PROVABLY ROBUST TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge transfer is an effective tool for learning, especially when labeled data is scarce or when training from scratch is prohibitively costly. The overwhelming majority of transfer learning literature is focused on obtaining *accurate* models, neglecting the issue of adversarial robustness. Yet, robustness is essential, particularly when transferring to safety-critical domains. We analyze and improve the robustness of a popular transfer learning framework consisting of two parts: a feature extractor and a classifier which is re-trained on the target domain. Our experiments show how adversarial training on the source domain affects robustness on source and target domain, and we propose the first provably robust transfer learning models. We obtain strong robustness guarantees by bounding the worst-case change in the extracted features while controlling the Lipschitz constant of the classifier. Our models maintain high accuracy while significantly improving provable robustness.

1 INTRODUCTION

Since their proposal, neural networks are constantly evolving as they are being adapted for many diverse tasks. In general, they have a tendency to become more complex and larger, since e.g. over-paramatrization has proven to be highly beneficial. Training such large and complex neural networks usually requires a huge amount of (labeled) high-quality data. Since this amount of data is not available in all domains, transfer learning was proposed. The idea is to transfer the knowledge of a trained model from the so called source domain to a similar, related task in a target domain for which only a small amount of data exists. Usually, the transfer is considered successful if the model achieves high accuracy on the target domain. However, accuracy is not the only desired property of neural networks. Adversarial robustness is often equally important, especially in safety-critical domains. Most standard (undefended) models, including transfer learning models as we show in Section 4, are not robust to adversarial attacks. Attackers can easily craft deliberate and unnoticeable input perturbations to change the prediction of a correctly classified instance to a different (wrong) class label.

We analyze and improve the robustness of a popular transfer learning framework consisting of two parts: a feature extractor f which extracts representations from the inputs and is trained on the source domain and a classifier h which maps extracted representations to predictions and is retrained on the target domain. So far, the robustness of this popular framework has been analysed only by two studies. Shafahi et al. (2020) show that adversarial training of the feature extractor and using a classifier with one layer improves robustness. Chen et al. (2021) extend the work of Shafahi et al. (2020) by enforcing a fixed Lipschitz-constant on the (multi-layer) classifier f. They propose an adversarial training procedure that minimizes the representation distance between inputs and corresponding adversarial examples. However, they rely solely on PGD-attacks for robustness evaluation. A robustness study based on provable robustness that takes both the feature extractor and the classifier into account, and quantifies how different design choices affect robustness, is still missing.

We fill this gap by analyzing the robustness of the feature extractor under various training procedures and the robustness of the classifier under different constraints. To study the robustness of the classifier proposed by Shafahi et al. (2020), we search for so called adversarial representations – perturbed representations that result in a wrong prediction but are close to the extracted representations and reachable from the input space. If the classifier h is a one-layer neural network (i.e. logistic regression), adversarial representations always exist for all inputs and all models regardless of the training procedure and the constraints. We can find them by solving a simple quadratic optimization problem with linear constraints. That is, we find the closest adversarial representation $\boldsymbol{z}'$ to the unperturbed representation $\boldsymbol{z} = f(\boldsymbol{x})$ such that $h(\boldsymbol{z}') \neq h(\boldsymbol{z})$. Therefore, we further test if the found adversarial representation $\boldsymbol{z}'$ is reachable from the input space. We say $\boldsymbol{z}'$ is *reachable* if we find a input $\boldsymbol{x}''$ that minimizes $||\boldsymbol{z}' - f(\boldsymbol{x}'')||_1$, and results in a wrong prediction, i.e. $h(f(\boldsymbol{x}'')) \neq h(\boldsymbol{z})$[1].

Our analysis shows that enforcing a fixed Lipschitz constant on the classifier in combination with adversarial training reduces the number of reachable adversarial representations and thus improves model robustness. However, even for the most robust model, we still find reachable adversarial representations, especially on the target domain. Thus, combining adversarial training with a fixed Lipschitz constant on the classifier is not enough to make the transfer learning framework robust. To obtain a provably robust framework, we propose to combine randomized smoothing, which allows to bound the worst-case perturbation in the features, with a given Lipschitz constant on the classifier. Since smoothing results in both verifiable and more robust models we obtain a provably robust transfer learning framework.

2 Related Work

Improving neural networks robustness is widely studied for standard tasks such as classification and regression, but there are few works that analyze how robustness properties can be transferred from the source to the target domain. In transfer learning, adversarial training has mainly been used to obtain feature representations that generalize better and to improve model accuracy on unperturbed data of the target domain. Allen-Zhu & Li (2021), Engstrom et al. (2019) and Ilyas et al. (2019) show how adversarial training improves feature representations and results in representations that are more aligned with humans. Utrera et al. (2021) and Salman et al. (2020a) analyze the effect of robust training in transfer learning. While Salman et al. (2020a) shows that robust training improves the accuracy on the unpertubed target domain data, Shafahi et al. (2020) shows the opposite, i.e. that adversarial training on the target domain results in more robust but less accurate target models.

The works by Shafahi et al. (2020) and Chen et al. (2021) are the most closely related to ours. They consider the same transfer learning framework as we do, i.e. models that can be decomposed into a feature extractor f and a classifier h. The feature extractor is trained on the source domain and is then frozen while the classifier h is retrained on the target domain. Shafahi et al. (2020) show that adversarial training of the feature extractor and using a one-layer-classifier improves robustness on the target domain. Chen et al. (2021) proposes an adversarial training procedure that minimizes the distance between adversarial and unperturbed representations (i.e. the output of the feature extractor f). Furthermore, Chen et al. (2021) argue that a one-layer-classifier is not accurate enough on the target domain and thus propose to use a multi-layer classifier with a Lipschitz constant of 1 (enforced using spectral normalization) to improve accuracy and robustness. Neither of these works studies provable robustness as we propose to do.

3 Constrained Transfer Learning models

A simple but popular transfer learning framework consists of two parts: a so called feature extractor f and a classifier h (see Figure 1). The prediction for input $\boldsymbol{x}$ is obtained as $\boldsymbol{x} = h(f(\boldsymbol{x}))$. The feature extractor is trained on the source domain and is then frozen, i.e. not changed during training on the target domain, while the classifier is retrained on the target domain. The idea of this framework is that in related tasks similar features are important, and thus the feature extractor can be transferred from the source domain to the target domain without adaptions, while the classifier maps extracted representations/features to classes and must be adapted to the target domain.

To improve the robustness of this framework we propose to: (i) make the feature extractor f robust on the source domain since it is not changed afterwards, while (ii) simultaneously constraining the classifier h so that retraining on the target domain does not decrease robustness. We achieve (i) via adversarial training and/or randomized smoothing, and (ii) by enforcing a fixed Lipschitz constant on the classifier. Enforcing a given Lipschitz constant on the entire model $f(h(\cdot))$ decreases its adaptability and tends to have a negative effect on the accuracy. However, constraining only the classifier h can sidestep this limitation since mapping learned representations to classes is often

[1]Unlike single-layer classifiers, we cannot compute the input corresponding to an adversarial representation in closed form. We use project gradient descent instead.

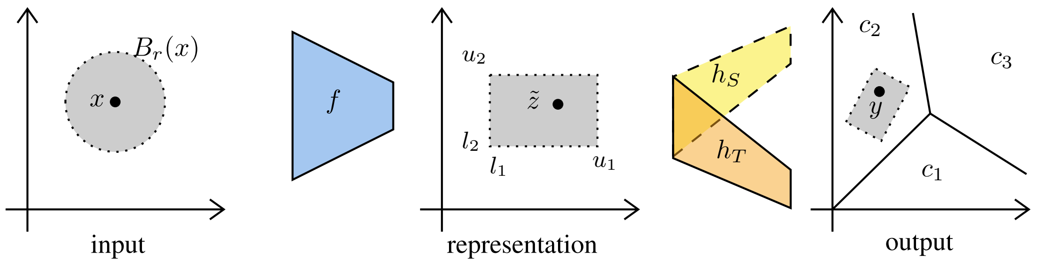

Figure 1: Transfer learning framework consisting of feature extractor f, classifier h_S on the source domain and h_T on the target domain. For input $\boldsymbol{x}$, $f(\boldsymbol{x}) = \boldsymbol{z}$ returns the representations, while $h_S(\boldsymbol{z}) = h_S(f(\boldsymbol{x}))$ and $h_T(\boldsymbol{z}) = h_T(f(\boldsymbol{x}))$ return the predictions on source and target domain respectively. To make transfer learning provably robust, we propose to apply median smoothing on f, and to enforce a Lipschitz constraint on h_S and h_T. Then, given a ball $B_r(\boldsymbol{x})$ of radius r around the clean $\boldsymbol{x}$, we can compute a set that is guaranteed to contain the representations $\boldsymbol{z}' = f(\boldsymbol{x}')$ for all $\boldsymbol{x}' \in B_r(\boldsymbol{x})$ (grey box). After propagating this set through h_T we check if the prediction changes.

easier than directly mapping inputs to classes, especially if the extractor f learns well-separated representations. As our results in Table 1 show, constraining only h leads to comparable or better accuracy on multi-layer classifiers. More importantly, given bounds on the output of the feature extractor f (obtained via randomized smoothing, see Section 5), our constrained h enables us to provide strong robustness guarantees.

Limitations of spectral normalization and orthonormal weight matrices. Chen et al. (2021) recently suggested to enforce a Lipschitz constant of 1 on the classifier by dividing the weights of each layer by their spectral norm (Miyato et al., 2018). Spectral normalization is popular because it is inexpensive and easy to implement with a single step of power iteration. However, since this approach only constrains the largest singular value of the weight matrix to be less than 1 it is not gradient-norm preserving, and is thus suboptimal as shown by Anil et al. (2019). A better alternative is to design a network with orthonormal weight matrices where all singular values equal 1 which preserves the gradient-norm and is easier to train. However, to do so we need to rely on expensive procedures such as Björck Orthonormalization (Cisse et al., 2017; Anil et al., 2019), the Caylee transform (Trockman & Kolter, 2021), the exponential map of a skew-symmetric matrix (Singla & Feizi, 2021), or a block-convolution orthogonal parameterization (Li et al., 2019).

Besides the computational limitations, Singla et al. (2021) argue that using orthonormal weight matrices in the last layer has a negative effect during training. Namely, when updating a single row $\boldsymbol{W}_j$ (corresponding to class j) we necessarily have to update all other rows $\boldsymbol{W}_i$ (corresponding to all other classes) to preserve the orthogonality constraints $\boldsymbol{W}_j \perp \boldsymbol{W}_i$, so an update that learns information about some class may necessitate the forgetting of information relevant for other classes.

Weight normalization based on the $\mathbf{L}_\infty$-norm. Unlike previous work, we propose to normalize the weights using the L_∞-norm. Our proposed weight normalization is computationally inexpensive while providing similar benefits to orthonormal weights – most importantly we can still derive provable robustness guarantees. Using as few constraints as possible preserves adaptability of the classifier h, which is important for retraining and achieving high accuracy on the target domain. Additionally, we combine our weight normalization with the gradient-norm preserving GroupSort activation function proposed by Anil et al. (2019) since it stabilizes gradient-based training of multi-layer Lipschitz-constrained networks and performs better than ReLU in practice.

Guarantees for $\mathbf{L}_\infty$-normalized networks. Let us consider learned representations $\boldsymbol{z} = f(\boldsymbol{x})$ and $\boldsymbol{z}' = f(\boldsymbol{x}')$ corresponding to inputs $\boldsymbol{x}, \boldsymbol{x}'$ and assume that $||\boldsymbol{z}' - \boldsymbol{z}||_\infty \leq \delta$. Let $\boldsymbol{W}^{(i)}$ be the weight matrix in the i-the layer. If $\boldsymbol{W}^{(i)}$ is constrained such that $||\boldsymbol{W}^{(i)}||_\infty \leq k_i$ where $k_i \in \mathbb{R}$ is a given constant for layer i, the distance between $\boldsymbol{z}$ and $\boldsymbol{z}'$ after the linear transformation in the i-th layer is bounded by the Cauchy-Schwarz inequality:

$$||(\boldsymbol{W}^{T(i)}(\boldsymbol{z}' - \boldsymbol{z}))||_\infty \leq ||\boldsymbol{W}^{T(i)}||_\infty ||\boldsymbol{z}' - \boldsymbol{z}||_\infty \leq k_i \delta. \tag{1}$$

Using an activation function with a Lipschitz constant of 1 (such as GroupSort) and considering a classifier with I layers results in the following bound on the outputs $\boldsymbol{y} = h(f(\boldsymbol{x}))$ and $\boldsymbol{y}' = h(f(\boldsymbol{x}'))$:

$$||\boldsymbol{y}' - \boldsymbol{y}||_\infty = ||h(\boldsymbol{z}') - h(\boldsymbol{z})||_\infty \leq \delta \prod_{i=1}^{I} k_i \tag{2}$$

Now, for a given $\boldsymbol{z}$ and $\boldsymbol{z}'$ we can easily verify whether they will obtain the same prediction. Let $s(\boldsymbol{z}) = \max(0, h_i(\boldsymbol{z}) - \max_{j \neq i} h_j(\boldsymbol{z}))$ be the classification margin where $h_j(\boldsymbol{z})$ is the logit for class j and i is the correct label. If $s(\boldsymbol{z}) > 2 \cdot \delta \cdot \prod_{i=1}^{I} k_i$ then $\boldsymbol{z}$ is provably robust (see Anil et al. (2019) and Tsuzuku et al. (2018)). Setting all $k_i = 1$ we obtain a classifier h with a Lipschitz constant of 1 w.r.t. the L_∞ norm.

Transfer benchmarks. Our analysis covers transfer learning tasks of different relatedness and difficulty. SVHN (Netzer et al., 2011) contains images of street view housing numbers, while MNIST (LeCun & Cortes, 2010) contains images of handwritten digits. Using SVHN as source domain and MNIST as target domain is a simple, related task, since the transfer is done from a difficult source domain to a simple target domain. CIFAR10 (Krizhevsky et al., 2009) and STL10 (Adam Coates, 2011) contain images of objects and are related domains. Since CIFAR10 contains low-resolution images (simple domain) and STL10 contains high-resolution images (difficult domain) transferring from CIFAR10 to STL10 is a related, but difficult task. FashionMNIST/FMNIST (Xiao et al., 2017) contains gray-scale images of clothes while KMNIST (Clanuwat et al., 2018) consists of images of Japanese characters. Using FMNIST as source domain and KMNIST as target domain is a distant transfer learning task. Since MNIST and KMNIST are both simple but distant domains, transferring from MNIST to KMNIST is a medium related, simple transfer learning task.

Table 1: Effect of enforcing a Lipschitz constant of 1 on the accuracy on the source domain and target domain. Constrained 4-layer classifiers are comparable or better than unconstrained.

		SVHN $\rightarrow$ MNIST		FMNIST $\rightarrow$ KMNIST		CIFAR10 $\rightarrow$ STL10		MNIST $\rightarrow$ KMNIST	
Constraint	Layers	source	target	source	target	source	target	source	target
None	1	95.5	96.0	92.8	54.1	84.4	64.1	99.8	71.0
WN	1	95.5	90.2	92.7	36.0	85.7	62.7	99.5	50.6
None	4	96.0	97.7	93.4	78.8	83.5	68.0	99.6	87.4
WN	4	95.1	98.2	92.9	77.2	86.2	67.8	99.6	89.6

Table 1 clearly shows two important points. First, (slightly) distant transfer learning tasks such as MNIST $\rightarrow$ KMNIST and FMNIST $\rightarrow$ KMNIST require multi-layer classifiers to achieve high accuracy on the target domain. One-layer classifiers h, as suggested by Shafahi et al. (2020), are not enough with and without weight normalization. They are not as adaptable as necessary and result in lower accuracy on the target domain. Second, models that contain multi-layer classifiers with Lipschitz constant 1 (WN) achieve similar accuracy as unconstrained models (None) regardless of the relatedness between source and target domain. This clearly shows that we need a (constrained) multi-layer classifier h to make the $f(h(\cdot))$ framework robust while maintaining high accuracy.

4 How do Constraints and Adversarial Training affect Robustness on Source and Target domain?

First, we analyze how adversarial training of the feature extractor and enforcing a given Lipschitz constant on the classifier affect the robustness of the transfer learning framework. We compare models with classifiers that have a Lipschitz constant of 1 and unconstrained classifiers in combination with four training procedures. As baseline we use *normal* (standard) training on clean input data with the cross entropy as loss function. The second training procedure, called *noise*, samples a noise vector from a standard normal distribution, clips it such that it is bounded by the L_2-size of ε, and adds it to each input. Training is done on the noisy samples with the cross entropy loss function. This training is also used for models subjected to randomized smoothing based verification (Cohen et al., 2019). In the third training procedure, called *adv*, we compute adversarial examples using PGD-attacks based on the L_2-norm with a perturbation size of ε and the cross entropy loss function. Our fourth training procedure called *adv-d* uses the same setting as the third one, but with a loss function that linearly combines cross entropy with minimizing the distance between original representations $f(\boldsymbol{x})$ and adversarial representations $f(\boldsymbol{x}')$ as proposed by (Chen et al., 2021).

Robustness of one-layer classifiers. Shafahi et al. (2020) claim that using a standard one-layer classifier improves model robustness. Since the weights of the classifier can theoretically become arbitrary large, this is a purely empirical result. We analyze this claim by quantifying the robustness of differently trained one-layer classifiers. Since the classifier maps representations (i.e. the outputs of the feature extractor) to classes, analyzing the robustness of the classifier requires us to operate in the feature/representation space. To this end, we compute the representations $\boldsymbol{z} = f(\boldsymbol{x})$ corresponding to a correctly classifier input $\boldsymbol{x}$ and search for representations $\boldsymbol{z}'$ in the neighborhood of $\boldsymbol{z}$ that result in a wrong prediction. Instead of doing a heuristic search for adversarial representations, we propose to solve this problem exactly. Since we are analyzing one-layer classifiers, we can formulate it as the following optimization problem:

$$\min_{\boldsymbol{z}', j \neq i} ||\boldsymbol{z} - \boldsymbol{z}'||_2 \;\; \text{s.t.} \; (\boldsymbol{W}_i \boldsymbol{z}' + b_i) - (\boldsymbol{W}_j \boldsymbol{z}' + b_j) < 0 \tag{3}$$

where i is the correct class, $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weights and bias of the one-layer classifier, and $\boldsymbol{W}_j$ is the j-th row of $\boldsymbol{W}$. If we assume that $\boldsymbol{z}' = \boldsymbol{z} + \epsilon$ and use $\boldsymbol{z}' - \boldsymbol{z} = (\boldsymbol{z} + \epsilon - \boldsymbol{z}) = \epsilon$ we can simplify the objective function and the constraint, obtaining:

$$\min_{\epsilon, j \neq i} ||\epsilon||_2 \;\; \text{s.t.} \; \boldsymbol{W}_{\Delta_j} \epsilon + \delta_j < 0 \tag{4}$$

where $\boldsymbol{W}_{\Delta_j} = \boldsymbol{W}_i - \boldsymbol{W}_j$ and $\delta_j = (\boldsymbol{W}_i - \boldsymbol{W}_j)z + b_i - b_j$ are constants that we can pre-compute. Since we use the L_2-norm as distance measure, Problem 4 is a quadratic optimization problem with linear constraints and can be solved efficiently.

Table 2 shows different metrics for the adversarial representations, i.e. the percentage of found adversarial representations that result in a wrong prediction (p_z), the mean L_∞ distance z_Δ between $\boldsymbol{z}$ and $\boldsymbol{z}'$. The larger the percentage and the smaller the distance, the less robust is the model. Our representation search shows that adversarial representations can be found for all inputs and models regardless of the constraints and the training procedure. Moreover, the distance between $\boldsymbol{z}$ and $\boldsymbol{z}'$ is relatively small given that we are in a 640-dimensional (theoretically) unbounded space.

The adversarial representation search raises an obvious question: Are such representations reachable from the input space? Using a standard attack (i.e. PGD-attacks) can theoretically return adversarial examples with a small distance in the input space (x_Δ), but a large distance in the feature/representation space. Thus, we search inputs $\boldsymbol{x}''$ that minimize $||\boldsymbol{z}' - f(\boldsymbol{x}'')||_1$ and result in a wrong prediction, i.e. $h(f(\boldsymbol{x}'')) \neq h(\boldsymbol{z})$. Since this problem cannot be solved in closed form, we use project gradient descent with $L_1(\boldsymbol{z}', f(\boldsymbol{x}''))$ as loss function.

Our input search (Table 2, right) shows that weight normalization strongly decreases the percentage of reachable adversarial representations p_x, i.e. representations for which we could find a close, wrongly classified input. Combining weight normalization with any adversarial training procedure further reduces the percentage or reachable adversarial features. For the SVHN $\rightarrow$ MNIST task reachable adversarial representations can be found for about at least one-fifth of the inputs on source and target domain. On the MNIST $\rightarrow$ KMNIST task, the source domain is simple and very robust, but on the target domain reachable adversarial representations are found for at least 39% of the inputs. All adversarial training procedures decrease the percentage of reachable representations and provide adversarial samples with a small distance to the unperturbed input, where standard adversarial training (adv) or training on noisy inputs (noise) results in the largest decrease.

In summary, even for the most robust model, reachable adversarial representations can be found for at least about one-fifth of the inputs on the target domain. Thus, combining a fixed Lipschitz constant with adversarial training is not strong enough to make this transfer learning framework sufficiently robust. Since we cannot rely solely on heuristic defenses, we propose to design a provably robust feature extractor to obtain models that are guaranteed to preserve their robustness on the

Table 2: Computing adversarial representations $\boldsymbol{z}'$ and inputs $\boldsymbol{x}''$ on SVHN/MNIST (source) and MNIST/KMNIST (target). Models: unconstrained (None), with Lipschitz constant 1 (WN). Metrics: model accuracy (Acc.), percentage of found adversarial representations (p_z) or adversarial inputs (p_x), mean L_∞ distance between adversarial representations ($z_\triangle$) and adversarial inputs ($x_\triangle$).

SVHN → MNIST			Rep. Search		Input Search		
Data, Con.	Training	Acc. [%]	p_z [%]	$z_\triangle$	p_x [%]	$x_\triangle$	$z_\triangle$
Source, None	normal	95.5	**100**	1.598	88.2	0.258	1.346
	noise	94.3	**100**	1.543	66.7	0.374	1.243
	adv	96.4	**100**	1.455	**59.1**	0.257	1.138
	adv-d	96.3	**100**	0.375	92.9	0.540	0.440
Source, WN	normal	95.5	**100**	0.046	42.0	0.082	2.034
	noise	93.6	**100**	0.036	**19.1**	0.274	1.538
	adv	96.2	**100**	0.059	21.0	0.102	1.641
	adv-d	96.4	**100**	0.011	22.4	0.461	0.791
Target, None	normal	96.0	**100**	0.152	51.7	0.122	1.454
	noise	95.7	**100**	0.152	33.4	0.265	0.732
	adv	96.8	**100**	0.195	**27.7**	0.180	0.755
	adv-d	95.2	**100**	0.084	40.3	0.504	0.278
Target, WN	normal	90.2	**100**	0.207	45.7	0.127	1.692
	noise	90.8	**100**	0.034	**20.4**	0.247	0.921
	adv	91.0	**100**	0.141	29.6	0.152	1.241
	adv-d	78.9	**100**	0.050	36.3	0.221	0.679
MNIST → KMNIST			Rep. search		Input search		
Data, Con.	Training	Acc. [%]	p_z [%]	$z_\triangle$	p_x [%]	$x_\triangle$	$z_\triangle$
Source, None	normal	99.8	**100**	1.628	**16.9**	0.108	1.637
	noise	99.6	**100**	1.558	23.9	0.812	1.632
	adv	99.5	**100**	1.724	18.2	0.731	1.279
	adv-d	99.6	**100**	0.672	81.8	0.860	0.911
Source, WN	normal	99.5	**100**	0.102	33.5	0.050	2.259
	noise	99.7	**100**	0.185	**0.4**	0.216	0.869
	adv	99.6	**100**	0.105	0.5	0.060	1.013
	adv-d	99.7	**100**	0.001	17.2	0.879	1.163
Target, None	normal	71.0	**100**	0.103	**51.7**	0.285	0.774
	noise	72.3	**100**	0.099	58.5	0.499	0.579
	adv	72.4	**100**	0.106	55.3	0.574	0.538
	adv-d	69.2	**100**	0.070	79.7	0.576	0.493
Target, WN	normal	50.6	**100**	0.171	52.1	0.233	0.991
	noise	51.3	**100**	0.129	47.5	0.345	0.832
	adv	51.3	**100**	0.112	**39.2**	0.461	0.600
	adv-d	44.5	**100**	0.072	64.5	0.594	0.549

target domain. In the following chapter we propose such a framework by proposing *representation smoothing* on the feature extractor f in combination with weight normalization of the classifier h.

5 Provably Robust Transfer Learning Models

As shown in Section 4, making the transfer learning framework robust requires a robust feature extractor as well as a robust classifier. Since heuristic defences are always inevitably broken by new attacks (Athalye et al., 2018; Tramer et al., 2020), we modify the transfer learning framework so that it becomes provably robust. To achieve this goal, we propose to use randomized smoothing. Randomized smoothing was suggested by Cohen et al. (2019) to certify predictions of classifiers and usually results in both verifiable predictions and more robust models. The idea of smoothing based verification techniques is to draw samples $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{x}, \sigma)$ from the close neighborhood of input $\boldsymbol{x}$ and propagate all samples through the neural network. The smooth prediction is computed by aggregating the outputs for all noisy samples. This aggregation can be done by computing the mean or the median (Chiang et al., 2020). We evaluate the performance of label smoothing as proposed by Cohen et al. (2019), and we propose two versions of median smoothing: logit smoothing on the entire $f(h(\cdot))$ model and representation smoothing on the extractor f.

Representation smoothing. The idea of our proposed approach is to smooth the feature extractor f, thus making it robust, without affecting the classifier h. By combining representation smoothing with a Lipschitz-constrained classifier h, we can certify the predictions of the entire $f(h(\cdot))$. Specifically, we draw S random samples $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{x}, \sigma)$, compute the corresponding features $\boldsymbol{z}_i = f(\boldsymbol{x}_i)$ and aggregate them by computing the median representation $\boldsymbol{m} = \text{median}\{\boldsymbol{z}_i\}_{i=1}^{S} \in \mathbb{R}^D$. Here D is the number of dimensions of the representation space ($D = 640$ for the architecture we use). Now, using standard statistical methods (Chiang et al., 2020) we can obtain a guaranteed lower bound $\boldsymbol{l} \in \mathbb{R}^D$ and upper bound $\boldsymbol{u} \in \mathbb{R}^D$ on $\boldsymbol{m}$. To account for the multiple comparisons problem that stems from smoothing in multiple dimensions we apply Bonferroni correction similar to Kumar & Goldstein (2021). The predicted class c is obtained by propagating the representation median through the classifier h and taking the argmax, i.e. $c = \arg\max_k h(\boldsymbol{m})_k$. To verify if the prediction is robust, we use the fact that the classifier h is L_∞-Lipschitz continuous. As explained in Section 3, we compute the maximum deviation between the median and a point $\boldsymbol{p} \in [\boldsymbol{l}, \boldsymbol{u}]$ w.r.t. to L_∞-norm i.e. $\delta = \max(L_\infty(\boldsymbol{u}-\boldsymbol{m}), L_\infty(\boldsymbol{m}-\boldsymbol{l}))$ and the margin $s(\boldsymbol{m}) = \max(0, h_c(\boldsymbol{m}) - \max_{k \neq c} h_k(\boldsymbol{m}))$. If $s(\boldsymbol{m}) > 2 \cdot \delta \cdot k$, where k is the Lipschitz constant of the classifier, then $\boldsymbol{m}$ is provably robust and we return the prediction c, otherwise we abstain.

Logit smoothing. In contrast to representation smoothing, logit smoothing smoothes the whole model, i.e. the feature extractor f and the classifier h. It operates in the logit space and aggregates the outputs by computing the median of the logits. Since we operate in a multi-dimensional space, we have to adapt median smoothing by using the Bonferroni correction (see Kumar & Goldstein (2021) for details). Logit smoothing returns the median logit $\boldsymbol{m} \in \mathbb{R}^K$, a lower bound $\boldsymbol{l} \in \mathbb{R}^K$ and an upper bound $\boldsymbol{u} \in \mathbb{R}^K$ on it (where K is the number of classes). For classifiers, the smooth prediction c is obtained as $c = \arg\max_k m_k$. Since we apply median smoothing for classification, unlike Chiang et al. (2020) that use it for regression, we propose to extend it by an abstain option. To this end, we test if more than one class might be predicted. This can be done by computing the maximum lower bound, i.e. $l_c = \max_k l_k$ and checking if $u_k < l_c, \forall k \neq c$ holds. If this inequality if fulfilled, only one class can be predicted and we return it, otherwise we abstain.

Setup. We compare representation smoothing, logit smoothing, and label smoothing on the source domain and the target domain, for unconstrained classifiers (None) and for classifiers with a Lipschitz constant of 1 enforced by our weight normalization (WN). We train the models using the four procedures (normal, noise, adv, and adv-d) described in the previous section, and we add a fifth training procedure (adv-s), proposed by Salman et al. (2020b) to improve the robustness of smoothed models. In adv-s we explicitly account for the smoothing distribution when computing adversarial examples. Table 3 and Table 4 quantify the verifiable robustness of our framework with a one-layer and a four-layer classifier on SVHN $\rightarrow$ MNIST, MNIST $\rightarrow$ KMNIST, FMNIST $\rightarrow$ KMNIST.

Comparison of smoothing approaches. Representation smoothing results in a significantly higher verifiable accuracy A_V than logit smoothing or label smoothing. On multi-layer classifiers (Table 3 and 4) representation smoothing verifies $2-3\%$ more samples. For SVHN $\rightarrow$ MNIST with a one-layer classifier, representation smoothing verifies about 20% more samples from the source domain and about 30% more target domain samples. On the target domain, representation smoothing is able to verify the majority of correct median predictions, while performance of label and logit smoothing is decreased. The first reason for this might be that weight normalization improves the robustness of the models. Second, representation smoothing aggregates the median in the high-dimensional representation space and computes certificates based on the L_∞-norm, which is small and independent of the number of dimensions. Combining L_∞-norm with aggregation of the median in a high-dimensional space might be beneficial in comparison to further propagation through the neural network and aggregation in a low dimension (logit) space. The Bonferroni correction, which is stricter in the feature space than in the logit space, did not have an observable (negative) effect on verifiable accuracy. Label smoothing and logit smoothing result in the same verifiable accuracy A_V. Since both approaches smooth in the logit space and are based on a highly similar theoretical framework as well as the same parameters, this is expected. Regarding the median accuracy A_M, i.e. using the smooth median without an abstain option to predict, representation smoothing and logit smoothing achieve the same accuracy. One possible explanation is that this happens because the classifier has a Lipschitz constant of 1.

Table 3: Smoothing on SVHN (source) $\rightarrow$ MNIST (target) using classifiers without (None) and with Lipschitz constant 1 (WN). Columns: data (source/target), constraint (None/WN), accuracy of the base classifier (A_B), median prediction accuracy (A_M), verifiable accuracy (A_V) and radius (Rad.)

1-layer classifier		Base	Label smooth.		Representation smooth.		Logit smooth.	
Data, Con.	Training	A_B[%]	A_V[%]	Rad.	A_M[%]	A_V[%]	A_M[%]	A_V[%]
	normal	94.4	26.1	0.049			69.4	26.1
	noise	93.2	79.8	0.153			93.5	79.8
Source	adv	95.7	46.4	0.073			76.9	46.4
None	adv-d	**95.8**	75.4	0.111			84.7	75.4
	adv-s	94.2	**82.1**	0.166			**94.6**	**82.1**
	normal	96.5	19.6	0.055	67.3	56.9	67.3	19.6
	noise	92.9	66.0	0.152	**93.3**	**91.0**	**93.3**	66.0
Source	adv	**96.7**	28.5	0.079	75.7	67.5	75.7	28.5
WN	adv-d	**96.7**	57.9	0.096	83.0	80.4	83.0	57.9
	adv-s	92.2	**72.5**	0.165	92.2	90.6	92.2	**72.5**
	normal	96.2	21.5	0.13			83.9	21.5
	noise	96.2	53.7	0.175			93.4	53.7
Target	adv	**97.3**	38.8	0.155			92.6	38.8
None	adv-d	96.4	**82.0**	0.173			91.7	**82.0**
	adv-s	96.1	69.6	0.192			**95.4**	69.6
	normal	89.6	5.3	0.116	83.1	63.8	83.1	5.3
	noise	**91.3**	26.6	0.168	89.1	82.2	89.1	26.6
Target	adv	89.9	11.1	0.135	83.0	68.6	83.0	11.1
WN	adv-d	77.5	**51.6**	0.148	74.3	69.9	74.3	**51.6**
	adv-s	90.1	36.5	0.179	**89.6**	**83.8**	**89.6**	36.5
4-layer classifier		Base	Label smooth.		Representation smooth.		Logit smooth.	
Data, Con.	Training	A_B[%]	A_V[%]	Rad.	A_M[%]	A_V[%]	A_M[%]	A_V[%]
	normal	95.5	61.6	0.065			70.0	61.6
	noise	93.2	**91.0**	0.155			93.0	**91.0**
Source	adv	95.7	63.7	0.072			80.0	63.7
None	adv-d	**96.7**	78.9	0.075			79.9	78.9
	adv-s	93.2	81.4	0.167			**93.6**	81.4
	normal	95.4	59.7	0.054	69.3	67.2	69.3	59.7
	noise	93.0	89.8	0.153	92.9	92.5	92.9	89.8
Source	adv	95.8	59.2	0.062	66.2	64.9	66.2	59.2
WN	adv-d	**96.8**	78.5	0.096	79.8	79.4	79.8	78.5
	adv-s	92.3	**90.9**	0.165	**93.7**	**92.9**	**93.7**	**90.9**
	normal	97.8	50.6	0.136			88.1	50.6
	noise	97.1	86.6	0.185			96.2	86.6
Target	adv	97.3	58.8	0.15			87.7	58.8
None	adv-d	97.9	89.8	0.17			91.0	89.8
	adv-s	**98.4**	**96.6**	0.201			**98.5**	**96.6**
	normal	98.1	64.8	0.12	80.8	77.8	80.8	64.8
	noise	97.7	90.8	0.18	96.4	95.5	96.4	90.8
Target	adv	98.5	71.8	0.147	89.7	88.1	89.7	71.8
WN	adv-d	**98.6**	85.8	0.168	87.9	87.5	87.9	85.8
	adv-s	97.4	**95.7**	0.199	**97.4**	**97.4**	**97.4**	**95.7**

Comparison of training procedures. The highest median accuracy and verifiable accuracy on the source and the target domain are most often obtained by using adversarial training based on attacks computed on the smooth model (adv-s) as proposed by (Salman et al., 2020b). Training on noisy samples (noise) is the second best training procedure, while standard adversarial training (adv) or including minimization of the representation distance between adversarial examples and unperturbed data in the loss function (adv-d) performs better than standard training (normal).

Comparison of classifiers with and without Lipschitz constant of 1. Models with a classifier of Lipschitz constant 1 (WN, dark background) and unconstrained models (bright background) result in similar verifiable accuracy according to label and logit smoothing. However, since representation smoothing significantly increases the verifiable accuracy (and requires a given Lipschitz constant on the classifier), combining representation smoothing with weight normalization (and adv-s training) results in the most robust models on the target domain as well as on the source domain.

Table 4: Smoothing on MNIST/FMNIST (source) $\rightarrow$ KMNIST (target) using 4-layer classifiers without (None) and with Lipschitz constant 1 (WN). Columns: data (source/target), constraint (None/WN), accuracy of the base classifier (A_B), median prediction accuracy (A_M), verifiable accuracy (A_V) and radius (Rad.).

MNIST $\rightarrow$ KMNIST		Base	Label smooth.		Representation smooth.		Logit smooth.	
Data, Con.	Training	A_B[%]	A_V[%]	Rad.	A_M[%]	A_V[%]	A_M[%]	A_V[%]
	normal	**99.8**	92.6	0.188			93.6	92.6
	noise	99.4	99.4	0.208			99.5	99.4
Source	adv	99.5	79.9	0.174			82.2	79.9
None	adv-d	99.6	97.5	0.202			97.9	97.5
	adv-s	**99.8**	**99.6**	0.208			**99.7**	**99.6**
	normal	99.6	88.3	0.193	89.8	89.2	89.8	88.3
	noise	99.7	99.6	0.208	99.7	99.6	99.7	99.6
Source	adv	99.9	97.6	0.196	98.3	98.2	98.3	97.6
WN	adv-d	99.4	98.5	0.202	99.0	99.0	99.0	98.5
	adv-s	**100.0**	**99.9**	0.209	**99.9**	**99.9**	**99.9**	**99.9**
	normal	86.8	67.4	0.131			77.6	67.4
	noise	88.3	80.4	0.178			88.4	80.4
Target	adv	87.6	60.0	0.127			71.7	60.0
None	adv-d	**92.0**	**87.5**	0.182			**90.3**	**87.5**
	adv-s	87.2	81.1	0.183			87.5	81.1
	normal	88.0	65.5	0.143	80.7	78.3	80.7	65.5
	noise	91.2	82.7	0.187	90.8	89.5	90.8	82.7
Target	adv	87.4	63.1	0.143	75.8	73.5	75.8	63.1
WN	adv-d	**92.3**	**89.6**	0.191	**91.3**	**91.1**	**91.3**	**89.6**
	adv-s	88.6	80.8	0.187	88.0	87.3	88.0	80.8
FMNIST $\rightarrow$ KMNIST		Base	Label smooth.		Representation smooth.		Logit smooth.	
Data, Con.	Training	A_B[%]	A_V[%]	Rad.	A_M[%]	A_V[%]	A_M[%]	A_V[%]
	normal	93.2	69.0	0.108			77.5	69.0
	noise	91.6	89.1	0.176			91.6	89.1
Source	adv	93.3	67.1	0.128			82.6	67.1
None	adv-d	**95.1**	88.5	0.146			88.6	88.5
	adv-s	91.8	**90.1**	0.191			**92.2**	**90.1**
	normal	92.7	62.8	0.092	73.0	70.8	73.0	62.8
	noise	92.7	**91.2**	0.181	**93.0**	**92.2**	**93.0**	**91.2**
Source	adv	93.2	78.4	0.126	83.9	82.5	83.9	78.4
WN	adv-d	**94.1**	90.2	0.168	90.3	90.2	90.3	90.2
	adv-s	91.8	91.0	0.19	91.8	91.4	91.8	91.0
	normal	**80.4**	1.3	0.015			34.8	1.3
	noise	79.4	24.7	0.063			**67.8**	24.7
Target	adv	79.6	2.4	0.022			44.5	2.4
None	adv-d	78.5	**39.9**	0.051			49.5	**39.9**
	adv-s	74.5	23.4	0.072			64.4	23.4
	normal	77.1	0.1	0.016	31.7	17.2	31.7	0.1
	noise	83.6	25.3	0.065	68.0	61.5	68.0	25.3
Target	adv	77.4	2.5	0.039	51.2	35.8	51.2	2.5
WN	adv-d	**86.4**	**56.6**	0.087	66.6	62.9	66.6	**56.6**
	adv-s	80.7	46.0	0.112	**78.9**	**74.4**	**78.9**	46.0

6 Conclusion

This work analyzes and improves the robustness of a popular transfer learning framework consisting of a feature extractor and a classifier which is re-trained on the target domain. Enforcing a given Lipschitz constant on the classifier in combination with adversarial training of the feature extractor decreases the number of reachable adversarial representations. Representation smoothing of a feature extractor, which was trained using smooth adversarial training, in combination with a classifier that has a Lipschitz constant of 1, results in a transfer learning framework that is provably and significantly more robust on the source domain as well as on the target domain.

7 Reproducibility Statement

Detailed information about the data sets, the transfer learning tasks, models, attacks and randomized smoothing is provided in the subsections of section A.1.

References

Andrew Y. Ng Adam Coates, Honglak Lee. An analysis of single layer networks in unsupervised feature learning. *AISTATS*, 2011.

Zeyuan Allen-Zhu and Yuanzhi Li. Feature purification: How adversarial training performs robust deep learning, 2021.

Cem Anil, James Lucas, and Roger Grosse. Sorting out Lipschitz function approximation. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 291–301. PMLR, 09–15 Jun 2019.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pp. 274–283. PMLR, 2018.

Dian Chen, Hongxin Hu, Qian Wang, Yinli Li, Cong Wang, Chao Shen, and Qi Li. Cartl: Cooperative adversarially-robust transfer learning, 2021.

Ping-yeh Chiang, Michael J. Curry, Ahmed Abdelkader, Aounon Kumar, John Dickerson, and Tom Goldstein. Detection as regression: Certified object detection by median smoothing, 2020.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pp. 854–863. PMLR, 2017.

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. *Neural Information Processing Systems, Machine Learning for Creativity and Design Workshop*, 2018.

Jeremy M Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing, 2019.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Adversarial robustness as a prior for learned representations, 2019.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features, 2019.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10. *Canadian Institute for Advanced Research*, 2009.

Aounon Kumar and Tom Goldstein. Center smoothing: Provable robustness for functions with metric-space outputs, 2021.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. *National Institute of Standards and Technology*, 2010.

Qiyang Li, Saminul Haque, Cem Anil, James Lucas, Roger B Grosse, and Jörn-Henrik Jacobsen. Preventing gradient attenuation in lipschitz constrained convolutional networks. *Advances in neural information processing systems*, 32:15390–15402, 2019.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. *Neural Information Processing Systems, Deep Learning and Unsupervised Feature Learning Workshop*, 2011.

Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better?, 2020a.

Hadi Salman, Greg Yang, Jerry Li, Pengchuan Zhang, Huan Zhang, Ilya Razenshteyn, and Sebastien Bubeck. Provably robust deep learning via adversarially trained smoothed classifiers, 2020b.

Ali Shafahi, Parsa Saadatpanah, Chen Zhu, Amin Ghiasi, Christoph Studer, David Jacobs, and Tom Goldstein. Adversarially robust transfer learning, 2020.

Sahil Singla and Soheil Feizi. Skew orthogonal convolutions. *arXiv preprint arXiv:2105.11417*, 2021.

Sahil Singla, Surbhi Singla, and Soheil Feizi. Householder activations for provable robustness against adversarial attacks, 2021.

Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *arXiv preprint arXiv:2002.08347*, 2020.

Asher Trockman and J Zico Kolter. Orthogonalizing convolutional layers with the cayley transform. *arXiv preprint arXiv:2104.07167*, 2021.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks, 2018.

Francisco Utrera, Evan Kravitz, N. Benjamin Erichson, Rajiv Khanna, and Michael W. Mahoney. Adversarially-trained deep nets transfer better: Illustration on image classification, 2021.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *Zalando SE*, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2017.

A APPENDIX

A.1 DETAILS OF THE EXPERIMENTAL SETUP

Data Sets. We use six data sets: The SVHN data set (Netzer et al., 2011) contains images of street view housing numbers, which are transformed to gray-scale during pre-processing. The MNIST data set (LeCun & Cortes, 2010) consists of gray-scale images of handwritten digits. While the CIFAR10 data set (Krizhevsky et al., 2009) contains $3 \times 32 \times 32$ (low-resolution) images, STL10 (Adam Coates, 2011) contains 3×96 (high-resolution) images of objects. Both data sets have 9 overlapping classes (airplane, bird, car, cat, deer, dog, horse, ship, truck) and one different class (frog in CIFAR10, monkey in STL10). The FashionMNIST/FMNIST data set (Xiao et al., 2017) contains gray-scale images of clothes and the KMNIST (Clanuwat et al., 2018) data set consists of gray-scale images of Japanese characters. Each training set is split into in training data (90%) and validation data (10%).

Transfer Learning Tasks Based on the six data sets discussed above we create four transfer learning tasks of different relatedness and difficulty. Usually a transfer learning task is simpler if the source domain is more complex or general compared to the target domain than the other way round. We consider the following transfer tasks (source domain $\rightarrow$ target domain): SVHN $\rightarrow$ MNIST (highly related), CIFAR10 $\rightarrow$ STL10 (related and difficult because of the transfer from a simple/low-resolution to a complex/high-resolution domain), MNIST $\rightarrow$ KMNIST (related) and FMNIST $\rightarrow$ KMNIST (distant).

Models. All models are based on the WideResNet-32-10 architecture (Zagoruyko & Komodakis, 2017) and can be decomposed in a feature extractor and a classifier. The feature extractor consists of 32 convolutional layers and a widening factor of 10. The classifiers consist of one or four fully connected layers with GroupSort activation functions (Anil et al., 2019). Models are implemented in Pytorch and optimized using Adam optimizer and a learning rate of 0.0001. Training is done on GPUs (1 TB SSD) with early stopping by evaluating the validation set loss using a frequency of 2 and a patience of 10 epochs. Weight normalization is implemented such that each row $\boldsymbol{W}_j^{(i)} \in \mathbb{R}^n$ of the weight matrix $\boldsymbol{W}^{(i)}$ is split into directions $\boldsymbol{V}_j^{(i)} \in \mathbb{R}^n$ and magnitude $g_j^{(i)} \in \mathbb{R}$, i.e. $\boldsymbol{W}_j^{(i)} = g_j^{(i)} \boldsymbol{V}_j^{(i)} / ||\boldsymbol{V}_j^{(i)}||$. During training, the parameters are updated and the magnitudes are projected back onto the allowed set restricted by the norm thresholds k_i, i.e. $g_j^{(i)} = \min(g_j^{(i)}, k_i)$. Since $L_2(\boldsymbol{x}) \subseteq L_\infty(\boldsymbol{x})$, this enforces the desired Lipschitz constant w.r.t. L_∞-norm on the weight matrix. We use the L_2 norm of each row, because an update does not clip the largest value to the Lipschitz constant k (and might result in a matrix with $\boldsymbol{W}_{j,k}^{(i)} = k$ for many entries), but preserves the direction of an updated w.r.t. the row during training.

Attacks. We use two different attack types: Noise attacks and Project Gradient Descent (PGD) attacks with attack radii of 0.1. The perturbation is bounded by the L_2-norm and applied to the input after data normalization. For adversarial training we use 10-step PGD attacks, while robustness analysis uses 50-step PGD attacks.

Randomized Smoothing. Randomized smoothing techniques draw samples $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{x}, \sigma)$ from the close neighborhood of input $\boldsymbol{x}$, propagate them through the neural network and aggregate the outputs to obtain a smooth prediction. We use $\sigma = 0.1$, draw 500 samples for each input and bind the probability of returning an incorrect answer/prediction by $\alpha = 10^{-4}$. Since median smoothing for multi-dimensional problems must be adapted to the number of dimensions, we scale such that the overall α remains 10^{-4} as described in Kumar & Goldstein (2021). Smoothing experiments are performed on a balanced, randomly chosen subset of the test set, which consists of 1000 instances.

Adversarial Feature and Input search. Since computing adversarial features $\boldsymbol{z}''$ for a one-layer classifier is a linear optimization problem, we solve it be using Gurobi. For the input search we use project gradient descent (with learning rate 0.05, regularization of 0.1 and at most 1000 steps) and minimize the L_1-distance between adversarial features $\boldsymbol{z}'$ and features $\boldsymbol{z}'' = f(\boldsymbol{x}'')$ corresponding to input $\boldsymbol{x}'$. The adversarial feature search and the input search are performed on a balanced, randomly chosen subset of the test set, which consists of 1000 instances.