Boosting Adversarial Robustness using Feature Level Stochastic Smoothing

Sravanti Addepalli[*], Samyak Jain[*], Gaurang Sriramanan[*], R. Venkatesh Babu
Video Analytics Lab, Department of Computational and Data Sciences
Indian Institute of Science, Bangalore, India

Abstract

Advances in adversarial defenses have led to a significant improvement in the robustness of Deep Neural Networks. However, the robust accuracy of present state-of-the-art defenses is far from the requirements in critical applications such as robotics and autonomous navigation systems. Further, in practical use cases, network prediction alone might not suffice, and assignment of a confidence value for the prediction can prove crucial. In this work, we propose a generic method for introducing stochasticity in the network predictions, and utilize this for smoothing decision boundaries and rejecting low confidence predictions, thereby boosting the robustness on accepted samples. The proposed Feature Level Stochastic Smoothing based classification also results in a boost in robustness without rejection over existing adversarial training methods. Finally, we combine the proposed method with adversarial detection methods, to achieve the benefits of both approaches.

1. Introduction

Deep Neural Networks are susceptible to carefully crafted imperceptible noise known as adversarial attacks [13], which can flip their predictions to completely unrelated classes with high confidence. The catastrophic impact of such attacks has led to significant interest towards building defenses against such attacks.

Adversarial training using Projected Gradient Descent (PGD), proposed by Madry *et al*. in 2018 [20] has been one of the most successful defenses so far. PGD adversarial training coupled with early stopping is still one of the leading defenses [23, 21] indicating that progress on the front of adversarial defenses has been meager since 2018. Schmidt *et al*. [26] show that adversarial training requires significantly more data when compared to standard training. On the CIFAR-10 dataset for example, Carmon *et al*. [7] demonstrate that a 7% increase in robustness requires 500K samples in addition to the 50K training samples of the original dataset. However, it is not practical to assume the availability of $10\times$ more data for training robust models.

Another avenue of research has been towards detecting adversarial samples [34, 24, 16, 12]. Such methods can be used to detect whether a test sample is adversarial or not, allowing the system to abstain from prediction on adversarial samples. However, most of the detection methods rely on identifying specific properties of adversarial images. Hence, they work well under a black-box setting, but fail in the presence of an adaptive adversary, whose goal is to craft an adversarial example which is similar to the distribution of natural images, specifically with respect to the property that is used for detection [31].

Recent work by Stutz *et al*. [30] demonstrates a method of detecting different types of adversaries using Confidence Calibrated Adversarial Training (CCAT). The authors propose to train networks that assign very low confidence to adversarial samples. This method is shown to induce low-confidence predictions to adversaries constrained in other ℓ_p-norm balls as well, and demonstrates remarkably high detection accuracies, while limiting the rejection rate on correctly classified clean samples to 1%. However, this method is overly sensitive to even small random perturbations, which could occur due to factors such as slight soiling on a sensor collecting data. Based on our evaluation on the CIFAR-10 dataset, CCAT rejects 78.60% of images corrupted with Bernoulli noise of magnitude $1/255$, while a model with standard training has an accuracy of 92.4% on the same (Ref: Table-3 of the Supplementary). Thus, the model is becoming more sensitive, while the requirement in general is to make models more robust. Additionally, as shown in Fig.2(a) of the Supplementary, CCAT incorrectly accepts a large fraction (36.5%) of adversarial examples at low perturbation magnitudes ($\delta = 3/255$) against an adaptive attack proposed by the authors [30].

Adversarial detection alone cannot meet the requirements of applications such as robotics and autonomous navigation systems. As an example, at fast driving speeds, a self-driving car abstaining from prediction can be dangerous, as the driver may not be able to take control instantly. Hence we need to limit the frequency at which the system

[*]Equal contribution.
Correspondence to: Sravanti Addepalli (sravantia@iisc.ac.in)

abstains from prediction, while also being able to reject hard images. An ideal system should therefore operate such that weak adversaries which can be correctly classified by an adversarially trained model are not rejected, while strong adversaries which are likely to be misclassified are rejected. Towards this end, we propose a unified framework that combines the merits of adversarial training and detection, while also overcoming the shortcomings of both.

We list the key contributions in our paper here:

- We propose Adversarial training using a stochastic classifier, which enforces one of the feature layers to follow a predefined distribution, facilitating sampling from the same during training and deployment.
- We propose Feature Level Stochastic Smoothing to achieve a boost in adversarial robustness over standard deterministic classifiers.
- We use the proposed stochastic classifier for rejecting low confidence samples, thereby resulting in a significant boost in adversarial robustness.
- We propose metrics to evaluate classifiers which improve robustness using adversarial training, while also incorporating a rejection scheme.
- Finally, we propose a scheme of combining our approach with Confidence-Calibrated Adversarial Training (CCAT) [30], to achieve the merits of both.

The code and pretrained models are available at:
https://github.com/val-iisc/FLSS

2. Related Works

Types of Adversarial Defenses: Amongst the most common methods used to produce robust networks is Adversarial Training, wherein the network is exposed to adversarial images during the training regime. Several other methods relied on introducing randomized or non-differentiable components either in the pre-processing stage or in the network architecture, so as to minimise the effectiveness of generated gradients. However, Athalye *et al.* [2] broke several such defense techniques [4, 19, 11, 33, 27], where it was shown that methods which relied on gradient obfuscation were not truly robust, as gradient masking effects could be successfully circumvented by an adaptive adversary. Though our method uses stochastic elements, the network remains completely differentiable end-to-end through the reparamaterisation trick, akin to that used in Variational Autoencoders [17], and does not rely on gradient obfuscation to achieve robustness. Further, we present a thorough evaluation of our model based on attacks introduced in [2] to verify the same.

Adversarial Training based Defenses: Madry *et al.* [20] proposed training on multi-step adversaries generated using Projected Gradient Descent (PGD), so as to minimise the worst-case loss within the given constraint set. PGD based training continues to be one of the most effective defenses against adversarial attacks known till date. He *et al.* [15] proposed to inject trainable Gaussian noise in the weights of the network as a regularization method while performing standard PGD training. This transforms the weight tensor to a noisy one, wherein the variance of the added Gaussian is parameterized and trainable. Further, Zhang *et al.* [36] introduced a framework to trade-off clean accuracy for adversarial robustness, using a multi-step training method called TRADES. Though TRADES was shown be more effective than PGD training, recent works such as that of Rice *et al.* [23] indicate that high capacity models trained using PGD with early stopping can achieve similar, if not better results. Following this, Pang *et al.* [21] showed that by tuning parameters such as weight decay and the step learning rate schedule, TRADES can achieve better robust accuracy compared to that of Rice *et al.* The current state-of-the-art robust accuracy is achieved using Adversarial Weight Perturbation (AWP) [32] where the loss maximization is done with respect to both input pixels and weight space of the network. Minimizing loss on a proxy network with perturbed weights is shown to result in significantly improved generalization to the test set, owing to the improved flatness of loss landscape. While the AWP formulation can be combined with any defense, AWP-TRADES achieves the state-of-the-art results currently.

Detection of Adversarial Examples: Another avenue of research in this field has been towards addressing the detection of adversarial perturbations. Gosh *et al.* [12] use a VAE with Gaussian Mixture density to perform thresholding based on the distance between the encoding of an input sample and the encoding of the predicted class label, combined with thresholding of the reconstruction error obtained from the decoder. However, this method is not scalable to datasets such as CIFAR-100, with a large number of classes.

Several other detection methods exist which seek to exploit the differences between clean and adversarial samples with respect to a given property. However, such methods are often effective only in black-box settings, and are susceptible to adaptive attacks in a white-box setting. Tramer *et al.* [31] systematically evaluate several defense methods, comprising both adversarial training as well as detection methods. The authors show that with carefully crafted adaptive attacks, several detection methods [24, 34, 16] could be circumvented. The detection method by Roth *et al.* [24] relies on the fact that network outputs of adversarial samples exhibit higher sensitivity to input noise when compared to natural images. The method incorporates thresholding of logits, which inherently assumes that adversarially perturbed samples either have highly confident scores for the incorrect class or lie abnormally close to decision boundaries, depending on the nature of the attack used (such as PGD [20] or CW [6] attacks). This detection method could be compromised using a feature level attack [25], which attempts to generate adversarial examples that exhibit properties of natural images by utilizing a guide image from a different class. We show that our method is robust against this class of feature level adversaries with different loss functions as well (Ref: Table-9 of the Supplementary).

Randomized Smoothing (RS): Cohen *et al.* [8] proposed the addition of Gaussian noise to input images during inference, in order to generate models that are certifiably robust to perturbations that lie within a specified ℓ_2 norm bound. The addition of Gaussian random noise can be used to produce a distribution of network predictions; a p-test on the top two most frequently predicted classes can then be utilised to identify the confidence level of the final averaged prediction. A given image can be rejected if its averaged prediction confidence lies below a pre-set threshold. In this work, we propose to add Gaussian noise in feature space and train the model to produce consistent predictions across multiple samples drawn. We obtain improved performance when compared to randomized smoothing baselines.

3. Preliminaries

In this paper, we consider a stochastic classifier C_θ, that maps an input image x to its corresponding softmax output $C_\theta(x, \epsilon)$ after sampling a noise vector ϵ from a fixed probability distribution such as the Standard Normal distribution $\mathcal{N}(0, I)$ with zero mean, and Identity covariance matrix. The stochastic classifier has two primary components: an Encoder network $E = \{\mu_{\theta_E}, \Sigma_{\theta_E}\}$, and a Multi-layer Perceptron M (consisting of one or more layers), such that:

$$C_\theta(x, \epsilon) = M(\mu_{\theta_E}(x) + \Sigma_{\theta_E}(x) \odot \epsilon)$$

where $\odot$ denotes element-wise multiplication. For an image x, we denote its corresponding ground-truth label as y. For a given sampled noise vector ϵ, we denote the cross-entropy loss for a data sample $\{x, y\}$ as $\ell_{CE}(C_\theta(x, \epsilon), y)$. Further, given a clean image x, we denote an adversarially modified counterpart as $\widetilde{x}$. In this paper, we primarily consider adversaries that are constrained in ℓ_∞ norm of $\delta = 8/255$:

$$\mathcal{A}(x) = \{\widetilde{x} : ||x - \widetilde{x}||_\infty \leq \delta\}$$

4. Proposed Approach

In this section, we discuss the proposed approach in detail. We first discuss the proposed stochastic smoothing based classifier, followed by details on how such classifiers can be used to boost adversarial robustness.

4.1. Feature Level Stochastic Smoothing

Standard classifiers are deterministic and can be defined as a function mapping from input space to output space,

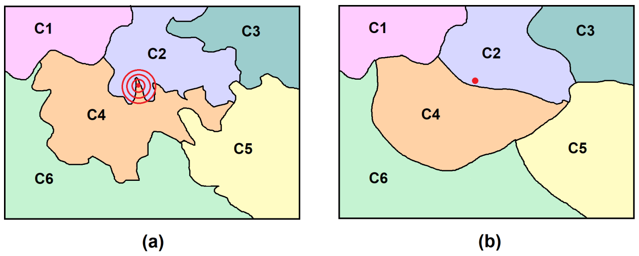

Figure 1. Decision boundaries (in feature space) of (a) Standard Classifier and (b) Feature Level Stochastic Smoothing based Classifier. The data sample which belongs to the class C2 gets incorrectly predicted as C4 in (a), whereas in (b) it is predicted correctly as C2. The smoothed classifier considers a majority vote over samples within the local neighborhood of the image as shown in (a).

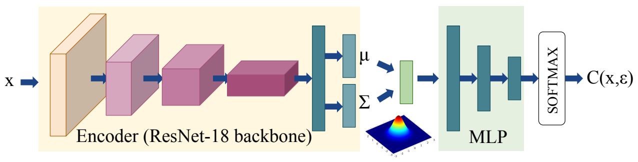

Figure 2. Feature Level Stochastic Smoothing Classifier: The network is trained such that output of the encoder follows a fixed distribution, thereby enabling sampling from this layer to generate a randomized pool of network outputs for a given input x.

leading to a unique output for every possible input in the domain of the function. Deep Networks can be used as deterministic classifiers and are known to achieve very high classification performance. However, training Deep Networks using the standard cross-entropy loss is known to force the network to predict outputs with a high confidence, even for out-of-distribution samples. Several methods [35, 30, 22] have been proposed to overcome this, allowing the confidence predictions of the classifier to be more informative. Some of these methods are also known to generate smoother decision boundaries [35].

In contrast to deterministic classifiers, some classifiers introduce stochasticity during training, typically for their regularizing effect. Randomness during test time is not preferred, as many applications require consistency in predictions while being deployed. This is avoided by taking expectation over the random components during test time either analytically, as done in dropout [29], or by finding a sample mean using a sufficiently large number of samples during test time, as done in randomized smoothing [8]. In this work, we propose Feature Level Stochastic Smoothing based classification, which enforces one of the feature layers to follow a predefined distribution such as the standard normal distribution, thereby facilitating sampling from the same during training and deployment. We use this to achieve the dual objective of smoothing decision boundaries, and for generating multiple randomized predictions for rejection. The diagram in Fig.1 shows that feature space decision boundaries of standard classifiers tend to be highly non-smooth, primarily due to the high dimensionality of the network parameters, leading to poor generalization, specifically to out-of-distribution samples. On the other hand, in the proposed method, we generate predictions by taking an expectation over randomized feature vectors, leading to smoother decision boundaries and better generalization. The expectation is approximated using a sample mean, and consistency among all outputs is used to estimate confidence of the prediction and decide whether the input sample should be rejected.

An implementation of the proposed Feature Level Stochastic Smoothing Classifier is shown in Fig.2. Motivated by the Variational Autoencoder formulation [17], we enforce the encoder network to predict mean (μ_{θ_E}) and variance (Σ_{θ_E}) vectors for each input sample, and use the reparameterization trick for ease of backpropagation. Hence, rather than sampling directly from the mean and variance predicted by the network, we sample from the standard normal distribution and pass $\mu_{\theta_E} + \Sigma_{\theta_E} \odot \epsilon$ to the rest of the network. The sampled feature vector is passed as input to the multi-layer perceptron head for the final classification. Every sampled feature vector gives a different random prediction. We describe the algorithm used for training and inference during test time in the following sections.

4.2. Training Algorithm

The proposed method uses feature level stochastic smoothing discussed in the above section for training robust classifiers. The algorithm for training is shown in Algorithm-1. For ease of notation, we consider a single sample at a time in the algorithm, however the training is done using mini-batches of size 128. In every mini-batch, two steps of training are done. The first step is meant for training on adversarial samples, while the second step uses only clean samples for training. We use adversarial samples constrained within an ℓ_∞ norm of $\delta = 8/255$ for training. An adversarial sample is generated using a 10-step PGD attack [20] and a 11^{th} AWP step [32]. An adversarial sample is initialized by combining an input image with noise sampled from $\mathcal{U}(-\delta, \delta)$, and is passed through the encoder E. A random vector, ϵ is sampled from the Standard Normal distribution and the reparameterized vector, $\mu_{\theta_E} + \Sigma_{\theta_E} \odot \epsilon$ is passed on to the MLP. Cross entropy loss is computed at the output of the network, and is maximized over 10 iterations to find the perturbed image. Following this, a 11^{th} iteration is used to perturb the network weights to maximize the training loss in Eq.1 using AWP [32]. During all 11 iterations, the same initial sampled ϵ value is used in order to ensure that the maximization objective remains consistent throughout. The adversary corresponding to the image x_i is denoted by $\widetilde{x}_i$ and is used in the following loss which is minimized during the first step of training:

$$L = \ell_{CE}(C_\theta(\widetilde{x}_i, \epsilon), y_i) + KL_1 + KL_2 + KL_3 \qquad (1)$$

The first loss term corresponds to the cross-entropy loss on adversarial samples, as utilised in PGD adversarial training [20] to obtain robust models. As shown in Algorithm-1 (L-6), KL_1 denotes the Kullback-Leibler (KL) divergence between the Gaussian distribution $\mathcal{N}(E(x_i))$ corresponding to the clean sample and the Standard Normal distribution. This term is adapted from the Variational Autoencoder [17] setting, and is crucial to enforce the feature representations to follow a known distribution, which in this case is a Standard Normal distribution. The KL_2 term (L-7 in Algo.1) is the KL divergence between the Gaussian distributions corresponding to a pair of adversarial and clean samples. This aids the encoder to learn a smooth function mapping in the ℓ_∞ ball of radius δ around each image, thereby assisting the adversarial training of the network. We use the closed form expression of KL divergence between two Gaussian distributions for the realization of KL_1 and KL_2. We additionally minimize KL_3 (L-8 in Algo.1), which is the KL divergence between the softmax outputs of an adversarial image with and without sampling. This encourages the network to produce consistent predictions across various samples of an adversarial image.

In the second step, the network is trained on clean samples using the following losses:

$$L = \ell_{CE}(C_\theta(x_i, \epsilon'), y_i) + KL_1 + KL_4 \qquad (2)$$

Here, the first term denotes the cross-entropy loss on clean samples, and the second term, KL_1 denotes the objective of enforcing the output of encoder to follow a Standard Normal distribution, as seen earlier. The third term, KL_4 (L-13 in Algo.1), is the KL divergence between the softmax predictions of a clean image and a sampled clean image. This loss is crucial to ensure consistency in predictions across various samples of a given clean image, thereby leading to improved non-rejection of clean samples. We present ablation experiments in Table-2 of the Supplementary, to highlight the significance of each of the loss terms.

4.3. Rejection Scheme

In addition to the goal of improving adversarial robustness, the proposed method also rejects samples which are hard to classify, thereby leading to a boost in the accuracy of accepted samples. In this section, we discuss details on the rejection scheme proposed to be used during deployment of the classifier. Every test sample would be passed through the Encoder network in order to obtain its corresponding mean and variance vectors. At the output of the encoder, N vectors are sampled from the Gaussian corresponding to these mean and variance vectors, and are further propagated through the MLP network to obtain N softmax vectors.

As discussed in Section-4.1, introduction of stochasticity during training is generally coupled with the use of expected value during test time, in order to obtain regularized

Algorithm 1 Adversarial Training using Feature Level Stochastic Smoothing Classifier

```
1: Input: Classifier Network $C_\theta$ (with parameters $\theta$, Encoder $E = \{\mu_{\theta_E}, \Sigma_{\theta_E}\}$, MLP $M$), Training Data $\{x_i, y_i\}_{i=1}^K$, Epochs $T$, Learning Rate $\eta$, Adversarial Perturbation function $A(C_\theta(x,\epsilon), y)$
2: for $epoch = 1$ to $T$ do
3:     for $i = 1$ to $K$ do
4:         Sample $\epsilon \sim \mathcal{N}(0, I)$
5:         $\widetilde{x}_i = A(C_\theta(x_i,\epsilon), y_i)$
6:         $KL_1 = KL\big(\mathcal{N}(E(x_i))||\mathcal{N}(0,I)\big)$
7:         $KL_2 = KL\big(\mathcal{N}(E(\widetilde{x}_i))||\mathcal{N}(E(x_i))\big)$
8:         $KL_3 = KL\big(C_\theta(\widetilde{x}_i,\epsilon)||C_\theta(\widetilde{x}_i,0)\big)$
9:         $L = \ell_{CE}(C_\theta(\widetilde{x}_i,\epsilon), y_i) + KL_1 + KL_2 + KL_3$
10:        $\theta = \theta - \eta \cdot \nabla_\theta L$
11:        Sample $\epsilon' \sim \mathcal{N}(0, I)$
12:        $KL_1 = KL\big(\mathcal{N}(E(x_i))||\mathcal{N}(0,I)\big)$
13:        $KL_4 = KL(C_\theta(x_i,\epsilon')||C_\theta(x_i,0))$
14:        $L = \ell_{CE}(C_\theta(x_i,\epsilon'), y_i) + KL_1 + KL_4$
15:        $\theta = \theta - \eta \cdot \nabla_\theta L$
16:    end for
17: end for
```

and deterministic outputs. In the absence of an analytical expression for the expectation, a sample mean over all N probability vectors can be used. It is to be noted that, as N approaches infinity, the sample mean would approach the expected value. Thus, higher values of N lead to much better estimates of the network output as shown in Fig.1(a) in the Supplementary, while they add to the test time complexity. We choose N to be 100 for our experiments. Since the N additional forward propagations are done only on the MLP, the test time overhead is insignificant for $N = 100$. We note that the increase is computational cost is 2% with 100 times sampling when parallelized on Nvidia-2080Ti.

In the proposed rejection scheme, we first find the class predictions for each of the N sampled outputs. Further, we define the class with maximum number of predictions (majority vote class) to be the class predicted by the *Smoothed Classifier*. We set rejection threshold based on frequency of the predicted class, which serves as a proxy to the confidence of prediction. If the frequency of the predicted class is below a predefined threshold f, the classifier rejects the sample, otherwise it returns the most frequent class as its prediction. We empirically find that the majority vote based rejection scheme leads to better improvements in robust accuracy, when compared to a rejection scheme based on finding sample mean across softmax predictions.

We discuss the important metrics for our proposed classifier along with the method used for selecting threshold in the following section.

4.4. Evaluation Metrics

The commonly used evaluation metrics for adversarially trained classifiers are accuracy on clean (or natural) samples, Acc_{nat} and accuracy on adversarial samples, Acc_{adv}. For methods which detect adversarial samples, the important metrics include True Positive Rate (TPR) and False Positive Rate (FPR), where the case of rejection is set to be the positive class. While it is important to ensure that all adversaries are detected (TPR), it is also necessary to limit the number of clean samples which are incorrectly predicted as adversarial, and hence rejected (FPR). The proposed classifier combines both adversarial training as well as detection, and therefore requires novel metrics which can better measure the effectiveness of the method.

Selection of threshold for rejection: We select the threshold for rejection such that not more than 10% of the clean samples are correctly classified and rejected [30]. It is to be noted that this metric is independent of the clean accuracy of the classifier, and hence the number of correctly classified clean samples that are allowed to be rejected are the same across all baselines for a given dataset. In practice, a hold-out validation set can be used for finding this threshold. However, in order to strictly ensure a fair comparison between baselines and the proposed method, we use the test set to find the threshold.

Metrics used for evaluation: We explain the terminology and metrics used for our evaluations here. We denote the accuracy on natural samples and adversarial samples in the *No Sampling* (NS) case by $Acc_{nat,NS}$ and $Acc_{adv,NS}$ respectively. In the proposed classifier, this metric is calculated by passing the mean vector from the encoder output directly to the MLP, without considering the variance. $Acc_{nat,0\%}$ and $Acc_{adv,0\%}$ denote the accuracy without rejection (but with sampling) on natural samples and adversarial samples respectively, while $Acc_{nat,10\%}$ and $Acc_{adv,10\%}$ denote the same with the rejection threshold set to 10%. For defining $Acc_{adv,10\%}$, we consider the worst case attack for every data sample as recommended by Carlini *et al*. [5]. The calculation of this metric along with other important metrics is described below.

Worst case robustness evaluation with rejection: For a data sample $\{x_i, y_i\}$, we denote the predicted label using any given classifier C by $C(x_i)$ and the decision of the rejection scheme (detector) by $D(x_i)$. $D(x_i) = 1$ denotes the case where the sample is rejected, while $D(x_i) = 0$ means that the sample is accepted by the classifier for prediction.

We denote the set of all perturbations of a data sample x_i, within the threat model defined in Section-3, by $\mathcal{A}(x_i)$.

We define S_{FC} (FC: Flag Correct) to be the set of all images which are not rejected by any adversary, and are predicted correctly by the classifier as shown below:

$$S_{FC} = \{i : D(\widetilde{x}_i) = 0\ , C(\widetilde{x}_i) = y_i\ \ \forall\, \widetilde{x}_i \in \mathcal{A}(x_i)\} \quad (3)$$

S_{FW} (FW: Flag Wrong) is defined as the set of all accepted images, incorrectly predicted for at least one attack:

$$S_{FW} = \{i : \exists\, \widetilde{x}_i \in \mathcal{A}(x_i) : D(\widetilde{x}_i) = 0, C(\widetilde{x}_i) \neq y_i\} \quad (4)$$

S_{FC} is computed in practice by obtaining the indices of correctly classified accepted samples for each attack, and finding an intersection of all such sets. Similarly, S_{FW} is computed by finding the indices of incorrectly classified samples which are accepted for each attack, and subsequently finding a union across all such sets.

$$S_{FC} = \bigcap_i S_{FC,attack_i}\ , \quad S_{FW} = \bigcup_i S_{FW,attack_i} \qquad (5)$$

We define the metrics FC and FW as the percentage of images that belong to the sets S_{FC} and S_{FW} respectively as follows, where $\mathcal{X}$ denotes the test set:

$$FC = \frac{|S_{FC}|}{|\mathcal{X}|} \cdot 100\ , \quad FW = \frac{|S_{FW}|}{|\mathcal{X}|} \cdot 100 \qquad (6)$$

We define the worst case robust accuracy on accepted samples to be the fraction of samples which are always accepted and correctly classified, under any possible adversarial attack. This fraction is defined on the set of images which are either always correctly classified, or are incorrectly predicted by at least a single attack.

$$Acc_{adv,10\%} = \frac{|S_{FC}|}{|S_{FC}| + |S_{FW}|} \cdot 100 \qquad (7)$$

We define R to be the set of all images which can be rejected using at least a single attack. MPR (Maximum Percentage Rejection) is defined as the maximum percentage of samples that can be rejected by the model.

$$R = \{i : \exists\ \widetilde{x}_i \in \mathcal{A}(x_i) : D(\widetilde{x}_i) = 1\},\ MPR = \frac{|R|}{|\mathcal{X}|} \cdot 100 \qquad (8)$$

For adversarial detection methods, MPR would be equal to 100% in the ideal case, as the goal of such methods is to reject all adversarial samples. This makes them susceptible to adversaries who aim to get all input images rejected.

5. Experiments and Results

We report results on the standard benchmark datasets, CIFAR-10 and CIFAR-100 [18]. We use ResNet-18 architecture [14] for all the experiments. We report all results with an ℓ_∞ constraint of $8/255$. We present more details on datasets, training settings and results in the Supplementary.

5.1. Baselines

In this paper, we approach the problem of improving the adversarial robustness of Deep Networks and rejecting low confidence samples in parallel. There is no prior work which we can directly compare our results with. Stutz *et al.* [30] show that confidence thresholding of state-of-the-art models such as PGD [20] and TRADES [36] give the best results for detection as well, when limited to a well-defined threat model. Therefore, PGD, TRADES and AWP [32] trained models can be used to achieve the combined goal of improving adversarial robustness, and also rejecting low confidence samples. We use these models combined with confidence thresholding as our primary baselines. If the confidence of the predicted class is lower than a predefined threshold, the samples are rejected, otherwise a classification output is predicted. The rejection threshold is set based on the same criteria as that defined in Section-4.4.

In addition to the baselines discussed above, we also consider the baselines PGD (Noise) and TRADES (Noise). These baselines are same as PGD or TRADES during training, however they differ during test time. For every test image, we generate 100 noise images by sampling each pixel from $\mathcal{U}[-32/255, 32/255]$ distribution. Each of these attack images is added to the test image, to generate 100 samples of the test image. These 100 samples are passed through the network to generate 100 softmax vectors. We implement the same rejection scheme that is used in the proposed approach for generating the predictions and for rejecting low confidence samples. For this baseline, reducing noise results in a very low rejection percentage, whereas increasing noise reduces the accuracy on clean samples significantly. We add noise only during inference (and not training) since we empirically find that training such models with Gaussian Noise augmentations results in degraded performance.

We consider the baseline of Randomized smoothing (RS) [8], where we train the model as described by the authors. We also consider a baseline which combines Randomized Smoothing with the TRADES defense, as proposed by Blum *et al.* [3], and reject images if the most frequently predicted class is less than a pre-defined threshold value. For the baseline using Parametric Noise Injection (PNI) [15], we utilize the stochasticity of the model to find a threshold corresponding to the criteria of rejecting 10% correctly classified natural images.

We further compare our method with the work by Stutz *et al.* [30], although they primarily prove robustness to unseen threat models, while in this work, we consider the robustness within a well defined threat model.

5.2. Attacks considered for Evaluation

The evaluations are done to predict the worst case accuracy across an ensemble of attacks. Across all evaluations (unless specified otherwise), we set the rejection threshold such that not more than 10% of the clean samples are correctly classified and rejected. The metrics discussed in Section-4.4 are reported.

Table 1. **White-Box Evaluation**: Performance (%) of models under an ensemble of 6 attacks : PGD, APGD-CE, APGD-DLR [10], PGD-CW [6], GAMA-PGD and GAMA-MT [28]. FC denotes the % of samples which are correctly classified and accepted for all attacks. FW denotes the % of samples which are accepted and incorrectly classified by at least one attack. MPR denotes the max % rejected samples.

Method	Thresholding	$Acc_{nat,NS}$ ↑	$Acc_{nat,0\%}$ ↑	$Acc_{nat,10\%}$ ↑	$Acc_{adv,NS}$ ↑	$Acc_{adv,0\%}$ ↑	$Acc_{adv,10\%}$ ↑	FC ↑	FW ↓	MPR
CIFAR-10										
PGD-AT [20, 23, 21]	Confidence	83.80	83.80	91.93	49.07	49.07	51.15	43.99	42.00	44.19
TRADES [36]	Confidence	81.77	81.77	90.26	49.43	49.43	51.85	**44.13**	40.97	44.34
AWP [32]	Confidence	80.58	80.58	89.14	49.80	49.80	53.01	44.06	39.05	44.01
CCAT [30]	Confidence	**89.92**	**89.92**	**97.52**	0.00	0.00	0.00	0.00	**8.52**	100.00
FLSS (**Ours**) (SD=1)	Confidence	80.51	80.51	89.10	**50.64**	50.64	54.06	42.28	35.84	42.18
FLSS (**Ours**) (SD=2)	Maj. Vote	80.51	77.68	89.63	**50.64**	**51.00**	**56.16**	43.16	33.69	47.42
CCAT [30] + FLSS (**Ours**)	Conf + Maj. Vote	80.51	-	89.10	**50.64**	-	**56.16**	43.16	33.69	47.42
CIFAR-100										
PGD-AT [20, 23, 21]	Confidence	56.13	56.13	74.30	25.40	25.40	26.06	23.30	66.08	59.08
TRADES [36]	Confidence	57.84	57.84	74.09	24.33	24.33	23.84	22.70	72.49	57.12
AWP [32]	Confidence	**58.21**	**58.21**	74.31	25.16	25.16	25.07	23.43	70.01	55.77
FLSS (**Ours**) (SD=1)	Confidence	51.86	51.86	70.86	**25.57**	**25.57**	28.79	**23.51**	58.14	56.48
FLSS (**Ours**) (SD=2)	Maj. Vote	51.86	47.50	**74.35**	**25.57**	**25.57**	**29.95**	22.01	**51.46**	65.61

Table-1 in the Supplementary shows results of the proposed approach and baseline models against an ensemble of five attacks: Projected Gradient Descent (PGD) with fixed step size, AutoPGD with Cross-Entropy loss (APGD-CE) and Difference of Logits Ratio Loss (APGD-DLR) [10], Fast Adaptive Boundary Attack (FAB) [9] and Square Attack [1]. The first four comprise of some of the strongest known white-box attacks, while Square attack is a query based black-box attack. We use 100 steps each for PGD, APGD-CE, APGD-DLR and FAB attack, and use 5000 queries for the Square attack. We observe that the proposed method achieves significantly better adversarial performance across different metrics. Based on these results, we select the following strong baselines for evaluations in the main paper: PGD, TRADES and AWP with confidence thresholding, and CCAT.

For our main evaluations in Table-1, we use the following ensemble of 100-step attacks, which were able to reliably estimate the worst-case performance of networks before and after rejection, at a reasonable computational budget: PGD [20], APGD-CE, APGD-DLR [10], PGD with CW loss [6], GAMA-PGD and GAMA-MT [28]. While AutoAttack [10] is strong enough to estimate robustness before rejection, we find that Maximum-Margin based attacks such as PGD with CW loss, GAMA-PGD and GAMA-MT are significantly better at estimating the true robustness after rejection, possibly because rejection implicitly relies on the confidence-margin of predictions.

5.3. Results

Test-time prediction using the proposed method (FLSS) involves sampling 100 times from the latent space, after which the majority vote class is predicted. While we use a Standard Normal distribution $\mathcal{N}(0, I)$ for sampling during training, the use of different scaling factors for standard deviation at test time can result in different robustness-accuracy trade-offs. For example, as shown in the CIFAR-10 results in Table-1, we achieve 2.1% higher robust accuracy after rejection ($Acc_{adv,10\%}$) using standard deviation scaling of 2 ($SD = 2$), when compared to 1. While the clean accuracy without rejection ($Acc_{nat,0\%}$) for $SD = 2$ is 2.83% lower than the case with $SD = 1$, we achieve higher natural accuracy after rejection ($Acc_{nat,10\%}$) at $SD = 2$, which is the main metric to consider. Therefore, we achieve better clean accuracy ($Acc_{nat,10\%}$) and adversarial robustness ($Acc_{adv,10\%}$) after rejection using $SD = 2$. We empirically find that increasing the scaling factor does not improve performance further. Hence, we consider $SD = 2$ as our primary approach across all datasets.

Overall, the important evaluation metrics to consider are natural accuracy ($Acc_{nat,10\%}$) and robust accuracy ($Acc_{adv,10\%}$) after rejection. We note that the proposed method FLSS ($SD = 2$) achieves a significantly higher robust accuracy after rejection ($Acc_{adv,10\%}$) when compared to the baselines across both datasets, at a comparable value of clean accuracy after rejection ($Acc_{nat,10\%}$). While the clean accuracy of CCAT [30] is exceptionally high (97.52%), the results of CCAT cannot be directly compared with ours, as CCAT is an algorithm for detection of adversaries. We discuss CCAT in detail in Section-5.4.

For CIFAR-10, we obtain an improvement of 3.15% on the $Acc_{adv,10\%}$ metric over the strongest baseline (AWP [32]). For CIFAR-100 we achieve an improvement of 3.89% over PGD-AT [20, 21, 23], which is the strongest baseline. We also obtain the best Clean Accuracy after rejection ($Acc_{nat,10\%}$) when compared to all baselines on CIFAR-100.

5.4. Combining FLSS with CCAT

As discussed in Section-1, although CCAT [30] achieves remarkably high detection accuracy while limiting the rejection percentage on correctly classified clean samples to

1%, it is overly sensitive to adversarial examples and random perturbations at low magnitudes. In some cases it achieves a very high rejection rate on easy samples, and in others it causes high misclassification on accepted samples. While the proposed approach (FLSS) achieves a significantly higher accuracy on adversarial samples after rejection ($Acc_{adv,10\%}$), we limit the rejection percentage of correctly classified clean images to 10%, which is much higher than that considered in CCAT (1%). Although Table-1 reports CCAT results with a threshold of 10%, the authors achieve the claimed results with a threshold of 1%.

We propose to achieve the merits of CCAT and FLSS (Ours) by using a combination of both models during test-time. The test samples would first be evaluated using CCAT to obtain a decision of Accept or Reject. Samples accepted by CCAT would be evaluated using FLSS without sampling in latent space, while samples rejected by CCAT would be evaluated using FLSS with a threshold corresponding to the 10% rejection criteria. Therefore, samples accepted by CCAT cannot be rejected by FLSS, ensuring that clean and correctly classified samples have a rejection threshold of 1% similar to CCAT. Adversarial examples which are rejected by CCAT would have a higher accuracy, since they are predicted or rejected using FLSS which is adversarially trained. Images corrupted with low magnitude random noise, which are rejected by CCAT would be accurately predicted by FLSS, thereby improving the sensitivity of the overall system against random noise.

The plots in Fig.2 of the Supplementary show results against the adaptive attack considered by Stutz *et al.* [30], which is a variant of maximum-margin loss coupled with momentum and backtracking. Against CCAT, this attack causes a high FW (accepted and incorrectly predicted images) in the range of $\delta \in [2/255, 5/255]$, despite having a very high Rejection rate (R). In fact, rejection causes Robust Accuracy in CCAT to reduce, indicating that misclassified samples are not being rejected. Such low magnitude adversarial examples can be reliably predicted using FLSS, since it is adversarially trained. This results in a significant reduction in Rejection rate and boost in $Acc_{adv,10\%}$ when CCAT is combined with FLSS.

The results of the combined method (CCAT + FLSS) are presented for CIFAR-10 dataset in Table-1. The clean accuracy corresponds to the No-Sampling case, while the adversarial accuracy is similar to FLSS ($SD = 2$). The pre-trained model available on the CCAT GitHub repository has been used for CIFAR-10. We do not report CCAT results on CIFAR-100 since the pre-trained model is not available. Therefore, the method that combines FLSS with CCAT achieves the merits of both. We achieve the results similar to FLSS at a threshold corresponding to 1% rejection of clean samples which are correctly predicted.

5.5. Evaluation against EOT attack

Since the proposed method utilizes randomization during inference, we evaluate its performance against Expectation over Transformation (EOT) attacks [2] (Table-4 in the Supplementary). We forward propagate each input image k times and use the average gradient direction to find the adversary. This is needed only for defenses which include randomization, and hence is not required for the baselines. We find that EOT attack is not stronger than the standard PGD attack, indicating that the gradients in the standard attacks are reliable to produce sufficiently strong attacks.

5.6. Checks to ensure absence of Gradient Masking

We evaluate the model against stronger multi-step attacks and attacks with multiple random restarts and observe that the model trained using the proposed approach is stable to attacks with 1000 steps and 10 random restarts (Tables-7 and 8 in the Supplementary). Further, as suggested by Athalye *et al.* [2], we perform the standard sanity checks and find that the proposed approach does not show gradient masking. We show evaluations of our model against the gradient-free attack, Square, in Tables-5 and 6 of the Supplementary. We note that gradient based attacks are stronger, thereby confirming the absence of gradient masking. We also evaluate our model against transfer-based black-box attacks (Tables-5 and 6 in the Supplementary) and note that they are significantly weaker than the other white-box attacks.

6. Conclusions

In this work, we seek to combine the desirable characteristics of adversarial training methods with detection techniques, while overcoming the deficiencies of both approaches. We propose adversarial training with Feature Level Stochastic Smoothing, wherein we utilise random smoothing of latent space features to obtain effectively smoother decision boundaries. Further, by enforcing the latent space to follow a fixed probability distribution during the training regime, we generate multiple predictions during inference, and subsequently assign a confidence score to each input sample based on consistency of predictions. By rejecting low confidence adversaries, we obtain a significant boost in performance on accepted samples over a wide range of attacks. Further, we demonstrate that by combining a popular adversarial detection method CCAT with the proposed approach, we achieve the merits of both methods.

7. Acknowledgements

This work was supported by Uchhatar Avishkar Yojana (UAY) project (IISC_10), MHRD, Govt. of India.

References

[1] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. *arXiv preprint arXiv:1912.00049*, 2019.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[3] Avrim Blum, Travis Dick, Naren Manoj, and Hongyang Zhang. Random smoothing might be unable to certify ℓ_∞ robustness for high-dimensional images. *arXiv preprint arXiv:2002.03517*, 2020.

[4] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018.

[5] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, and Aleksander Madry. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[6] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[7] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems*, pages 11192–11203, 2019.

[8] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

[9] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. *arXiv preprint arXiv:1907.02044*, 2019.

[10] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv preprint arXiv:2003.01690*, 2020.

[11] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, and Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. In *International Conference on Learning Representations*, 2018.

[12] Partha Ghosh, Arpan Losalka, and Michael J Black. Resisting adversarial attacks using gaussian mixture variational autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 541–548, 2019.

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Zhezhi He, Adnan Siraj Rakin, and Deliang Fan. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 588–597, 2019.

[16] Shengyuan Hu, Tao Yu, Chuan Guo, Wei-Lun Chao, and Kilian Q Weinberger. A new defense against adversarial images: Turning a weakness into a strength. In *Advances in Neural Information Processing Systems*, pages 1635–1646, 2019.

[17] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[18] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[19] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Michael E. Houle, Dawn Song, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *International Conference on Learning Representations*, 2018.

[20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Tsipras Dimitris, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[21] Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. *arXiv preprint arXiv:2010.00467*, 2020.

[22] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[23] Leslie Rice, Eric Wong, and J Zico Kolter. Overfitting in adversarially robust deep learning. *arXiv preprint arXiv:2002.11569*, 2020.

[24] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. The odds are odd: A statistical test for detecting adversarial examples. *arXiv preprint arXiv:1902.04818*, 2019.

[25] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J Fleet. Adversarial manipulation of deep representations. *arXiv preprint arXiv:1511.05122*, 2015.

[26] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pages 5014–5026, 2018.

[27] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*, 2018.

[28] Gaurang Sriramanan, Sravanti Addepalli, Arya Baburaj, and R Venkatesh Babu. Guided Adversarial Attack for Evaluating and Enhancing Adversarial Defenses. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[29] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[30] David Stutz, Matthias Hein, and Bernt Schiele. Confidence-calibrated adversarial training: Generalizing to unseen attacks. In *Proceedings of the International Conference on Machine Learning*, 2020.

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

[32] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33, 2020.

[33] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*, 2018.

[34] Xuwang Yin, Soheil Kolouri, and Gustavo K Rohde. Adversarial example detection and classification with asymmetrical adversarial training. *arXiv preprint arXiv:1905.11475*, 2019.

[35] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[36] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019.