Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem

Matthias Hein
University of Tübingen

Maksym Andriushchenko
Saarland University

Julian Bitterwolf
University of Tübingen

Abstract

Classifiers used in the wild, in particular for safety-critical systems, should not only have good generalization properties but also should know when they don't know, in particular make low confidence predictions far away from the training data. We show that ReLU type neural networks which yield a piecewise linear classifier function fail in this regard as they produce almost always high confidence predictions far away from the training data. For bounded domains like images we propose a new robust optimization technique similar to adversarial training which enforces low confidence predictions far away from the training data. We show that this technique is surprisingly effective in reducing the confidence of predictions far away from the training data while maintaining high confidence predictions and test error on the original classification task compared to standard training.

1. Introduction

Neural networks have recently obtained state-of-the-art performance in several application domains like object recognition and speech recognition. They have become the de facto standard for many learning tasks. Despite this great success story and very good prediction performance there are also aspects of neural networks which are undesirable. One property which is naturally expected from any classifier is that it should know when it does not know or said more directly: far away from the training data a classifier should not make high confidence predictions. This is particularly important in safety-critical applications like autonomous driving or medical diagnosis systems where such an input should either lead to the fact that other redundant sensors are used or that a human doctor is asked to check the diagnosis. It is thus an important property of a classifier which however has not received much attention despite the fact that it seems to be a minimal requirement for any classifier.

There have been many cases reported where high confidence predictions are made far away from the training data by neural networks, e.g. fooling images [30], for out-of-distribution images [15] or in a medical diagnosis task [23]. Moreover, it has been observed that, even on the original task, neural networks often produce overconfident predictions [12]. A related but different problem are adversarial samples where very small modifications of the input can change the classifier decision [34, 11, 32]. Apart from methods which provide robustness guarantees for neural networks [14, 37, 31, 26] which give still only reasonable guarantees for small networks, up to our knowledge the only approach which has not been broken again [6, 5, 2] is adversarial training [25] using robust optimization techniques.

While several methods have been proposed to adjust overconfident predictions on the true input distribution using softmax calibration [12], ensemble techniques [20] or uncertainty estimation using dropout [10], only recently the detection of out-of-distribution inputs [15] has been tackled. The existing approaches basically either use adjustment techniques of the softmax outputs [9, 24] by temperature rescaling [12] or they use a generative model like a VAE or GAN to model boundary inputs of the true distribution [22, 36] in order to discriminate in-distribution from out-of-distribution inputs directly in the training process. While all these approaches are significant steps towards obtaining more reliable classifiers, the approaches using a generative model have been recently challenged by [28, 16] which report that generative approaches can produce highly confident density estimates for inputs outside of the class they are supposed to model. Moreover, note that the quite useful models for confidence calibration on the input distribution like [10, 12, 20] cannot be used for out-of-distribution detection as it has been observed in [23]. Another approach is the introduction of a rejection option into the classifier [35, 4], in order to avoid decisions the classifier is not certain about.

In this paper we will show that for the class of ReLU

networks, that are networks with fully connected, convolutional and residual layers, where just ReLU or leaky ReLU are used as activation functions and max or average pooling for convolution layers, basically any neural network which results in a piecewise affine classifier function, produces arbitrarily high confidence predictions far away from the training data. This implies that techniques which operate on the output of the classifier cannot identify these inputs as out-of-distribution inputs. On the contrary we formalize the well known fact that RBF networks produce almost uniform confidence over the classes far away from the training data, which shows that there exist classifiers which satisfy the minimal requirement of not being confident in areas where one has never seen data. Moreover, we propose a robust optimization scheme motivated by adversarial training [25] which simply enforces uniform confidence predictions on noise images which are by construction far away from the true images. We show that our technique not only significantly reduces confidence on such noise images, but also on other unrelated image classification tasks and in some cases even for adversarial samples generated for the original classification task. The training procedure is simple, needs no adaptation for different out-of-distribution tasks, has similar complexity as standard adversarial training and achieves similar or marginally worse generalization performance on the original classification task.

2. ReLU networks produce piecewise affine functions

We quickly review in this section the fact that ReLU networks lead to continuous piecewise affine classifiers, see [1, 8], which we briefly summarize in order to set the ground for our main theoretical result in Section 3.

Definition 2.1. *A function $f : \mathbb{R}^d \to \mathbb{R}$ is called* piecewise affine *if there exists a finite set of polytopes $\{Q_r\}_{r=1}^M$ (referred to as* linear regions *of f) such that $\cup_{r=1}^M Q_r = \mathbb{R}^d$ and f is an affine function when restricted to every Q_r.*

Feedforward neural networks which use piecewise affine activation functions (e.g. ReLU, leaky ReLU) and are linear in the output layer can be rewritten as continuous piecewise affine functions [1]. This includes fully connected, convolutional, residual layers and even skip connections as all these layers are just linear mappings. Moreover, it includes further average pooling and max pooling. More precisely, the classifier is a function $f : \mathbb{R}^d \to \mathbb{R}^K$, where K are the number of classes, such that each component $f_i : \mathbb{R}^d \to \mathbb{R}$, is a continuous piecewise affine function and the K components $(f_i)_{i=1}^K$ have the same set of linear regions. Note that explicit upper bounds on the number of linear regions have been given [27].

In the following we follow [8]. For simplicity we just present fully connected layers. Denote by $\sigma : \mathbb{R} \to \mathbb{R}$, $\sigma(t) = \max\{0, t\}$, the ReLU activation function, by $L+1$ the number of layers and $W^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ and $b^{(l)} \in \mathbb{R}^{n_l}$ respectively are the weights and offset vectors of layer l, for $l = 1, \dots, L+1$ and $n_0 = d$. For $x \in \mathbb{R}^d$ one defines $g^{(0)}(x) = x$. Then one can recursively define the pre- and post-activation output of every layer as

$$f^{(k)}(x) = W^{(k)} g^{(k-1)}(x) + b^{(k)}, \quad \text{and}$$
$$g^{(k)}(x) = \sigma(f^{(k)}(x)), \quad k = 1, \dots, L,$$

so that the resulting classifier is obtained as $f^{(L+1)}(x) = W^{(L+1)} g^{(L)}(x) + b^{(L+1)}$.

Let $\Delta^{(l)}, \Sigma^{(l)} \in \mathbb{R}^{n_l \times n_l}$ for $l = 1, \dots, L$ be diagonal matrices defined elementwise as

$$\Delta^{(l)}(x)_{ij} = \begin{cases} \operatorname{sign}(f_i^{(l)}(x)) & \text{if } i = j, \\ 0 & \text{else.} \end{cases},$$

$$\Sigma^{(l)}(x)_{ij} = \begin{cases} 1 & \text{if } i = j \text{ and } f_i^{(l)}(x) > 0, \\ 0 & \text{else.} \end{cases}.$$

Note that for leaky ReLU the entries would be 1 and α instead. This allows to write $f^{(k)}(x)$ as composition of affine functions, that is

$$\begin{aligned} f^{(k)}(x) =& W^{(k)} \Sigma^{(k-1)}(x) \Big(W^{(k-1)} \Sigma^{(k-2)}(x) \\ & \times \Big(\dots \Big(W^{(1)} x + b^{(1)} \Big) \dots \Big) + b^{(k-1)} \Big) + b^{(k)}, \end{aligned}$$

We can further simplify the previous expression as $f^{(k)}(x) = V^{(k)} x + a^{(k)}$, with $V^{(k)} \in \mathbb{R}^{n_k \times d}$ and $a^{(k)} \in \mathbb{R}^{n_k}$ given by

$$V^{(k)} = W^{(k)} \Big(\prod_{l=1}^{k-1} \Sigma^{(k-l)}(x) W^{(k-l)} \Big) \quad \text{and}$$
$$a^{(k)} = b^{(k)} + \sum_{l=1}^{k-1} \Big(\prod_{m=1}^{k-l} W^{(k+1-m)} \Sigma^{(k-m)}(x) \Big) b^{(l)}.$$

The polytope $Q(x)$, the linear region containing x, can be characterized as an intersection of $N = \sum_{l=1}^L n_l$ half spaces given by

$$\Gamma_{l,i} = \big\{ z \in \mathbb{R}^d \,\big|\, \Delta^{(l)}(x) \big(V_i^{(l)} z + a_i^{(l)} \big) \geq 0 \big\},$$

for $l = 1, \dots, L$, $i = 1, \dots, n_l$, namely

$$Q(x) = \bigcap_{l=1,\dots,L} \bigcap_{i=1,\dots,n_l} \Gamma_{l,i}.$$

Note that N is also the number of hidden units of the network. Finally, we can write

$$f^{(L+1)}(z)\Big|_{Q(x)} = V^{(L+1)}z + a^{(L+1)},$$

which is the affine restriction of f to $Q(x)$.

3. Why ReLU networks produce high confidence predictions far away from the training data

With the explicit description of the piecewise linear classifier resulting from a ReLU type network from Section 2, we can now formulate our main theorem. It shows that, as long a very mild condition on the network holds, for any $\epsilon > 0$ one can always find for (almost) **all** directions an input z far away from the training data which realizes a confidence of $1 - \epsilon$ on z for a certain class. However, before we come to this result, we first need a technical lemma needed in the proof, which uses that all linear regions are polytopes and thus convex sets.

Lemma 3.1. *Let $\{Q_i\}_{l=1}^R$ be the set of linear regions associated to the ReLU-classifier $f : \mathbb{R}^d \to \mathbb{R}^K$. For any $x \in \mathbb{R}^d$ there exists $\alpha \in \mathbb{R}$ with $\alpha > 0$ and $t \in \{1, \ldots, R\}$ such that $\beta x \in Q_t$ for all $\beta \geq \alpha$.*

All the proofs can be found in the supplementary material. Using Lemma 3.1 we can now state our first main result.

Theorem 3.1. *Let $\mathbb{R}^d = \cup_{l=1}^R Q_l$ and $f(x) = V^l x + a^l$ be the piecewise affine representation of the output of a ReLU network on Q_l. Suppose that V^l does not contain identical rows for all $l = 1, \ldots, R$, then for almost any $x \in \mathbb{R}^d$ and $\epsilon > 0$ there exists an $\alpha > 0$ and a class $k \in \{1, \ldots, K\}$ such that for $z = \alpha x$ it holds*

$$\frac{e^{f_k(z)}}{\sum_{r=1}^K e^{f_r(z)}} \geq 1 - \epsilon.$$

Moreover, $\lim_{\alpha \to \infty} \frac{e^{f_k(\alpha x)}}{\sum_{r=1}^K e^{f_r(\alpha x)}} = 1.$

Please note that the condition that for a region the linear part V^l need not contain two identical rows is very weak. It is hardly imaginable that this is ever true for a normally trained network unless the output of the network is constant anyway. Even if it is true, it just invalidates the assertion of the theorem for the points lying in this region. Without explicitly enforcing this condition it seems impossible that this is true for all possible asymptotic regions extending to infinity (see Figure 1). However, it is also completely open how this condition could be enforced during training of the network.

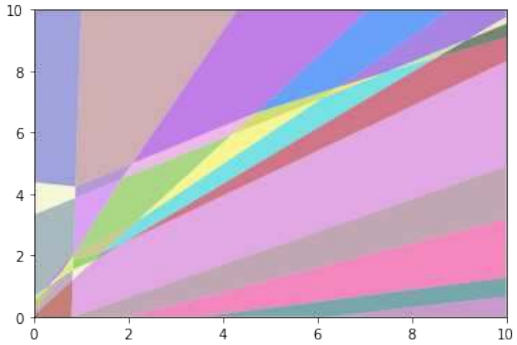

Figure 1: A decomposition of $\mathbb{R}^2$ into a finite set of polytopes for a two-hidden layer ReLU network. The outer polytopes extend to infinity. This is where ReLU networks realize arbitrarily high confidence predictions. The picture is produced with the code of [17].

The result implies that for ReLU networks there exist infinitely many inputs which realize arbitrarily high confidence predictions of the networks. It is easy to see that the temperature rescaling of the softmax, $\frac{e^{f_k(x)/T}}{\sum_{l=1}^K e^{f_l(x)/T}}$, for temperature $T > 0$, as used in [24], will not be able to detect these cases, in particular since the first step of the method in [24] consists of going in the direction of increasing confidence. Also it is obvious that using a reject option in the classifier, see e.g. [3], will not help to detect these instances either. The result is negative in the sense that it looks like that without modifying the architecture of a ReLU network it is impossible to prevent this phenomenon. Please note that from the point of view of Bayesian decision theory the softmax function is the correct transfer function [21] for the cross-entropy loss turning the classifier output $f_k(x)$ into an estimate $\mathrm{P}(Y = k\,|x, f) = \frac{e^{f_k(x)}}{\sum_{l=1}^K e^{f_l(x)}}$ for the conditional probability at x.

While the previous result seems not to be known, the following result is at least qualitatively known [11] but we could not find a reference for it. In contrast to the ReLU networks it turns out that Radial Basis Function (RBF) networks have the property to produce approximately uniform confidence predictions far away from the training data. Thus there exist classifiers which satisfy the minimal requirement which we formulated in Section 1. In the following theorem we explicitly quantify what "far away" means in terms of parameters of the RBF classifier and the training data.

Theorem 3.2. *Let $f_k(x) = \sum_{l=1}^N \alpha_{kl} e^{-\gamma \|x - x_l\|_2^2}$, $k = 1, \ldots, K$ be a RBF-network trained with cross-entropy loss on the training data $(x_i, y_i)_{i=1}^N$. We define $r_{\min} =$*

$\min_{l=1,\ldots,N} \|x - x_l\|_2$ *and* $\alpha = \max_{r,k} \sum_{l=1}^{N} |\alpha_{rl} - \alpha_{kl}|$. *If* $\epsilon > 0$ *and*

$$r_{\min}^2 \geq \frac{1}{\gamma} \log\Big(\frac{\alpha}{\log(1+K\epsilon)}\Big),$$

then for all $k = 1, \ldots, K$,

$$\frac{1}{K} - \epsilon \;\leq\; \frac{e^{f_k(x)}}{\sum_{r=1}^{K} e^{f_r(x)}} \;\leq\; \frac{1}{K} + \epsilon.$$

We think that it is a very important open problem to realize a similar result as in Theorem 3.2 for a class of neural networks. Note that arbitrarily high confidence predictions for ReLU networks can be obtained only if the domain is unbounded e.g. $\mathbb{R}^d$. However, images are contained in $[0,1]^d$ and thus Theorem 3.1 does not directly apply, even though the technique can in principle be used to produce high-confidence predictions (see experiments in the supplement). In the next section we propose a novel training scheme enforcing low confidence predictions on inputs far away from the training data.

4. Adversarial Confidence Enhanced Training

In this section we suggest a simple way to adjust the confidence estimation of a neural network far away from the training data, not necessarily restricted to ReLU networks studied in Theorem 3.1. Theorem 3.1 tells us that for ReLU networks a post-processing of the softmax scores is not sufficient to avoid high-confidence predictions far away from the training data - instead there seem to be two potential ways to tackle the problem: a) one uses an extra generative model either for the in-distribution or for the out-distribution or b) one modifies directly the network via an adaptation of the training process so that uniform confidence predictions are enforced far away from the training data. As recently problems with generative models have been pointed out which assign high confidence to samples from the out-distribution [28] and thus a) seems less promising we explore approach b).

We assume that it is possible to characterize a distribution of data points p_{out} on the input space for which we are sure that they do not belong to the true distribution p_{in} resp. the set of the intersection of their supports has zero or close to zero probability mass. An example of such an out-distribution p_{out} would be the uniform distribution on $[0,1]^{w \times h}$ ($w \times h$ gray scale images) or similar noise distributions. Suppose that the in-distribution consists of certain image classes like handwritten digits, then the probability mass of all images of handwritten digits under the p_{out} is zero (if it is really a low-dimensional manifold) or close to zero.

In such a setting the training objective can be written as a sum of two losses:

$$\frac{1}{N} \sum_{i=1}^{N} L_{CE}(y_i, f(x_i)) + \lambda\, \mathbb{E}\big[L_{p_{\text{out}}}(f, Z)\big], \qquad (1)$$

where $(x_i, y_i)_{i=1}^N$ is the i.i.d. training data, Z has distribution p_{out} and

$$L_{CE}(y_i, f(x_i)) = -\log\Big(\frac{e^{f_{y_i}(x_i)}}{\sum_{k=1}^{K} e^{f_k(x_i)}}\Big) \qquad (2)$$

$$L_{p_{\text{out}}}(f, z) = \max_{l=1,\ldots,K} \log\Big(\frac{e^{f_l(z)}}{\sum_{k=1}^{K} e^{f_k(z)}}\Big). \qquad (3)$$

L_{CE} is the usual cross entropy loss on the original classification task and $L_{p_{\text{out}}}(f, z)$ is the maximal log confidence over all classes where the confidence of class l is given by, $\frac{e^{f_l(z)}}{\sum_{k=1}^{K} e^{f_k(z)}}$, with the softmax function as the link function. The full loss can be easily minimized by using SGD with batchsize B for the original data and adding $\lceil \lambda B \rceil$ samples from p_{out} on which one enforces a uniform distribution over the labels. We call this process in the following *confidence enhancing data augmentation (CEDA)*. We note that in a concurrent paper [16] the same scheme has been proposed, where they use as p_{out} existing large image datasets, whereas we favor an agnostic approach where p_{out} models a certain "noise" distribution on images.

The problem with CEDA is that it might take too many samples to enforce low confidence on the whole out-distribution. Moreover, it has been shown in the area of adversarial manipulation that data augmentation is not sufficient for robust models and we will see in Section 5 that indeed CEDA models still produce high confidence predictions in a neighborhood of noise images. Thus, we propose to use ideas from robust optimization similar to adversarial training which [34, 11, 25] apply to obtain robust networks against adversarial manipulations. Thus we are enforcing low confidence not only at the point itself but actively minimize the worst case in a neighborhood of the point. This leads to the following formulation of *adversarial confidence enhancing training (ACET)*

$$\frac{1}{N} \sum_{i=1}^{N} L_{CE}(y_i, f(x_i)) + \lambda\, \mathbb{E}\Big[\max_{\|u - Z\|_p \leq \epsilon} L_{p_{\text{out}}}(f, u)\Big], \qquad (4)$$

where in each SGD step one solves (approximately) for a given $z \sim p_{\text{out}}$ the optimization problem:

$$\max_{\|u - z\|_p \leq \epsilon} L_{p_{\text{out}}}(f, u). \qquad (5)$$

In this paper we use always $p = \infty$. Note that if the distributions p_{out} and p_{in} have joint support, the maximum in (5) could be obtained at a point in the support of the true distribution. However, if p_{out} is a generic noise distribution like uniform noise or a smoothed version of it, then the number of cases where this happens has probability mass close to zero under p_{out} and thus does not negatively influence in (4) the loss L_{CE} on the true distribution. The optimization of ACET in (4) can be done using an adapted version of the PGD method of [25] for adversarial training where one performs projected gradient descent (potentially for a few restarts) and uses the u realizing the worst loss for computing the gradient. The resulting samples are more informative and thus lead to a faster and more significant reduction of high confidence predictions far away from the training data. We use $\epsilon = 0.3$ for all datasets. We present in Figure 2 and 3 for MNIST and CIFAR-10 a few noise images together with their adversarial modification u generated by applying PGD to solve (5). One can observe that the generated images have no structure resembling images from the in-distribution.

5. Experiments

In the evaluation, we follow [15, 24, 22] by training on one dataset and evaluating the confidence on other out of distribution datasets and noise images. In contrast to [24, 22] we neither use a different parameter set for each test dataset [24] nor do we use one of the test datasets during training [22]. More precisely, we train on MNIST, SVHN, CIFAR-10 and CIFAR-100, where we use the LeNet architecture on MNIST taken from [25] and a ResNet architecture [13] for the other datasets. We also use standard data augmentation which includes random crops for all datasets and random mirroring for CIFAR-10 and CIFAR-100. For the generation of out-of-distribution images from p_{out} we proceed as follows: half of the images are generated by randomly permuting pixels of images from the training set and half of the images are generated uniformly at random. Then we apply to these images a Gaussian filter with standard deviation $\sigma \in [1.0, 2.5]$ as lowpass filter to have more low-frequency structure in the noise. As the Gaussian filter leads to a contrast reduction we apply afterwards a global rescaling so that the maximal range of the image is again in $[0, 1]$.

Training: We train each model normally (plain), with confidence enhancing data augmentation (CEDA) and with adversarial confidence enhancing training (ACET). It is well known that weight decay alone reduces overconfident predictions. Thus we use weight decay with regularization parameter $5 \cdot 10^{-4}$ for all models leading to a strong baseline (plain). For CEDA (1) and ACET (4) we both use $\lambda = 1$, that means 50% of the samples in each batch are from the original training set and 50% are noise samples as described before. For ACET we use $p = \infty$ and $\epsilon = 0.3$ and optimize with PGD [25] using 40 iterations and stepsize 0.0075 for all datasets. All models are trained for 100 epochs with ADAM [18] on MNIST and SGD+momentum for SVHN/CIFAR-10/CIFAR-100. The initial learning rate is 10^{-3} for MNIST and 0.1 for SVHN/CIFAR-10 and it is reduced by a factor of 10 at the 50th, 75th and 90th of the in total 100 epochs. More results and details can be found in the supplementary material. The code is available at https://github.com/max-andr/relu_networks_overconfident.

Evaluation: We report for each model (plain, CEDA, ACET) the test error and the mean maximal confidence (for each point this is $\max_{k=1,\dots,K} \frac{e^{f_k(x)}}{\sum_{l=1}^{K} e^{f_l(x)}}$), denoted as MMC, on the test set. In order to evaluate how well we reduce the confidence on the out-distribution, we use four datasets on CIFAR-10 [19] and SVHN [29] (namely among CIFAR-10, CIFAR-100, SVHN, ImageNet-, which is a subset of ImageNet where we removed classes similar to CIFAR-10, and the classroom subset of LSUN [39] we use the ones on which we have *not* trained) and for MNIST we evaluate on EMNIST [7], a grayscale version of CIFAR-10 and Fashion MNIST [38]. Additionally, we show the evaluation on noise, adversarial noise and adversarial samples. The noise is generated in the same way as the noise we use for training. For adversarial noise, where we maximize the maximal confidence over all classes (see $L_{p_{\text{out}}}(f, z)$ in (3)), we use PGD with 200 iterations and stepsize 0.0075 in the ϵ ball wrt the $\|\cdot\|_\infty$-norm with $\epsilon = 0.3$ (same as in training). Note that for training we use only 40 iterations, so that the attack at test time is significantly stronger. Finally, we check also the confidence on adversarial samples computed for the test set of the in-distribution dataset using 80 iterations of PGD with $\epsilon = 0.3$, stepsize 0.0075 for MNIST and $\epsilon = 0.1$, stepsize 0.0025 for the other datasets. The latter two evaluation modalities are novel compared to [15, 24, 22]. The adversarial noise is interesting as it actively searches for images which still yield high confidence in a neighborhood of a noise image and thus is a much more challenging than the pure evaluation on noise. Moreover, it potentially detects an over-adaptation to the noise model used during training in particular in CEDA. The evaluation on adversarial samples is interesting as one can hope that the reduction of the confidence for out-of-distribution images also reduces the confidence of adversarial samples as typically adversarial samples are off the data

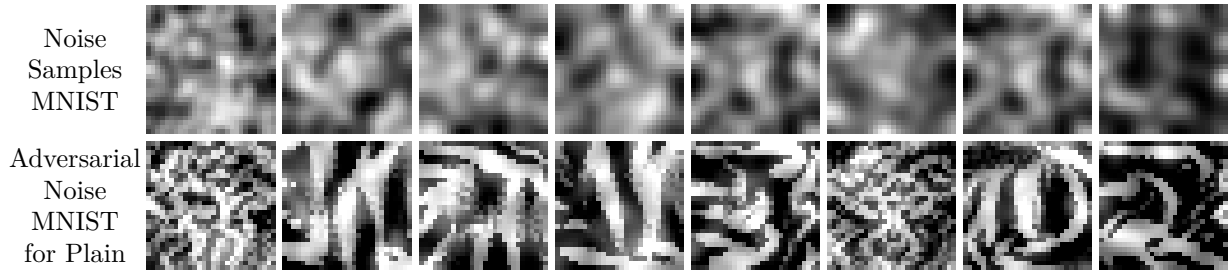

Figure 2: Top row: our generated noise images based on uniform noise resp. permuted MNIST together with a Gaussian filter and contrast rescaling. Bottom row: for each noise image from above we generate the corresponding adversarial noise image using PGD with 40 iterations maximizing the second part of the loss in ACET for the plain model. Note that neither in the noise images nor in the adversarially modified ones there is structure similar to a MNIST image. For ACET and CEDA it is very difficult to generate adversarial noise images for the fully trained models thus we omit them.

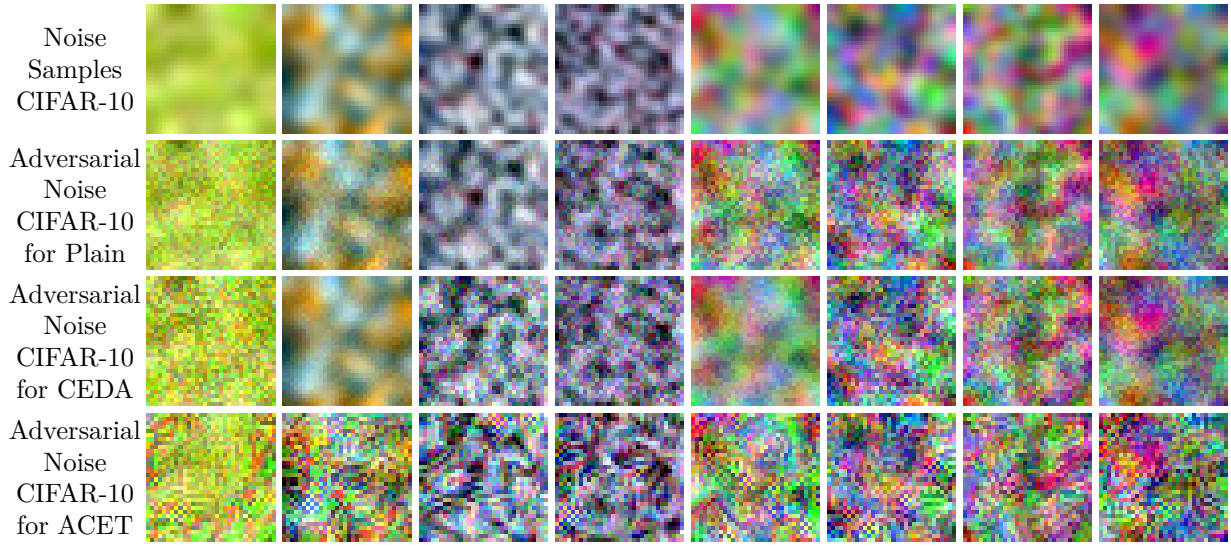

Figure 3: Top row: our generated noise images based on uniform noise resp. permuted MNIST together with a Gaussian filter and contrast rescaling (similar to Figure 2). Bottom rows: the corresponding adversarial images for the plain, CEDA, and ACET models. Neither the noise nor the adversarial noise images show similarity to CIFAR-10 images.

manifold [33] and thus are also out-of-distribution samples (even though their distance to the true distribution is small). Note that our models have never seen adversarial samples during training, they only have been trained using the adversarial noise. Nevertheless our ACET model can reduce the confidence on adversarial samples. As evaluation criteria we use the average confidence, the area under the ROC curve (AUC) where we use the confidence as a threshold for the detection problem (in-distribution vs. out-distribution). Moreover, we report in the same setting the false positive rate (FPR) when the true positive rate (TPR) is fixed to 95%. All results can be found in Table 1.

Main Results: In Table 1, we show the results of plain (normal training), CEDA and ACET. First of all, we observe that there is almost no difference between the test errors of all three methods. Thus improving the confidence far away from the training data does not impair the generalization performance. We also see that the plain models always produce high confidence predictions on noise images and completely fail on adversarial noise. CEDA produces low confidence on noise images but mostly fails on adversarial noise which was to be expected as similar findings have been made for the creation of adversarial samples. Only ACET consistently produces low confidence predictions on adversarial noise and has high AUROC. For the out-of-distribution datasets CEDA and ACET improve most of the time the maximum confidence and the AUROC, sometimes with very strong improvements like on MNIST evaluated on FMNIST or SVHN evaluated on LSUN. However, one observes that it is more difficult to reduce the confidence for related tasks e.g. MNIST evaluated on EMNIST or CIFAR-10 evaluated on LSUN, where the image structure is more similar. Finally, an interesting outcome is that ACET reduces

Trained on **MNIST**	Plain (TE: **0.51%**)			CEDA (TE: 0.74%)			ACET (TE: 0.66%)		
	MMC	AUROC	FPR@95	MMC	AUROC	FPR@95	MMC	AUROC	FPR@95
MNIST	**0.991**	–	–	0.987	–	–	0.986	–	–
FMNIST	0.654	0.972	0.121	0.373	0.994	0.027	**0.239**	**0.998**	**0.003**
EMNIST	0.821	0.883	0.374	0.787	0.895	0.358	**0.752**	**0.912**	**0.313**
grayCIFAR-10	0.492	0.996	0.003	0.105	**1.000**	**0.000**	**0.101**	**1.000**	**0.000**
Noise	0.463	0.998	0.000	**0.100**	**1.000**	**0.000**	**0.100**	**1.000**	**0.000**
Adv. Noise	1.000	0.031	1.000	**0.102**	**0.998**	**0.002**	0.162	0.992	0.042
Adv. Samples	0.999	0.358	0.992	0.987	0.549	0.953	**0.854**	**0.692**	**0.782**
Trained on **SVHN**	Plain (TE: **3.53%**)			CEDA (TE: 3.50%)			ACET (TE: 3.52%)		
	MMC	AUROC	FPR@95	MMC	AUROC	FPR@95	MMC	AUROC	FPR@95
SVHN	**0.980**	–	–	0.977	–	–	0.978	–	–
CIFAR-10	0.732	0.938	0.348	0.551	0.960	0.209	**0.435**	**0.973**	**0.140**
CIFAR-100	0.730	0.935	0.350	0.527	0.959	0.205	**0.414**	**0.971**	**0.139**
LSUN CR	0.722	0.945	0.324	0.364	0.984	0.084	**0.148**	**0.997**	**0.012**
Imagenet-	0.725	0.939	0.340	0.574	0.955	0.232	**0.368**	**0.977**	**0.113**
Noise	0.720	0.943	0.325	**0.100**	**1.000**	**0.000**	**0.100**	**1.000**	**0.000**
Adv. Noise	1.000	0.004	1.000	0.946	0.062	0.940	**0.101**	**1.000**	**0.000**
Adv. Samples	1.000	0.004	1.000	0.995	0.009	0.994	**0.369**	**0.778**	**0.279**
Trained on **CIFAR-10**	Plain (TE: 8.87%)			CEDA (TE: 8.87%)			ACET (TE: **8.44%**)		
	MMC	AUROC	FPR@95	MMC	AUROC	FPR@95	MMC	AUROC	FPR@95
CIFAR-10	**0.949**	–	–	0.946	–	–	0.948	–	–
SVHN	0.800	0.850	0.783	0.327	0.978	0.146	**0.263**	**0.981**	**0.118**
CIFAR-100	0.764	0.856	0.715	**0.761**	**0.850**	0.720	0.764	0.852	**0.711**
LSUN CR	0.738	**0.872**	**0.667**	**0.735**	0.864	0.680	0.745	0.858	0.677
Imagenet-	0.757	0.858	0.698	0.749	0.853	0.704	**0.744**	**0.859**	**0.678**
Noise	0.825	0.827	0.818	**0.100**	**1.000**	**0.000**	**0.100**	**1.000**	**0.000**
Adv. Noise	1.000	0.035	1.000	0.985	0.032	0.983	**0.112**	**0.999**	**0.008**
Adv. Samples	1.000	0.034	1.000	1.000	0.014	1.000	**0.633**	**0.512**	**0.590**
Trained on **CIFAR-100**	Plain (TE: **31.97%**)			CEDA (TE: 32.74%)			ACET (TE: 32.24%)		
	MMC	AUROC	FPR@95	MMC	AUROC	FPR@95	MMC	AUROC	FPR@95
CIFAR-100	**0.751**	–	–	0.734	–	–	0.728	–	–
SVHN	0.570	0.710	0.865	0.290	0.874	0.410	**0.234**	**0.912**	**0.345**
CIFAR-10	0.560	0.718	0.856	0.547	0.711	0.855	**0.530**	**0.720**	**0.860**
LSUN CR	0.592	0.690	0.887	0.581	0.678	0.887	**0.554**	**0.698**	**0.881**
Imagenet-	0.531	0.744	0.827	0.504	0.749	0.808	**0.492**	**0.752**	**0.819**
Noise	0.614	0.672	0.928	0.010	1.000	0.000	**0.010**	**1.000**	**0.000**
Adv. Noise	1.000	0.000	1.000	0.985	0.015	0.985	**0.013**	**0.998**	**0.003**
Adv. Samples	0.999	0.010	1.000	0.999	0.012	1.000	**0.863**	**0.267**	**0.975**

Table 1: On the four datasets MNIST, SVHN, CIFAR-10, and CIFAR-100, we train three models: Plain, CEDA and ACET. We evaluate them on out-of-distribution samples (other image datasets, noise, adversarial noise and adversarial samples built from the test set on which was trained). We report test error of all models and show the mean maximum confidence (MMC) on the in- and out-distribution samples (lower is better for out-distribution samples), the AUC of the ROC curve (AUROC) for the discrimination between in- and out-distribution based on confidence value (higher is better), and the FPR at 95% true positive rate for the same problem (lower is better).

the confidence on adversarial examples, see Figure 4 for an illustration, and achieves on MNIST very high AUROC values so that one can detect adversarial examples via thresholding the confidence. Obviously, plain and CEDA fail on this task. The good performance of ACET is to some extent unexpected as we just bias the model towards uniform confidence over all classes far away from the training data, but adversarial examples are still close to the original image. In summary, ACET does improve confidence estimates significantly

	Plain				ACET			
	MNIST	SVHN	CIFAR-10	CIFAR-100	MNIST	SVHN	CIFAR-10	CIFAR-100
Median α	1.5	28.1	8.1	9.9	$> 10^6$	**49.8**	**45.3**	**9.9**
% overconfident	98.7%	99.9%	99.9%	99.8%	**0.0%**	**50.2%**	**3.4%**	**0.0%**

Table 2: We evaluate all trained models on uniform random inputs scaled by a constant $\alpha \geq 1$ (note that the resulting inputs will not constitute valid images anymore, since in most cases they exceed the $[0, 1]^d$ box). We find the minimum α such that the models outputs 99.9% confidence on them, and report the median over 10 000 trials. As predicted by Theorem 3.1 we observe that it is always possible to obtain overconfident predictions just by scaling inputs by some constant α, and for plain models this constant is smaller than for ACET. For MNIST the value of α was so high that we ran into numerical problems. **Second row:** we show the percentage of overconfident predictions (higher than 95% confidence) when projecting back the α-rescaled uniform noise images back to $[0, 1]^d$. One observes that there are much less overconfident predictions for the ACET models compared to standard training.

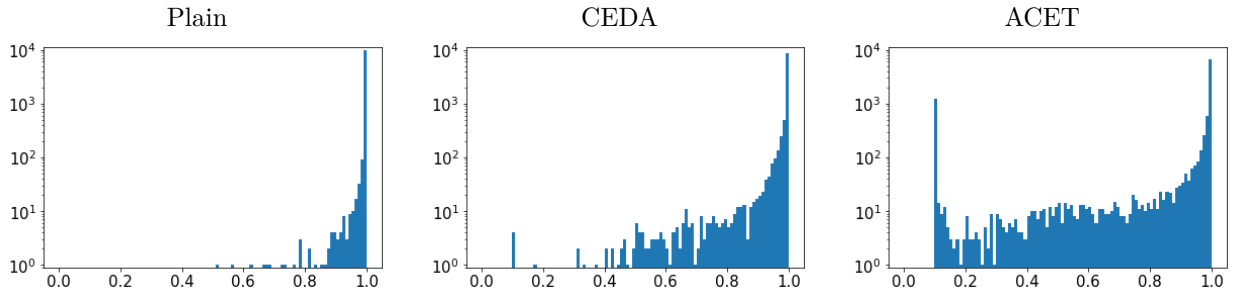

Figure 4: Histogram of confidence values (logarithmic scale) of adversarial samples for MNIST test points. ACET is the only model where most adversarial samples have very low confidence. Note, however that the ACET model has not been trained on adversarial samples of MNIST, but only on adversarial noise.

compared to the plain but also compared to CEDA, in particular on adversarial noise and adversarial examples. In particular, its very good effect also on adversarial examples is an interesting side effect and shows in our opinion that the models have become more reliable.

Far away high confidence predictions: Theorem 3.1 states that ReLU networks always attain high confidence predictions far away from the training data. The two network architectures used in this paper are ReLU networks. It is thus interesting to investigate if the confidence-enhanced training CEDA or ACET makes it harder to reach high confidence than for the plain model. We do the following experiment: we take uniform random noise images x and then search for the smallest α such that the classifier attains 99.9% confidence on αx. This is exactly the construction from Theorem 3.1 and the result can be found in Table 2.

We observe in Table 2 that indeed the required upscaling factor α is significantly higher for CEDA and ACET than for the plain model which implies that our models also influence the network far away from the training data. Moreover, this also shows that even training methods explicitly aiming at counteracting the phenomenon of high confidence predictions far away from the training data, cannot prevent this.

6. Conclusion

We have shown in this paper that the problem of arbitrarily high confidence predictions of ReLU networks far away from the training data cannot be avoided even with modifications like temperature rescaling [12]. It is an inherent problem of the neural network architecture and thus can only be resolved by changing the architecture. On the other hand we have shown that CEDA and in particular ACET are a good way to reach much better confidence estimates for image data. CEDA and ACET can be directly used for any model with little implementation overhead. For the future it would be desirable to have network architectures which have provably the property that far away from the training data the confidence is uniform over the classes: the network knows when it does not know.

Acknowledgements

M.H. and J.B. acknowledge support from the BMBF through the Tübingen AI Center (FKZ: 01IS18039A) and by the DFG TRR 248, project number 389792660 and the DFG Excellence Cluster "Machine Learning - New Perspectives for Science", EXC 2064/1, project number 390727645.

References

[1] R. Arora, A. Basuy, P. Mianjyz, and A. Mukherjee. Understanding deep neural networks with rectified linear unit. In *ICLR*, 2018. 2

[2] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018. 1

[3] P. Bartlett and M. H. Wegkamp. Classification with a reject option using a hinge loss. *JMLR*, 9:1823–1840, 2008. 3

[4] A. Bendale and T. Boult. Towards open set deep networks. In *CVPR*, 2016. 1

[5] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*, 2017. 1

[6] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017. 1

[7] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. Emnist: an extension of mnist to handwritten letters. preprint, arXiv:1702.05373v2, 2017. 5

[8] F. Croce and M. Hein. A randomized gradient-free attack on relu networks. In *GCPR*, 2018. 2

[9] T. DeVries and G. W. Taylor. Learning confidence for out-of-distribution detection in neural networks. preprint, arXiv:1802.04865v1, 2018. 1

[10] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016. 1

[11] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015. 1, 3, 4

[12] C. Guo, G. Pleiss, Y. Sun, and K. Weinberger. On calibration of modern neural networks. In *ICML*, 2017. 1, 8

[13] K. He, X. Zhang, , S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 5

[14] M. Hein and M. Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *NIPS*, 2017. 1

[15] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017. 1, 5

[16] D. Hendrycks, M. Mazeika, and T. Dietterich. Deep anomaly detection with outlier exposure. In *ICLR*, 2019. 1, 4

[17] M. Jordan, U. Lewis, and A. G. Dimakis. Provable certificates for adversarial examples: Fitting a ball in the union of polytopes. *arXiv preprint, arXiv:1903.08778*, 2019. 3

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[19] A. Krizhevsky. Learning multiple layers of features from tiny images. technical report, 2009. 5

[20] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, 2017. 1

[21] M. Lapin, M. Hein, and B. Schiele. Loss functions for top-k error: Analysis and insights. In *CVPR*, 2016. 3

[22] K. Lee, H. Lee, K. Lee, and J. Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *ICLR*, 2018. 1, 5

[23] C. Leibig, V. Allken, M. S. Ayhan, P. Berens, and S. Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific Reports*, 7, 2017. 1

[24] S. Liang, Y. Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR*, 2018. 1, 3, 5

[25] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Valdu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. 1, 2, 4, 5

[26] M. Mirman, T. Gehr, and M. Vechev. Differentiable abstract interpretation for provably robust neural networks. In *ICML*, 2018. 1

[27] G. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *NIPS*, 2014. 2

[28] E. Nalisnick, A. Matsukawa, Y. Whye Teh, D. Gorur, and B. Lakshminarayanan. Do deep generative models know what they don't know? preprint, arXiv:1810.09136v1, 2018. 1, 4

[29] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 5

[30] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, 2015. 1

[31] A. Raghunathan, J. Steinhardt, and P. Liang. Certified defenses against adversarial examples. In *ICLR*, 2018. 1

[32] P. F. S.-M. Moosavi-Dezfooli, A. Fawzi. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, pages 2574–2582, 2016. 1

[33] D. Stutz, M. Hein, and B. Schiele. Disentangling adversarial robustness and generalization. In *CVPR*, 2019. 6

[34] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR*, pages 2503–2511, 2014. 1, 4

[35] A. Tewari and P. Bartlett. On the consistency of multiclass classification methods. *Journal of Machine Learning Research*, 8:1007–1025, 2007. 1

[36] W. Wang, A. Wang, A. Tamar, X. Chen, and P. Abbeel. Safer classification by synthesis. preprint, arXiv:1711.08534v2, 2018. 1

[37] E. Wong and J. Z. Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, 2018. 1

[38] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. preprint, arXiv:1708.07747, 2017. 5

[39] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. preprint, arXiv:1506.03365v3, 2015. 5