this approach—both entropy and uncertainty maximization, when implemented *either in image or latent space*: (i) do not drastically decrease the clean test accuracy relative to PGD, while at the same time (ii) it is robust to PGD attacks, and (iii) its fast variant does not suffer from catastrophic overfitting, see § 5.

2 Related work

Adversarial training. In the context of computer vision, small perturbations in image space that are not visible to the human eye can fool a well-performing classifier into making wrong predictions. This observation motivated an active line of research on adversarial training (see Biggio & Roli, 2018, and references therein)—namely, training with such adversarial samples to obtain robust classifiers. However, further empirical studies showed that such training reduces the training accuracy, indicating the two objectives—robustness and generalization—are competing (Tsipras et al., 2019; Su et al., 2018). Zhang et al. (2019) characterize this trade-off by decomposing the robust error as the sum of the natural error and the boundary error. They further introduce TRADES, which adds a regularizer encouraging smoothness in the neighborhood of samples from the data distribution. Although unsupervised as ours, TRADES is orthogonal to our work, and it would be interesting to explore combining it with the herein proposed uncertainty-based AT. Most similar to ours, Zhang et al. (2020) argue that "friendly" attacks–attacks which limit the distance from the boundary when crossing it through early stopping–can yield competitive robustness. Their approach differs from ours as they do not consider the notion of uncertainty which also more elegantly handles the "manual" stopping.

Latent-space attacks. Stutz et al. (2019) postulate that the observed drop in clean test accuracy appears because the adversarial perturbations leave the data-manifold, and that 'on-manifold adversarial attacks' will hurt less the clean test accuracy. The authors thus propose to use perturbations in the latent space of a VAE-GAN (Larsen et al., 2016; Rosca et al., 2017), and recently Yuksel et al. (2021) used Normalizing Flows for this purpose due to their exactly reversible encoder-decoder structure. Our approach is orthogonal to these works as it can also be applied in latent space—see § 5.2, and interestingly, we show that our method does not decrease notably the clean test accuracy relative to common loss-based AT, while it notably improves the model's robustness relative to standard training.

Maximizing entropy. As one way to estimate the model's uncertainty is using entropy—see § 3, *maximum entropy* (Pereyra et al., 2017) approaches can be seen as relevant to ours. In particular, in the context of standard classification, Pereyra et al. (2017) penalize the confident predictions by adding a regularizer that maximizes the entropy of the output distribution. Moreover, in the context of adversarial training, the results of (Cubuk et al., 2017, §3.2) indicate that adding such a regularizer improves the model robustness to PGD attacks.

Uncertainty estimation. While standard DNN training performs a maximum likelihood estimation of the parameters $\boldsymbol{\omega} \in \Omega$, training Bayesian Neural Networks (BNNs) extends to estimating the posterior distribution, providing a mathematically grounded framework for uncertainty. However, due to the integration with respect to the whole parameter space Ω, BNNs come at a prohibitive computational cost that is often intractable for DNNs. A popular epistemic uncertainty estimation method is *deep ensembles* (Lakshminarayanan et al., 2017), which trains a large number of models on the dataset and combines their predictions to estimate a predictive distribution over the weights. Gal & Ghahramani (2016) further show that *Dropout* (Srivastava et al., 2014) when applied to a neural network approximates Bayesian inference of a Gaussian processes (Rasmussen & Williams, 2005). The proposed *Monte Carlo Dropout* (MC Dropout)—which applies Dropout at inference time—allows for computationally efficient uncertainty estimation.

3 PRELIMINARIES

Adversarial training. The inner maximization problem of Eq. AT can be implemented in several ways. As in general the optimization is non-convex, Lyu & Liang (2015) propose approximating the inner maximization problem with Taylor expansion and then applying a Lagrangian multiplier. For ℓ_∞-bounded attacks, this linearization yields the FGSM (Goodfellow et al., 2015), with its perturbation defined as:

$$\boldsymbol{\delta}_{\text{FGSM}} \triangleq \varepsilon \cdot \text{Sign}\big(\nabla_{\boldsymbol{x}} \mathcal{L}(\mathcal{C}_{\boldsymbol{\omega}}(\boldsymbol{x}), \boldsymbol{y}) \big) \,, \tag{FGSM}$$

where $\text{Sign}(\cdot)$ denotes the sign function. To improve the catastrophic overfitting of FGSM, Tramèr et al. (2018) propose adding a random vector $\boldsymbol{\xi}$ to FGSM as follows:

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

where $\boldsymbol{\xi} \sim U([-\varepsilon, \varepsilon]^d)$, $\alpha \in [0, 1]$ is selected step size, and Π is projection on the ℓ_∞–ball. The PGD method (Madry et al., 2018) applies FGSM for $i = 1, \ldots, k$ steps (with $\boldsymbol{\delta}^0_{\text{PGD}} \triangleq \mathbf{0}$):

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

PGD with k steps is often referred to as PGD-k.

Uncertainty estimation. Assuming an ensemble of M models $\{\mathcal{C}^{(m)}_{\boldsymbol{\omega}}\}_{m=1}^M$ (e.g., trained independently or sampled with MC Dropout, see § 2) where each model outputs non-scaled values–"logits", we define $\hat{\boldsymbol{y}} \in \mathbb{R}^C$ as the average prediction:

$$\hat{\boldsymbol{y}} = \frac{1}{M} \sum_{m=1}^{M} \text{Softmax}\big(\mathcal{C}^{(m)}_{\boldsymbol{\omega}}(\boldsymbol{x}) \big) \,.$$

Given $\hat{\boldsymbol{y}}$, there are several ways to estimate the model's uncertainty, see (Gal, 2016, §3.3). We use the *entropy* of the output distribution (over the classes) to quantify the uncertainty estimate of a given sample $\boldsymbol{x}$:

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{\omega}) = - \sum_{c \in C} \hat{\boldsymbol{y}}_c \log \hat{\boldsymbol{y}}_c \,. \tag{E}$$

If approximating (E) using a *single* model $M = 1$, uncertainty estimation with entropy is equivalent to the entropy of the model's output distribution (and as we shall see in 4.2, uncertainty-based AT can be seen as *maximum entropy* attacks).

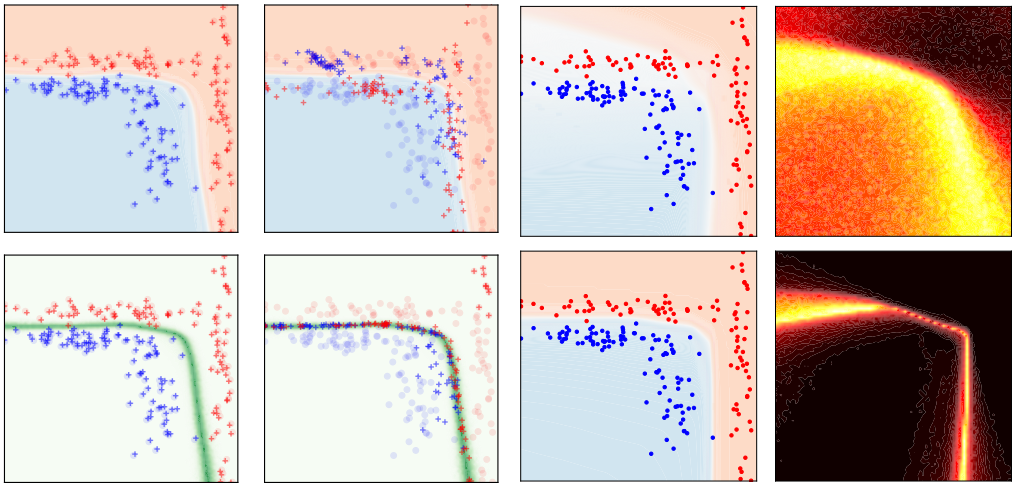
4 UNCERTAINTY TARGETED ATTACKS (UTA)

4.1 MOTIVATING EXAMPLE

In the context of computer vision, adversarial training aims at finding "imperceptible" perturbation which does not change the label of the input data-point–see AT-Asm. Thus the selected ε value for training–herein denoted with ε_{train}–is typically small. In this section, we argue that satisfying AT-Asm "globally" can be in some cases very restrictive. As an example, Fig. 1 depicts a toy experiment with non-isotropic distances between samples of opposite classes in $\mathbb{R}^2$. In particular, there are two regions with notably different distances between samples from the opposite class, let us denote the respective optimal ε values for each region with ε_1 and ε_2, and from Fig. 1, $\varepsilon_1 << \varepsilon_2$. In this section, we focus on PGD—top row, and below we discuss two *potential* issues of loss-based attacks.

AT-Asm enforces that we select $\varepsilon_{train} = \varepsilon_1$. In cases when $\varepsilon_1 << \varepsilon_2$ adversarial training only marginally improves the model's robustness in the second region where ε_2 can be used locally during training as such perturbations will not modify the class label of samples of that region. Moreover, if the data is disproportionately concentrated—e.g. small mass of the ground truth data distribution is concentrated in the first region, and the remaining is concentrated in the second region—one still needs to select $\varepsilon_{train} = \varepsilon_1$, as per AT-Asm.

To later contrast the behavior of loss-based PGD with uncertainty-based PGD, in Fig. 1a-1b we consider loss-based PGD with ε_{train} which *violates* AT-Asm (thus this value would not be used in practice by practitioners). From Fig. 1 we observe that PGD can perturb the sample by moving it on the opposite side of the boundary—resulting in *mislabeled* samples. See Fig. 8 for such analysis on CIFAR-10. While Fig. 1a-b illustrates the difference between attacks for a *fixed model*, Fig. 1c illustrates the decision boundaries *after* adversarial training with PGD. Similarly, Fig. 1d shows that for this "violating" case PGD training is characterized with large oscillations of the decision boundary, resulting in inefficient training. See App. A for results on the same dataset but using Gaussian Process as a model.



(a) 1 step (b) 20 steps (c) Decision boundary (d) Boundary oscillations

Figure 1: 2D experiment—performance of loss-based **PGD** (top) Vs. **UTA**-based PGD (bottom) using ε which *violates* the AT-Asm. **Columns a-b.** Attacks to *a fixed trained classifier*, with 1 and 20 steps (columns), resp. $\circ$ and $+$ depict the clean and perturbed samples, resp., and their color blue/red depicts their class. *Top row:* background depicts the assigned probability to each class of the attacked model, including its decision boundary. *Bottom row:* background depicts the uncertainty estimates of a 10 model ensemble, where darker green is higher. **Column c.** Decision boundaries *after* adversarial training with *large* ε (and $k = 15$, $\alpha = 0.05$). **Column d.** Summary of the boundary oscillations over the updates, for a fixed mini-batch size for both PGD and UTA. For each point $\boldsymbol{x} \in \mathbb{R}^2$ we count how many times the classifier changed its prediction, and depict with darker to lighter color zero to many changes, resp. See § 4.1 and § 4.3 for discussion.

4.2 UNCERTAINTY & MAXIMUM-ENTROPY ADVERSARIAL TRAINING

Unlike standard loss-based attacks, we propose uncertainty-guided exploration. Uncertainty Targeted Attacks (UTA) aim at finding perturbations which *maximize the model's uncertainty estimate* for the samples of the training dataset, and can be applied in either the data or the latent space:

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

where the encoder $\mathcal{E}$ is the identity when perturbations are applied in the input space. The above formulation also applies for latent space UTA perturbations, where with abuse of notation, the model can be seen as having an encoder part $\mathcal{E}:\boldsymbol{x}\mapsto\boldsymbol{z}$ and a classification part $\mathcal{C}_{\boldsymbol{\omega}}:\boldsymbol{z}\mapsto\boldsymbol{y}$, and in that case $\boldsymbol{\delta}\in\mathbb{R}^l$, where l is the dimension of the latent space. When approximating UTA using a *single* model $M=1$, UTA can be seen as *maximum entropy* attacks.

Similar to loss-based AT, the inner maximization of UTA—for both image and latent-space perturbations—can be implemented analogously to PGD and FGSM:

$$\boldsymbol{\delta}_{\mathrm{u}}^{i} \triangleq \prod_{\|\cdot\|_\infty\leq\varepsilon}\Big(\boldsymbol{\delta}_{\mathrm{UTA}}^{i-1}+\alpha\cdot\mathrm{Sign}\big(\nabla_{\boldsymbol{x}}\,\mathcal{H}(\mathcal{E}(\boldsymbol{x})+\boldsymbol{\delta}_{\mathrm{UTA}}^{i-1},\boldsymbol{\omega})\big)\Big)\,, \tag{UTA-PGD}$$

see Alg. 1 for details where for simplicity we use a single model $M=1$. For brevity, we often refer combining UTA-PGD with PGD and FGSM below as *UTA* and *UTA with one step*, resp.

Algorithm 1 Pseudocode of Uncertainty Targeted Attacks in input space using a single model.

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 $t\in 0,\ldots,T-1$ **do**
  Sample $\boldsymbol{x},\boldsymbol{y}\sim p_d$
  $\boldsymbol{\delta}_u^0\leftarrow\mathbf{0}$
  for $i\in 0,\ldots,K-1$ **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 $\left\|\boldsymbol{\delta}_u^{i+1}\right\|_\infty\leq\varepsilon$)*
  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: $\boldsymbol{\omega}_T$

4.3 ADVANTAGES OF UTA

In this section, we primarily argue that having ε values that differ in different regions of the input data space leads to improved generalization while assuming infinite model capacity.

In the general case, the largest ε value which does not break the AT-Asm assumption *locally* would *differ* among m non-overlapping local regions of the input data space, and let us denote these with $\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_m$. Note that, as such none of the $\varepsilon_i, i\in 1,\ldots,m$ when applied to the corresponding *i-th* region changes the label class. Let without loss of generality $\varepsilon_1<\varepsilon_2<\cdots<\varepsilon_m$. AT-Asm forces that for training we select $\varepsilon_{train}=min(\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_m)=\varepsilon_1$. The decision boundary of a model f trained adversarially with ε_{train} lies within a margin that is far by *at least* $\frac{\varepsilon_{train}}{2}$ from any training data point. Using that maximizing the margin of the decision boundary improves the generalization of the model (Vapnik, 1995), implies that using the corresponding ε_i for the i-th region will lead to improved generalization relative to when choosing single "global" ε_{train} value. Thus, having "adaptive" ε-value which does not break the AT assumption *locally* improves the generalization of the model, and in turn, its robustness.

Uncertainty-based AT achieves the above desired goal in an elegant way: as UTA does not cross the decision boundary, it effectively uses varying $\varepsilon_i \leq \varepsilon_{train}$, for different local regions of the input data space, where ε_{train} is selected beforehand. From Fig. 1, we observe that UTA-based training or more precisely UTA-PGD is relatively less sensitive to the choice of ε_{train}. Importantly, inline with the above discussion, using large ε for UTA-PGD during training does not deteriorate its final clean accuracy, as Fig. 1c depicts. Finally, from Fig. 1d we observe that even for large selected values of ε_{train}–which value breaks AT-Asm solely *globally*, the boundary decision of UTA training oscillates relatively less to that of PGD and mostly close to the ground-truth decision boundary. By being able to use larger values for ε, UTA allows for larger "exploration" of the input space and in turn learn "better" latent representations and decision boundary, characterized by better clean-accuracy and PGD robustness trade-off, as we shall see in § 5. This is in sharp contrast to PGD, which is forced to use the smallest ε_1 even in cases when $\varepsilon_1 << \varepsilon_m$, as using ε_{train} value which violates AT-Asm performs poorly. Moreover, we postulate that such non-isotropic margins as in Fig. 1 are more likely to occur in real-world datasets.

Importantly, when training with a larger ε, we shall see in § 5 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.

Additional advantages. Relative to standard AT methods, UTA is *unsupervised*, as it does not use the ground truth labels y to compute the perturbations (note that in Eq. UTA the labels are used solely to update the model, which is taken as supervised for simplicity). Thus it can be extended to unsupervised methods which output probability estimates. Also, it includes more general perturbations relative to loss-based AT as, depending on the proximity of the data point at hand, will either move it toward the decision boundary, or toward "unexplored" regions of the training set.

5 Experiments

Since in this work we propose a more abstract framework of using uncertainty-based adversarial training, rather than loss-based, many of the existing AT methods can be combined with this framework, e.g. PGD and UTA-PGD. Thus, the empirical evaluation does *not* aim to provide a new state-of-art result, but rather verify that for a given fixed setup, uncertainty-based adversarial training achieves improved generalization-robustness trade-off.

We conduct three main types of experiments, (i) image-space perturbations—see § 5.1, (ii) latent-space perturbations—see § 5.2, as well as (iii) experiments evaluating the robustness to catastrophic overfitting, see § 5.3.

Datasets. We evaluate on MNIST (Lecun & Cortes, 1998), Fashion-MNIST (Xiao et al., 2017), SVHN (Netzer et al., 2011) and CIFAR-10 (Krizhevsky, 2009). Throughout our experiments, we use three different architectures: multilayer perceptron (MLP) for MNIST, LeNet (Lecun et al., 1998) for Fashion-MNIST and SVHN, with increased capacity for the latter, as well as ResNet-18 (He et al., 2016) for CIFAR-10. See App. B for details on the implementation.

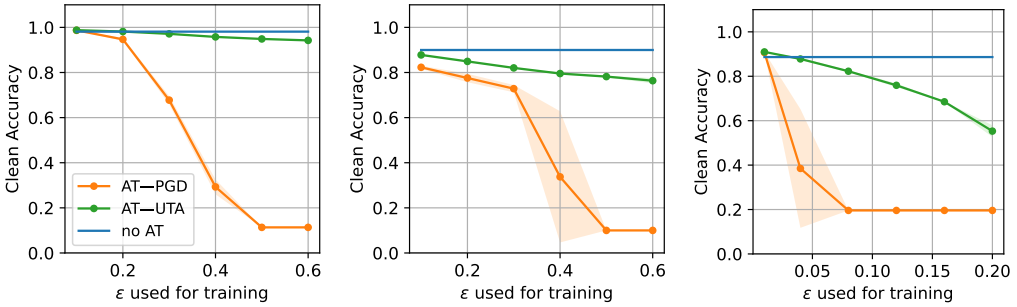
5.1 Input-space experiments

Methods. We compare: (i) **UTA**: for which we only use a single model for fair comparison, see App. C for additional results with an ensemble, and (ii) **PGD** (Madry et al., 2018).

	Fashion-MNIST				MNIST			
$\varepsilon_{\text{test}}$	.05	.1	.2	.3	.05	.1	.2	.3
PGD-AT								
$\varepsilon_{\text{train}} = .1$	79.6	75.1	44.6	41.2	97.3	93.6	61.7	49.4
$\varepsilon_{\text{train}} = .2$	75.3	73.1	66.9	40.4	92.7	89.4	75.8	50.1
$\varepsilon_{\text{train}} = .3$	70.7	68.7	63.8	51.5	64.4	60.8	53.0	41.6
$\varepsilon_{\text{train}} = .4$	32.7	31.7	29.5	26.5	27.4	26.4	25.1	24.0
UTA-AT								
$\varepsilon_{\text{train}} = .1$	82.3	71.9	44.1	43.9	97.2	92.5	55.4	49.5
$\varepsilon_{\text{train}} = .2$	81.3	77.1	62.9	42.5	96.9	94.3	75.9	50.3
$\varepsilon_{\text{train}} = .3$	78.9	75.3	67.9	47.1	95.6	93.1	79.8	52.0
$\varepsilon_{\text{train}} = .4$	76.2	72.72	65.6	56.3	93.8	90.6	76.8	53.0

	SVHN			
$\varepsilon_{\text{test}}$	.005	.01	.02	.04
PGD-AT				
$\varepsilon_{\text{train}} = .01$	86.6	80.5	65.7	49.1
$\varepsilon_{\text{train}} = .04$	37.6	36.5	34.4	29.9
$\varepsilon_{\text{train}} = .08$	19.6	19.6	19.6	19.6
$\varepsilon_{\text{train}} = .12$	19.6	19.6	19.6	19.6
$\varepsilon_{\text{train}} = .16$	19.6	19.6	19.6	19.6
UTA-AT				
$\varepsilon_{\text{train}} = .01$	86.0	78.3	61.1	47.3
$\varepsilon_{\text{train}} = .04$	84.8	81.0	71.8	55.7
$\varepsilon_{\text{train}} = .08$	79.3	75.8	68.7	55.6
$\varepsilon_{\text{train}} = .12$	72.9	69.8	63.7	52.7
$\varepsilon_{\text{train}} = .16$	65.7	63.0	57.8	48.3

Table 1: *Robustness-generalization summary*: comparison between PGD and *single-model* UTA on **Fashion-MNIST**, **MNIST** and **SVHN**. For a given $\varepsilon_{\text{train}}$ and $\varepsilon_{\text{test}}$, the listed accuracies are the average between the clean and robust accuracy, the latter given by AutoAttack (Croce & Hein, 2020), results are averaged over multiple runs. The best score for each $\varepsilon_{\text{test}}$ is highlighted in yellow.



(a) MNIST (b) Fashion-MNIST (c) SVHN

Figure 2: *Clean accuracy* comparison between PGD and *single-model* UTA on **MNIST**, **Fashion-MNIST** and **SVHN**, for varying ε used for training (x-axis). The results are averaged over multiple runs.

MNIST and Fashion-MNIST setups. Both for UTA and PGD we train several models for $\varepsilon_{\text{train}} \in \{0.1, 0.2, 0.3, 0.4\}$. We use the standard AutoAttack pipeline (Croce & Hein, 2020) to evaluate the robustness of each model to various attacks and for various $\varepsilon_{\text{test}} \in \{0.05, 0.1, 0.2, 0.3\}$. During training, the step size α is set to 0.01. The number of steps is always set to $K = \frac{\varepsilon_{\text{train}}}{0.01}$.

SVHN setup. For both UTA and PGD, we train several models for $\varepsilon_{\text{train}} \in \{0.01, 0.04, 0.08, 0.12, 0.16\}$. We use the standard AutoAttack pipeline to evaluate the robustness of each model for various $\varepsilon_{\text{test}} \in \{0.005, 0.01, 0.02, 0.04\}$. During training, for both methods, we use an α triangular scheduler. The initial and last step size α used for both methods is set to 0.005, for steps between 0 and $K/2$, α is increased linearly from 0.005 to $\max(0.005, \varepsilon_{\text{test}}/5)$, for steps between $K/2$ and K, α is decreased linearly to 0.005.

Clean accuracy. In Fig. 2, we compare the clean test accuracy of PGD vs. UTA-PGD trained models on Fashion-MNIST, SVHN and MNIST, resp. In all of our input-space experiments, we observe the clean accuracy of UTA is *consistently better* than the clean accuracy of PGD trained models, for all $\varepsilon_{\text{train}}$. We argue that this observation is in line with our hypothesis that UTA-perturbed samples do not cross the decision boundary often or largely, resulting in better clean test accuracy. These results on real data confirm the relevance of the toy experiment in § 4.1

Robustness-generalization trade-off. In Table 1, we compare the trade-offs between clean accuracy and robustness, for models trained with PGD and UTA, using different $\varepsilon_{\text{train}}$, and evaluated

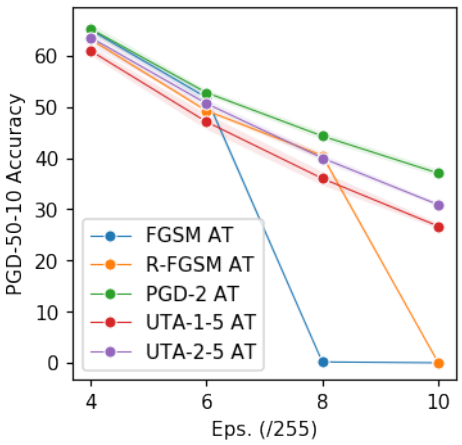


Figure 3: *Catastrophic overfitting* (CO) on **CIFAR-10** using ResNet 18, averaged over 3 runs. PGD robustness (50 iterations, 10 restarts) evaluated for multiple $\varepsilon_{\text{test}}$ foor FGSM, Random-FGSM, PGD with two steps, UTA with 1 step and an ensemble of 5 models, and UTA with 2 steps and and ensemble of 5 models.

$\varepsilon_{\text{test}}$	.05	.1	.2	.3
LS-PGD-AT				
$\varepsilon_{\text{train}} = .1$	27.9	22.7	18.9	18.9
$\varepsilon_{\text{train}} = .2$	11.3	11.3	11.3	11.3
$\varepsilon_{\text{train}} = .3$	11.3	11.3	11.3	11.3
$\varepsilon_{\text{train}} = .4$	11.3	11.3	11.3	11.3
LS-UTA-AT				
$\varepsilon_{\text{train}} = .1$	94.1	64.6	49.6	49.6
$\varepsilon_{\text{train}} = .2$	93.5	66.0	49.6	49.6
$\varepsilon_{\text{train}} = .3$	93.8	65.6	49.6	49.6
$\varepsilon_{\text{train}} = .4$	93.2	67.0	49.5	49.5

Table 2: *Latent-space robustness/ generalization trade-offs* reached when training with perturbations in latent space and testing with input-space perturbations, on **MNIST**. Methods compared are UTA-AT in latent space (LS-UTA-AT), and PGD-AT in latent space (LS-PGD-AT). The scores given are the average between clean accuracy and robust accuracy obtained using AutoAttack. Results are averaged over multiple runs, the best score for each $\varepsilon_{\text{test}}$ is highlighted in yellow..

on several $\varepsilon_{\text{test}}$. The robust accuracy is obtained using the standard pipeline of AutoAttack, which considers four different attacks: APGD_{CE}, $\text{APGD}^{\text{T}}_{\text{DLR}}$, FAB^{T}, and Square Attacks. The trade-off is expressed in terms of the average between the clean and robust accuracies. We observe that with UTA it is possible to achieve better trade-offs between robustness and clean accuracy relative to PGD. For the three datasets considered, for almost all $\varepsilon_{\text{test}}$ considered, UTA-AT gives a better average between clean and robust accuracy, as highlight in yellow.

5.2 Latent space experiments

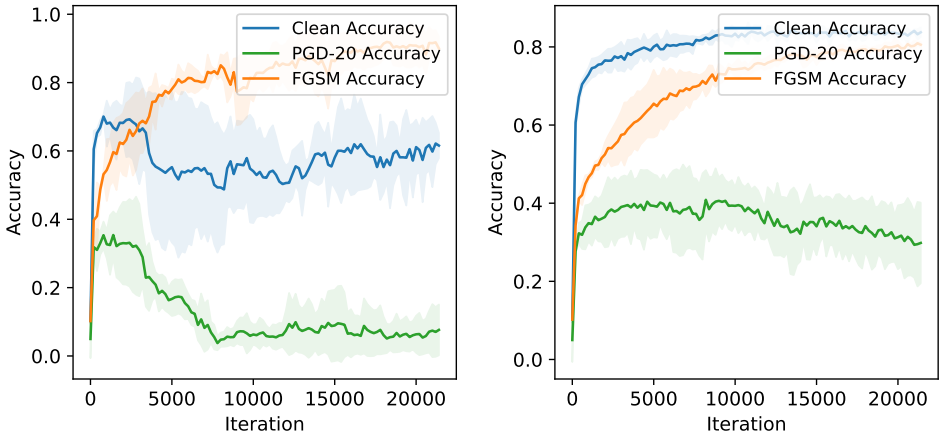
Methods & Setup. We compare: (i) **UTA** with ensemble of five models, and (ii) **PGD** (Madry et al., 2018). As encoder $\mathcal{E}(\boldsymbol{x})$ we use the convolutional and the first fully connected layers of LeNet, and the classifier consists of the last two fully connected layers of LeNet, see App. B for details. The encoder maps everything into a latent space of dimension 120. Perturbations are computed in this latent space. In our experiments we used $\varepsilon_{\text{train}} \in \{0.1, 0.2, 0.3, 0.4\}$. Both the encoder and the classifier are trained together. As for previous experiments on MNIST, we test using the standard pipeline of AutoAttack as done for the input space experiments in Section 5.1, with perturbations in input space and $\varepsilon_{\text{test}} \in \{0.05, 0.1, 0.2, 0.3\}$. For training, the values of α are computed with a triagular scheduler, as described in Section 5.1 for the SVHN setup.

Results. Table 2 shows the *image-space* robustness, while training with *latent-space* UTA perturbations and *latent-space* PGD, on the MNIST dataset. We observe that UTA-AT in latent space leads to better robustness/generalization trade-offs for each considered value of $\varepsilon_{\text{test}}$. For $\varepsilon_{\text{test}} \geq 0.2$, for UTA-AT, the robust accuracy falls to 0, yet the clean accuracy remains high, in sharp contrast with PGD-AT in latent space for which the clean accuracy falls to random for $\varepsilon_{\text{train}} \geq 0.2$ (see Fig. 13). See Fig. 13 for details on PGD robustness and clean accuracy.

5.3 Catastrophic overfitting

Fashion-MNIST. We used a LeNet model trained on the Fashion-MNIST dataset. Both for FGSM and single model UTA *with one step* we used during training $\varepsilon = 0.2$ and $\alpha = \varepsilon$; and for testing we used PGD with 20 iterations, $\varepsilon = 0.2$ and $\alpha = 0.01$. See App. B for further details.

CIFAR-10. For the experiments on CIFAR-10 we used five-model UTA sampled with MC Dropout, using ResNet-18 (He et al., 2016), see App. B for details.



(a) FGSM training

(b) single model UTA training

Figure 4: Catastrophic Overfitting (CO) on the **FashionMNIST** dataset using a LeNet model; results are averaged over 3 runs. **Left**: 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 FGSM after CO. **Right**: training with UTA-1 (single step) 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 AT lead to higher clean accuracy.

Results. Fig. 4a and 4b depict the results on FashionMNIST which evaluate if the methods are prone to Catastrophic Overfitting (CO). Relative to FGSM, single model UTA notably improves CO, as although there is a downward trend in PGD-20 accuracy after a certain number of iterations, it does not reduce to 0.

6 Discussion

Building on the well-established notion of uncertainty estimation—which gives high estimates *both* at the decision boundary and at un-explored regions of the input space (epistemic uncertainty)—we proposed uncertainty-targeted attacks that perturb a training sample in a direction that maximizes the uncertainty of the model.

Our extensive results on MNIST, Fashion-MNIST, SVHN and CIFAR-10, with three different models, and in input data and latent space, indicate that this approach is promising as it degrades less the clean test accuracy relative to PGD, while being as robust as PGD relative to standard training, and its one-step variant improves the reported catastrophic overfitting of FGSM. Furthermore, in contrast to PGD, our method supports very large training ε, which improves the model's robustness to ε larger than previously considered. Enhanced robustness to large ε is a desirable property as it shows the boundary decision is optimally far away from data points.

Potential directions include exploring different uncertainty estimation methods, as well as finding formal connections between UTA and online learning.

References

Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. In *NeurIPS*, 2020.

Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2018.07.023.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases*, pp. 387–402, 2013. ISBN 978-3-642-40994-3.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2206–2216. PMLR, 2020.

Ekin D. Cubuk, Barret Zoph, Samuel S. Schoenholz, and Quoc V. Le. Intriguing properties of adversarial examples, 2017.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv:1702.08608*, 2017.

Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059, 2016.

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

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

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*, volume 29, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Master's thesis, 2009.

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*, pp. 6405–6416, Red Hook, NY, USA, 2017. Curran Associates Inc.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016.

Yann Lecun and Corinna Cortes. The MNIST database of handwritten digits. 1998. URL http://yann.lecun.com/exdb/mnist/.

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pp. 2278–2324, 1998.

Chunchuan Lyu and Kaizhu Hug Liang. A unified gradient regularization family for adversarial examples. *arXiv:1511.06385*, 2015.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011. URL http://ufldl.stanford.edu/housenumbers/.

Gabriel Pereyra, G. Tucker, J. Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv:1701.06548*, 2017.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks, 2017.

Leslie N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472, 2017. doi: 10.1109/WACV.2017.58.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

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)*, pp. 6976–6987, 2019.

Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy? - A comprehensive study on the robustness of 18 deep image classification models. In *ECCV*, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://openreview.net/forum?id=rkZvSe-RZ.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2019.

Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal. On feature collapse and deep kernel learning for single forward pass uncertainty. *arXiv preprint arXiv:2102.11409*, 2021.

Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995. ISBN 0-387-94559-8.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning, 2015.

Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *ICLR*, 2020. URL https://openreview.net/forum?id=BJx040EFvH.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747*, 2017.

Oguz Kaan Yuksel, Sebastian U. Stich, Martin Jaggi, and Tatjana Chavdarova. Semantic perturbations with normalizing flows for improved generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7472–7482. PMLR, 2019.

Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan S. Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11278–11287. PMLR, 2020.

A UNCERTAINTY-BASED TOY EXPERIMENTS WITH GAUSSIAN PROCESS

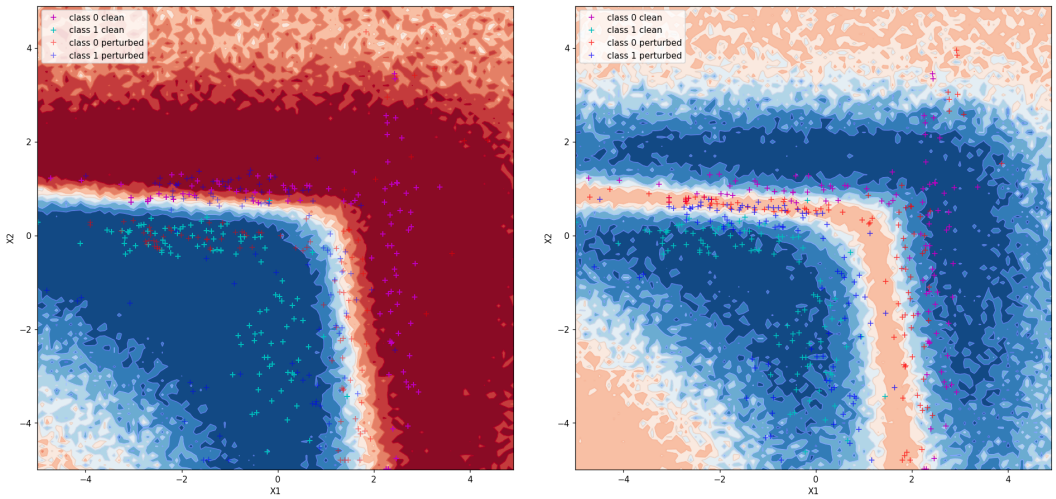
In this section, we consider additional uncertainty estimation methods, and we show that UTA can be implemented with these too. To further improve the reader's intuition of how this particular family of attack works we show the results of some toy experiments using the same non-isotropic dataset introduced in Section 4.1. However, in this section, we focus on estimating the uncertainty using Gaussian Processes–considered a gold standard in uncertainty estimation with low dimensional data.

In particular, we use the *Deterministic Uncertainty Estimation* (DUE) method (van Amersfoort et al., 2021). This method aims at estimating epistemic uncertainty using a single forward pass through the network by exploiting Gaussian Processes applied to deep architectures (Wilson et al., 2015). We choose this particular method because it is known to give higher uncertainty estimates to outliers relative to other popular methods (see Fig. 1 van Amersfoort et al., 2021).

A.1 VISUALIZATIONS OF THE PGD AND UTA PERTURBATIONS

In this section, we first train a model using DUE on the clean (unperturbed) 2D dataset, and then we depict the different attacks.

In Fig. 5 we can see the decision boundary and the uncertainty, measured in terms of entropy of the model in classifying the clean dataset (without adversarial training). In this case, as in 4.1, we observe that while the PGD-50 samples cross notably the boundary, the UTA perturbations do not. Moreover, UTA perturbations push the samples both towards the decision boundary and towards unexplored regions of the dataset. Both the PGD and UTA attacks are applied using $\varepsilon = 1$ and $\alpha = 0.1$.



(a) PGD-50 attack on DUE

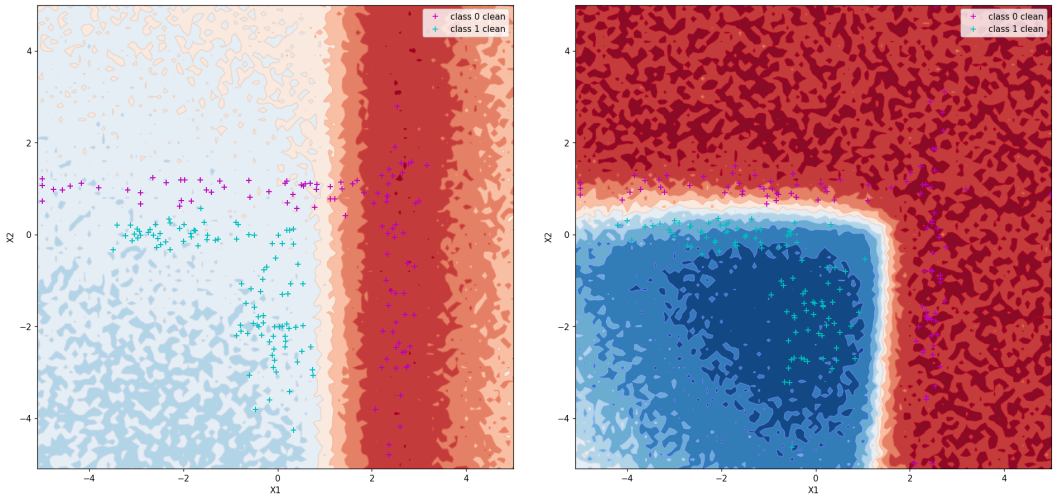
(b) UTA-50 attack on DUE

Figure 5: PGD and UTA attacks on a *fixed* DUE (van Amersfoort et al., 2021) model trained solely with clean data on the toy example from § 4.1. Light and dark blue crosses denote clean and perturbed samples of class 1, and similarly, pink and red crosses denote clean and perturbed samples of class 0. **Left**: the background depicts the loss landscape of the model (and its decision boundary), clean dataset and perturbed samples using the PGD-50 attack. Note how the PGD samples cross the boundary and go to the other class' region. **Right**: entropy of the model, clean dataset and UTA-50 perturbed samples. UTA samples do not cross the boundary and are able to go towards unexplored regions of the dataset.

A.2 ADVERSARIAL TRAINING USING THE DUE CLASSIFIER

In this section, we use the same setup as in the previous section, however, we train adversarially the DUE model using PGD and UTA attacks. For both PGD and UTA we use: $\varepsilon = 1$ and $\alpha = 0.1$.

Fig. 6 depicts the results. We observe that PGD-AT led to a very different decision boundary compared to the one obtained by training only with clean samples and leads to a model that is not able to classify the dataset with good performances. On the other hand, UTA-AT preserves a good decision boundary, very similar to the one shown in Figure 5.



(a) Decision boundary of a PGD AT model.

(b) Decision boundary of a UTA AT model.

Figure 6: PGD and UTA AT on a DUE model trained on a non-isotropic toy example dataset. **Left**: decision boundary obtained by training adversarially with PGD-50 perturbed samples. The model is not able to correctly classify the dataset. **Right**: decision boundary obtained by training adversarially with UTA-50 perturbed samples. The model is not is now able to classify correctly the dataset and the decision boudnary is much more similar to the one obtained by training with clean samples only (shown in Figure 5).

B DETAILS ON THE IMPLEMENTATION

Source code. Our source code is provided in this anonymous repository: https://anonymous.4open.science/r/Uncertainty-Targeted-Attacks-5BD3/README.md. In this section, we list the details of the implementation.

B.1 ARCHITECTURES & HYPERPARAMETERS

In this section, we describe in detail the architecture used for our experiments for the various datasets.

B.1.1 ARCHITECTURE FOR EXPERIMENTS ON MNIST IN THE IMAGE SPACE

For experiments on the **MNIST** dataset, we used a simple feedfoward network with 2 fully connected layers: the first with 28×28 inputs and 256 outputs, while the second with 256 inputs and 10 outputs. A flattening layer was used before the first fully connected layer, a $ReLU$ function between the first and the second layer and a $Softmax$ activation function after the second fully connected layer. The parameters of the models are initialized using PyTorch default initialization.

B.1.2 ARCHITECTURE FOR EXPERIMENTS ON FASHION-MNIST AND SVHN

FashionMNIST. We used a LeNet architecture as described in table 3. This network has been trained for 100 epochs using the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.001. The parameters of the models are initialized using PyTorch default initialization.

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 3: LeNet architecture used for experiments on **FashionMNIST** and for latent-space **MNIST** experiments. With $h\times w$ we denote the kernel size. With $c_{in} \rightarrow y_{out}$ we denote 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.

SVHN. For **SVHN** we increase the capacity of the LeNet model, as described in detail in Tab. 4. We used Adam (Kingma & Ba, 2015) as an optimization method and step size of 0.001.

B.1.3 ARCHITECTURE FOR EXPERIMENTS ON MNIST IN THE LATENT SPACE

For the latent space experiments the same architecture and training setting is used, but the network is split in an encoder part and a classifier part. For the encoder all the convolutional layers and the first fully connected layer of the LeNet model are used. The classifier consisted in the last two fully connected layer of model. Also in this case the network has been trained using the Adam optimizer with a learning rate of 0.001. Model's parameters are initialized using PyTorch default initialization.

LeNet
Input: $x \in \mathbb{R}^{3\times 28\times 28}$
conv. (kernel: 5×5, $3 \rightarrow 32$; padding: 2; stride: 1)
ReLU
max pooling (kernel: 2×2; stride: 2)
convolutional (kernel: 5×5, $32 \rightarrow 64$; stride: 1)
ReLU
max pooling (kernel: 2×2; stride: 2)
Flattening
fully connected ($64 \times 6 \times 6 \rightarrow 256$)
ReLU
fully connected ($256 \rightarrow 120$)
ReLU
fully connected ($120 \rightarrow 10$)
$Softmax(\cdot)$

Table 4: LeNet architecture used for experiments on **SVHN**. With $h\times w$ we denote the kernel size. With $c_{in} \rightarrow y_{out}$ we denote 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 (part of the ℓ–th layer)
Bypass:
conv. (ker: 1×1, $64 \rightarrow 64 \times \ell$; str: 2; pad: 1), if $\ell \neq 1$
Batch Normalization, if $\ell \neq 1$
Feedforward:
conv. (ker: 3×3, $64 \rightarrow 64 \times \ell$; str: $1_{\ell=1}/2_{\ell\neq1}$; 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: $x \in \mathbb{R}^{3\times 32\times 32}$
conv. (ker: 3×3; $3 \rightarrow 64$; str: 1; pad: 1)
Batch Normalization
$ReLU$
MCD ($p = 0.2$)
$3\times$ResBlock ($\ell = 1$)
$6\times$ResBlock ($l \in [2, 3, 4]$)
$ReLU$
AvgPool (ker:4×4)
Linear($512 \rightarrow 10$)

Table 5: 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 we list the layers sequentially, however, note that the bypass layers operate in parallel with the layers denoted as "feedforward" (He et al., 2016). The ResNet block for the model (right) differs if it is the first block in the network (following the input to the model).

B.2 Architecture for Experiments on CIFAR-10

The ResNet-18 setup on CIFAR-10 is as in (Andriushchenko & Flammarion, 2020). Table 5 lists the architecture used for the experiments on CIFAR-10. We use a ResNet18 (He et al., 2016) architecture, modified to accommodate the MC-dropout sampling procedure. The modification consists of adding a dropout layer with dropout probability $p = 0.2$ after each convolutional layer. In order to have a very fast version of the attack (same computational cost as FGSM) we use only one UTA step, and sample only one model with MC-dropout. For 9b and 9b, our models were trained for 200 epochs using the SGD optimizer with Nesterov momentum and with an initial learning rate of 0.1 decayed by factor of 15 after 60, 120, 160 epochs. For Fig. 9c and Fig. 3 in order to reach faster convergence we trained a ResNet18 model for 90 epochs with MCD using a cyclic learning rate scheduling (Smith, 2017) with a maximum LR of 0.2 for the FGSM, R-FGSM and PGD-2 and of 0.1 for the UTA-1-5 and UTA-2-5 attacks.

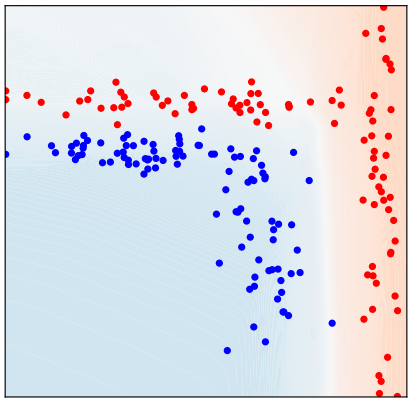


Figure 7: TRADES applied on our 2D toy example with $\lambda = 1$ and for a large ε similar to the one used in 1c. Red samples in the top left corner end up being wrongly classified as the TRADES objective tries to align predictions of both red and blue datapoints in this area.

C Additional Results and Analysis

C.1 TRADES

TRADES (Zhang et al., 2019) adds a regularizer to encourage the smoothness of the output around samples from the data distribution. It is described by the following objective:

$$\min_f \mathbb{E}\big[\mathcal{L}(f(X), Y) + \lambda \max_{X' \in \mathbb{B}(X,\varepsilon)} \mathcal{L}(f(X), f(X'))\big]$$

Where $\mathcal{L}$ is the loss, f the model, and $\mathbb{B}(X, \varepsilon)$ the ball centered on X of radius ε.

This approach differs from UTA in several important points:

- It finds a perturbation X' that maximizes the loss between $f(X)$ and $f(X')$, which makes it more similar to standard PGD than to UTA. It will share a similar drawback as it will also tend to cross the decision boundary.
- By minimizing $\mathcal{L}(f(X), f(X'))$, TRADES is increasing the similarity between $f(X)$ and $f(X')$, hence smoothing the output in the neighborhood of X. In contrast, given an input/label pair (X, Y), given the adversarial sample X', UTA would try to minimize $\mathcal{L}(f(X'), Y)$.

In Fig. 7 we apply TRADES to our 2D toy example. We observe TRADES share the same problem as PGD as it can fail to cope with non-isotropic ε.

C.2 Visual Appearance of the Attacks

In Fig. 8 we use large ε-ball to obtain clearly visible difference between the PGD and UTA-PGD attacks, as well as to see if the insights from the experiment from Fig. 1, § 4.1 could extend to real world setups. The topmost row depicts clean samples from CIFAR-10. Using an already trained classifier on CIFAR-10, we show how UTA and PGD attacks look, in the middle and bottom rows, respectively. We observe that for large ε-ball PGD attacks *can* make the content non-recognizable for human eye, whereas the class of UTA perturbed samples remains perceptible.

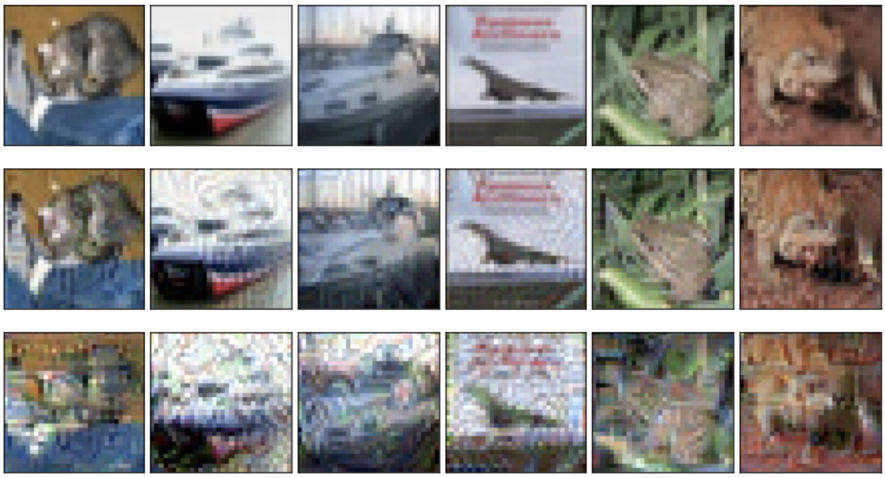
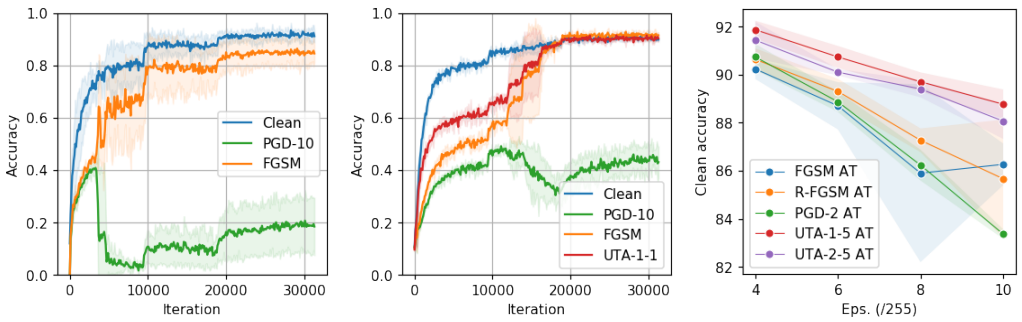


Figure 8: Illustration of the difference between PGD and UTA attacks to classifier trained on CIFAR-10: *(i)* top row: clean samples, *(ii)* middle row: UTA perturbations, *(iii)* bottom row: PGD attacks, where for UTA and PGD we use same setup (1000 steps, step size of 0.001, $\varepsilon = \infty$). We use large number of steps to verify empirically if the difference between UTA and PGD depicted in Fig. 1 holds on real-world datasets as well. Contrary to the PGD-perturbed samples, the correct class of the UTA-perturbed ones remains perceptible. See § C.2.

C.3 Omitted Results

In this section we show additional results omitted from the paper, obtained by applying adversarial attacks both in the image and in the latent space.

In Fig. 9 we evaluate catastrophic overfitting (CO) on the CIFAR-10 (Krizhevsky, 2009) dataset, where we use single model UTA with single step, see § B.2 for details on the implementation. We observe that UTA is notably more robust to CO relative to FGSM.



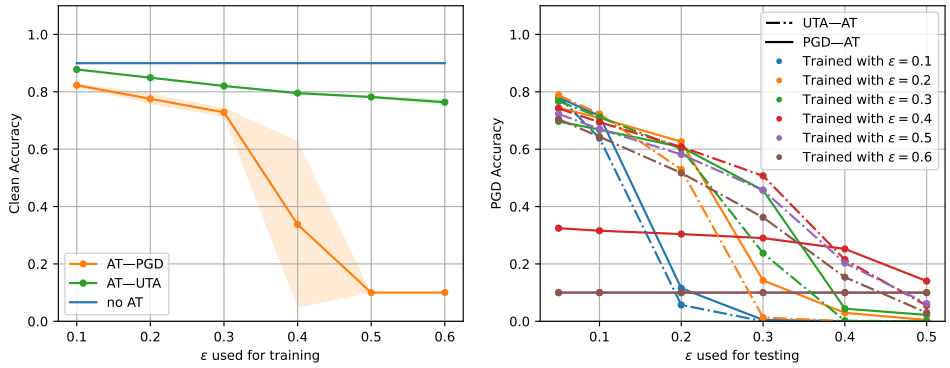
(a) training with FGSM

(b) training with UTA

(c) Clean accuracy obtained from AT models trained with different ε

Figure 9: Catastrophic overfitting (CO) on CIFAR-10 using ResNet 18, averaged over 3 runs. (a): training with FGSM–with step size $\alpha = 8/255$, $\varepsilon = \alpha$; 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 ($\varepsilon = \alpha = 8/255$). We observe that UTA is more robust to CO relative to 9a. (c): Clean accuracy comparison from different AT and UTA methods with different values of the perturbation radius ε. We observe that UTA methods lead to higher clean accuracy than PGD methods. In Fig. 3 we show the adversarial accuracy related to the same experiments.

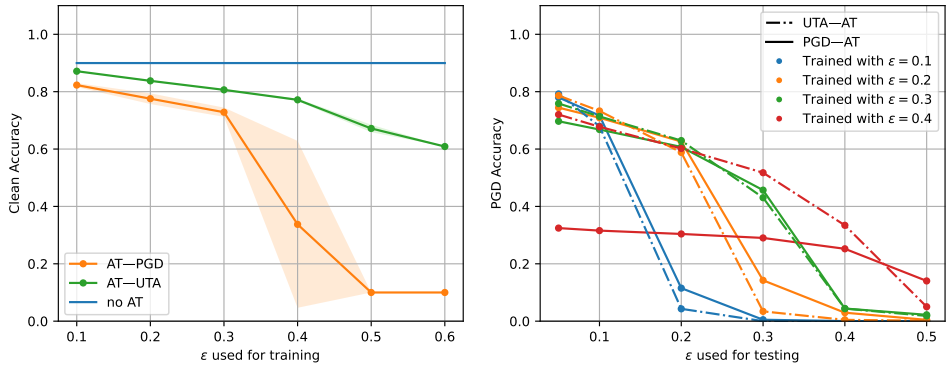
Fig. 10 depicts additional results to those in Table 1 on Fashion-MNIST, whereas Fig. 11 uses ensemble of five models. The robustness naturally decreases when increasing $\varepsilon_{\text{test}}$. Note that the PGD robustness for input data normalized between $[0, 1]$ and for $\varepsilon_{\text{test}} = 0.5$ should be close to 0 as it



(a) Clean test accuracy (b) PGD robustness

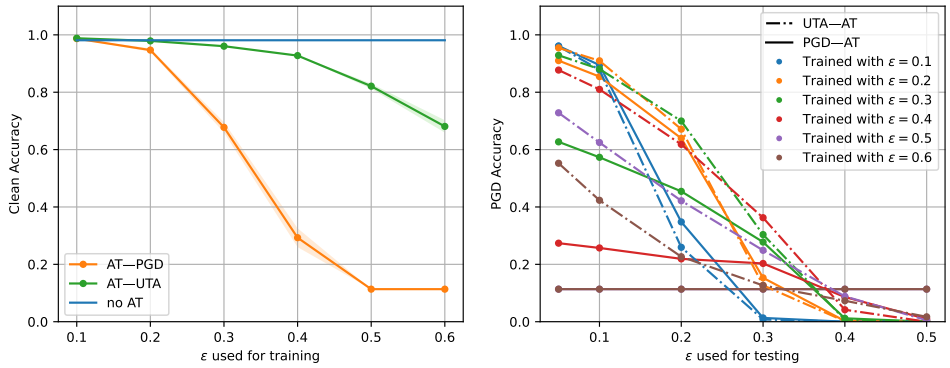
Figure 10: Full comparison (in addition to the results in Table 1) between PGD and *single-model UTA* on **Fashion-MNIST**, results are averaged over multiple runs. **Left:** accuracy on the unperturbed test dataset, for varying $\varepsilon_{\text{train}}$. **Right:** PGD robustness on the test data points, for varying $\varepsilon_{\text{test}}$ (x-axis), where dashed-dotted curves are UTA, and solid curves are PGD.

is always possible for such large ε to find an intersection between any pair of L_∞ balls. Similarly, Fig. 12 shows the results on MNIST using an ensemble of five models. Finally, Fig. 13 depicts the clean test accuracy on MNIST, where for the UTA attacks we use ensemble of five models.



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

Figure 11: Comparison between PGD and *UTA 5-model ensemble* on **Fashion-MNIST**, results are averaged over multiple runs. **Left:** accuracy on the unperturbed test dataset, for varying ε. **Right:** PGD-10 robustness, for varying ε, where dashed-dotted curves are UTA, and solid curves are PGD.



(a) Clean accuracy on test dataset

(b) PGD robustness on test dataset

Figure 12: Comparison between PGD and *five-model ensemble UTA* on **MNIST**, results are averaged over multiple runs. **Left:** accuracy on the unperturbed test dataset, for varying ε. **Right:** PGD-10 robustness, for varying ε, where dashed-dotted curves are UTA, and solid curves are PGD.

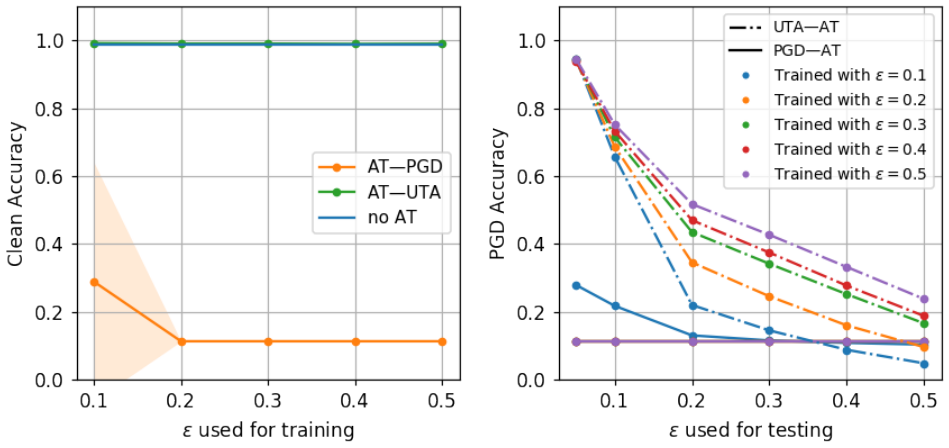


Figure 13: Comparison between latent-space PGD and latent-space UTA attacks on **MNIST**, where for the latter we use ensemble of five models. See § 5.2 for setup summary and discussion. **Left**: accuracy on the unperturbed test dataset for varying ε. Training with UTA perturbations in latent space is not affecting the clean accuracy, unlike PGD. **Right**: PGD robustness for varying ε, dashed-dotted curves are UTA and solid curves are PGD. 10 random restarts are used for the PGD testing.