REGroup: Rank-aggregating Ensemble of Generative Classifiers for Robust Predictions

Lokender Tiwari$^{1,2}$ Anish Madan$^1$ Saket Anand$^1$ Subhashis Banerjee$^{3,4}$

$^1$IIIT-Delhi  $^2$TCS Research  $^3$IIT Delhi  $^4$Department of Computer Science, Ashoka University

https://lokender.github.io/REGroup.html

Abstract

Deep Neural Networks (DNNs) are often criticized for being susceptible to adversarial attacks. Most successful defense strategies adopt adversarial training or random input transformations that typically require retraining or fine-tuning the model to achieve reasonable performance. In this work, our investigations of intermediate representations of a pre-trained DNN lead to an interesting discovery pointing to intrinsic robustness to adversarial attacks. We find that we can learn a generative classifier by statistically characterizing the neural response of an intermediate layer to clean training samples. The predictions of multiple such intermediate-layer based classifiers, when aggregated, show unexpected robustness to adversarial attacks. Specifically, we devise an ensemble of these generative classifiers that rank-aggregates their predictions via a Borda count-based consensus. Our proposed approach uses a subset of the clean training data and a pre-trained model, and yet is agnostic to network architectures or the adversarial attack generation method. We show extensive experiments to establish that our defense strategy achieves state-of-the-art performance on the ImageNet validation set.

1. Introduction

Deep Neural Networks (DNNs) have shown outstanding performance on many computer vision tasks such as image classification [27], speech recognition [22], and video classification [26]. Despite showing superhuman capabilities in the image classification task [21], the existence of adversarial examples [48] have raised questions on the reliability of neural network solutions for safety-critical applications.

Adversarial examples are carefully manipulated adaptations of an input, generated with the intent to fool a classifier into misclassifying them. One of the reasons for the attention that adversarial examples garnered is the ease with which they can be generated for a given model by simply maximizing the corresponding loss function. This is achieved by simply using a gradient based approach that finds a small perturbation at the input which leads to a large change in the output [48]. This apparent instability in neural networks is most pronounced for deep networks that have an accumulation effect over the layers. This effect results in taking the small, additive, adversarial noise at the input and amplifying it to substantially noisy feature maps at intermediate layers that eventually influences the softmax probabilities enough to misclassify the perturbed input sample. This observation of amplification of input noise over the layers is not new, and has been pointed out in the past [48]. The recent work by [57] addresses this issue by introducing feature denoising blocks in a network and training them with adversarially generated examples.

The iterative nature of generating adversarial examples
makes their use in training to generate defenses computationally very intensive. For instance, the adversarially trained feature denoising model proposed by [57] takes 38 hours on 128 Nvidia V100 GPUs to train a baseline ResNet-101 with ImageNet. While we leverage this observation of noise amplification over the layers, our proposed approach avoids any training or fine-tuning of the model. Instead, we use a representative subset of training samples and their layer-wise pre-activation responses to construct mixture density based generative classifiers, which are then combined in an ensemble using ranking preferences.

Generative classifiers have achieved varying degrees of success as defense strategies against adversarial attacks. Recently, [18] studied the class-conditional generative classifiers and concluded that it is impossible to guarantee robustness of such models. More importantly, they highlight the challenges in training generative classifiers using maximum likelihood based objective and their limitations w.r.t. discriminative ability and identification of out-of-distribution samples. While we propose to use generative classifiers, we avoid using likelihood based measures for making classification decisions. Instead, we use rank-order preferences of these classifiers which are then combined using a Borda count-based voting scheme. Borda counts have been used in collective decision making and are known to be robust to various manipulative attacks [43].

In this paper, we present our defense against adversarial attacks on deep networks, referred to as Rank-aggregating Ensemble of Generative classifiers for robust predictions (REGroup). At inference time, our defense requires white-box access to a pre-trained model to collect the pre-activation responses at intermediate layers to make the final prediction. We use the training data to build our generative classifier models. Nonetheless, our strategy is simple, network-agnostic, does not require any training or fine-tuning of the network, and works well for a variety of adversarial attacks, even with varying degrees of hardness. Consistent with recent trends, we focus only on the ImageNet dataset to evaluate the robustness of our defense and report performance superior to defenses that rely on adversarial training [28] and random input transformation [41] based approaches. Finally, we present extensive analysis of our defense with two different architectures (ResNet and VGG) on different targeted and untargeted attacks. Our primary contributions are summarized below:

- We present REGroup, a retraining free, model-agnostic defense strategy that leverages an ensemble of generative classifiers over intermediate layers of the model.
- We model each layer-wise generative classifier as a simple mixture distribution of neural responses obtained from a subset of training samples. We discover that both positive and negative pre-activation values contain information that can help correctly classify adversarially perturbed samples.
- We leverage the robustness inherent in Borda-count based consensus over the generative classifiers.
- We show extensive comparisons and analysis on the ImageNet dataset spanning a variety of adversarial attacks.

2. Related Work

Several defense techniques have been proposed to make neural networks robust to adversarial attacks. Broadly, we can categorize them into two approaches that: 1. Modify training procedure or modify input before testing; 2. Modify network or change hyper-parameters and optimization procedure.

2.1. Modify Training/Inputs During Testing

Some approaches of defenses in this category are mentioned below. Adversarial training [62, 34, 54, 45]. Data compression [4] suppresses the high-frequency components and presents an ensemble-based defense approach. Data randomization [53, 56] based approaches apply random transformations to the input to defend against adversarial examples by reducing their effectiveness. PixelDefend [47] sets out to find the image with the highest probability within an $\epsilon$-neighbourhood of the original image, thereby moving the image back towards distribution seen in training data. Defense-GAN [44] tries to model the distribution of unperturbed images and at inference, it generates an image close to what was provided but without adversarial perturbations. These two methods use techniques to generate a clean version of the input and pass to the classifier.

2.2. Modify Network/Network Add-ons

Defenses under this category address the detection of adversarial attacks or cater to both detection and correction of prediction. The aim of detection only defenses is to highlight if an example is adversarial and prevent it from further processing. These approaches include employing a detector sub-network [33], training the main classifier with an outlier class [19], using convolution filter statistics [29], or applying feature squeezing [58] to detect adversarial examples. However, all of these methods have shown to be ineffective against strong adversarial attacks [9] [46]. Full defense approaches include applying defensive distillation [39] [37] to use the knowledge from the output of the network to re-train the original model and improve the resilience of a network to small perturbations. Another approach is to augment the network with a sub-network called Perturbation Rectifying
Network (PRN) [1] to detect the perturbations; if the perturbation is detected, then PRN is used to classify the input image. However, later it was shown that the Carlini Wagner (C&W) attack successfully defeated the defensive distillation approach.

2.3. ImageNet Focused Defense Approaches

A few approaches have been evaluated on the ImageNet dataset, most of which are based on input transformations or image denoising. Nearly all these defenses designed for ImageNet have failed a thorough evaluation, with a regularly updated list maintained at [31]. The approaches in [40] and [30] claimed 81% and 75% accuracy respectively under adversarial attacks. But after a thorough evaluation [2] and accounting for obfuscated gradients [3], the accuracy for both was reduced to 0%. Similarly, [55] and [20] claimed 86% and 75% respectively, but these were also reduced to 0% [3]. A different approach proposed in [25] claimed an accuracy 27.9% but later it was also reduced to 0.1% [16]. For a comprehensive related work on attacks and defenses, we suggest reader to refer [11].

3. REGroup Methodology

Well-trained deep neural networks have a hierarchical structure, where the early layers transform inputs to feature spaces capturing local or more generic patterns, while later layers aggregate this local information to learn more semantically relevant representations. In REGroup, we use many of the higher layers and learn class-conditional generative classifiers, which are simple mixture-distributions estimated from the pre-activation neural responses at each layer from a subset of training samples. An ensemble of these layer-wise generative classifiers is used to make the final prediction by performing a Borda count-based rank-aggregation. Ranking preferences have been used extensively in robust fitting problems in computer vision [12, 23, 49], and we show its effectiveness in introducing robustness in DNNs against adversarial attacks.

Fig. 1 illustrates the overall working of REGroup. The approach has three main components: First, we use each layer as a generative classifier that produces a ranking preference over all classes. Second, each of these class-conditional generative classifiers are modeled using a mixture-distribution over the neural responses of the corresponding layer. Finally, the individual layer’s class ranking preferences are aggregated using Borda count-based scoring to make the final predictions. We introduce the notation below and discuss each of these steps in detail in the subsections that follow.

Notation. In this paper, we will always use $\ell$, $i$ and $j$ for indexing the $\ell^{th}$ layer, $i^{th}$ feature map and the $j^{th}$ input sample respectively. The true and predicted class label will be denoted by $y$ and $\hat{y}$ respectively. A classifier can be represented in a functional form as $\hat{y} = F(x)$, it takes an input $x$ and predicts its class label $\hat{y}$. We define $\phi^{\ell i}$ as the $\ell^{th}$ layer’s $i^{th}$ pre-activation feature map, i.e., the neural responses before they pass through the activation function. For convolutional layers, this feature map $\phi^{\ell i}$ is a 2D array, while for a fully connected layer, it is a scalar value.

3.1. DNN Layers as Generative Classifiers

We use the highest $k$ layers of a DNN as generative classifiers that use the pre-activation neural responses to produce a ranking preference\(^1\) over all classes. The layer-wise generative classifiers are modeled as a class-conditional mixture distribution, which is estimated using only a pretrained network and a small subset $S$ of the training data. Let $S$ contain only correctly classified training samples\(^2\), which we can further divide into $M$ subsets, one for each class $i$, $S = \bigcup_{y=1}^{M} S_y$, where $S_y$ is the subset containing samples that have labels $y$.

3.1.1 Layerwise Neural Response Distributions

Our preliminary observations indicated that while the ReLU activations truncate the negative pre-activations during the forward pass, these values still contain semantically meaningful information. Our ablative studies in Fig. 5 confirm this observation and additionally, on occasion, we find that the negative pre-activations are complementary to the positive ones. Since the pre-activation features are real-valued, we compute the features $\phi^{\ell i}_{j}$ for the $j^{th}$ sample $x_j$, and define its positive ($P^{\ell i}_j$) and negative ($N^{\ell i}_j$) response accumulators as $P^{\ell i}_j = \max(0, \phi^{\ell i}_{j})$, $N^{\ell i}_j = \max(0, -\phi^{\ell i}_{j})$.

For convolutional layers, these accumulators represent the overall strength of positive and negative pre-activation responses respectively, when aggregated over the spatial dimensions of the $i^{th}$ feature map of the $\ell^{th}$ layer. On the other hand, for the linear layers, the accumulation becomes trivial with each neuron having a scalar response $\phi^{\ell i}_{j}$. We can now represent the $\ell^{th}$ layer by the positive and negative response accumulator vectors denoted by $P^{\ell}_{j}$ and $N^{\ell}_{j}$ respectively. We normalize these vectors and define the layer-wise probability mass function (PMF) for the positive and negative responses as $P^{\ell i}_j = \frac{P^{\ell i}_j}{||P^{\ell i}_j||_1}$ and $N^{\ell i}_j = \frac{N^{\ell i}_j}{||N^{\ell i}_j||_1}$ respectively.

Our interpretation of $P^{\ell i}_j$ and $N^{\ell i}_j$ as a PMF could be justified by drawing an analogy to the softmax output, which is also interpreted as a PMF. However, it is worth emphasizing that we chose the linear rescaling of the accumulator

\(^1\)A rank is assigned to each class based on a score. In the case of ImageNet dataset, the class with rank-1 is most preferred/likely class, while rank-1000 is the least preferred/likely class

\(^2\)We took 50,000 out of $\sim$ 1.2 millions training images from ImageNet dataset, 50 per class.
vectors rather than directly applying a softmax normalization. By separating out the positive and negative accumulators, we obtain two independent representations for each layer, which is beneficial to our rank-aggregating ensemble discussed in the following sections. A softmax normalization over a feature map comprising of positive and negative responses would have entirely suppressed the negative responses, discarding all its constituent semantic information. An additional benefit of the linear scaling is its simple computation. Algorithm 1 summarizes the computation of the layer-wise PMFs for a given training sample.

\begin{algorithm}
\caption{Layerwise PMF of neural responses.}
\begin{algorithmic}
\STATE $H \times W$ represents the spatial dimensions of pre-activation features. For $\ell$th convolutional layer the dimensions of feature maps $H \times W = r^\ell \times s^\ell$, and for linear layers the dimensions of neuron output $H \times W = 1 \times 1$.
\STATE \textbf{Input:} $x_j$ pre-activation features $\phi_j^{\ell i} \in \mathbb{R}^{H \times W}$
\FOR{$\ell \in [1..n]$}
\STATE $P_j^{\ell i} = \sum_{y} \max(0, \phi_j^{\ell i})$, \hspace{0.5cm} (sum over $H$, $W$)
\STATE $N_j^{\ell i} = \sum_{y} \max(0, -\phi_j^{\ell i})$, \hspace{0.5cm} (sum over $H$, $W$)
\ENDFOR
\STATE $P_j^{\ell} \leftarrow P_j^{\ell} + \delta$, \hspace{0.5cm} $N_j^{\ell} \leftarrow N_j^{\ell} + \delta$
\STATE $P_j^{\ell} \leftarrow \frac{P_j^{\ell}}{\sum_{y} P_j^{\ell}}, \hspace{0.5cm} N_j^{\ell} \leftarrow \frac{N_j^{\ell}}{\sum_{y} N_j^{\ell}}$ (PMFs)
\end{algorithmic}
\end{algorithm}

3.1.2 Layerwise Generative Classifiers

We model the layerwise generative classifiers for class $y$ as a class-conditional mixture of distributions, with each mixture component as the PMFs $P_j^{\ell i}$ and $N_j^{\ell i}$ for a given training sample $x_j \in S_y$. The generative classifiers corresponding to the positive and negative neural responses are then defined as the following mixture of PMFs

$$C_y^{+\ell} = \sum_{j: x_j \in S_y} \lambda_j P_j^{\ell i}, \hspace{0.5cm} C_y^{-\ell} = \sum_{j: x_j \in S_y} \lambda_j N_j^{\ell i} \hspace{0.5cm} (1)$$

where the weights $\lambda_j$ are nonnegative and add up to one in the respective equations. Here, $\lambda_j$ is proportional to the softmax probability of the sample $x_j$, and $\delta$ is the small constant used for numerical stability. We choose the weights to be proportional to the softmax probability value as predicted by the network given the input $x_j$. Using the subset of training samples $S$, we construct the class-conditional mixture distributions, $C^{+\ell}_y$ and $C^{-\ell}_y$ at each layer $\ell$ only once. At inference time, we input a test sample $x_j$ from the test set $T$, to the network and compute the PMFs $P_j^{\ell}$ and $N_j^{\ell}$ using Algorithm 1. As our test input is a PMF and the generative classifier is also a mixture distribution, we simply use the KL-Divergence between the classifier model $C^{+\ell}$ and the test sample $P_j^{\ell}$ as a classification score as

$$P_{KL}(\ell, y) = \sum_i C_y^{+\ell i} \log \left( \frac{C_y^{+\ell i}}{P_j^{\ell i}} \right), \forall y \in \{1, \ldots, M\}$$

and similarly for the negative PMFs

$$N_{KL}(\ell, y) = \sum_i C_y^{-\ell i} \log \left( \frac{C_y^{-\ell i}}{N_j^{\ell i}} \right), \forall y \in \{1, \ldots, M\}$$

We use a simple classification rule and select the predicted class $\hat{y}$ as the one with the smallest KL-Divergence with the test sample PMF. However, rather than identifying $\hat{y}$, at this stage we are only interested in rank-ordering the classes, which we simply achieve by sorting the KL-Divergences (Eqns. (2) and (3)) in ascending order. The resulting ranking preferences of classes for the $\ell$th layer are given below in Eqns. (4) and (5) respectively. Where, $R_y^{+\ell}$ is the rank (position of $y^{th}$ class in the ascending order of KL-Divergences in $P_{KL}$) of $y^{th}$ class in the $\ell$th layer preference list $R_y^{\ell}$.

$$R_y^{+\ell} = [R_y^{+1}, R_y^{+2}, \ldots, R_y^{+\ell}, \ldots, R_y^{+M}] \hspace{0.5cm} (4)$$

$$R_y^{-\ell} = [R_y^{-1}, R_y^{-2}, \ldots, R_y^{-\ell}, \ldots, R_y^{-M}] \hspace{0.5cm} (5)$$

3.2. Robust Predictions with Rank Aggregation

Rank aggregation based preferential voting for making group decisions is widely used in selecting a winner in a democratic setup [43]. The basic premise of preferential voting is that $n$ voters are allowed to rank $m$ candidates in the order of their preferences. The rankings of all $n$ voters are then aggregated to make a final prediction.

Borda count [6] is one of the approaches for preferential voting that relies on aggregating the rankings of all the voters to make a collective decision [43, 24]. The other popular voting strategies to find a winner out of $m$ different choices include Plurality voting [52], and Condorcet winner [60]. In Plurality voting, the winner would be the one who gets the maximum fraction of votes, while Condorcet winner is the one who gets the majority votes.

3.2.1 Rank Aggregation Using Borda Count

Borda count is a generalization of the majority voting. In a two-candidates case it is equivalent to majority vote. The Borda count for a candidate is the sum of the number of candidates ranked below it by each voter. In our setting, while processing a test sample $x_j \in T$, every layer acts as two independent voters based on $P^{\ell}$ and $N^{\ell}$. The number of classes i.e $M$ is the number of candidates. The Borda
count for the $y^{th}$ class at the $\ell^{th}$ layer is denoted by $B_{\ell y}^{k_\ell} = B_{\ell y}^{k_\ell} + B_{\ell y}^{n_{\ell}}$, where $B_{\ell y}^{k_\ell}$ and $B_{\ell y}^{n_{\ell}}$ are the individual Borda count of both the voters and computed as shown in eq. (6).

$$B_{\ell y}^{k_\ell} = (M - R_{\ell y}^{k_\ell}), \quad B_{\ell y}^{n_{\ell}} = (M - R_{\ell y}^{n_{\ell}}) \quad (6)$$

### 3.2.2 Hyperparameter settings

We aggregate the Borda counts of highest $k$ layers of the network, which is the only hyperparameter to set in REGroup. Let $B_{k}^{k_y}$ denote the aggregated Borda count of $y^{th}$ class from the last $k$ layers irrespective of the type (convolutional or fully connected). Here, $n$ is the total number of layers. The final prediction would be the class with maximum aggregated Borda count.

$$B_{k}^{k_y} = \sum_{\ell = n-k+1}^{n} B_{\ell y}^{k_y}$$

$$= \sum_{\ell = n-k+1}^{n} B_{\ell y}^{k_\ell} + B_{\ell y}^{n_{\ell}}, \forall y \in \{1..M\}$$

$$\hat{y} = \arg\max_{y} B_{k}^{k_y} \quad (7)$$

To determine the value of $k$, we evaluate REGroup on 10,000 correctly classified samples from the ImageNet Validation set at each layer, using per layer Borda count i.e $\hat{y} = \arg\max_{y} B_{\ell y}^{k_\ell}$. We select $k$ to be the number of later layers at which we get at-least 75% accuracy. This can be viewed in the context of the confidence of individual layers on discriminating samples of different classes. We follow the above heuristic and found $k = 5$ for both the architectures ResNet-50 and VGG-19, which we use in all our experiments. An ablation study with all possible values of $k$ is included in section 5.

### 4. Experiments

In this section, we evaluate robustness of REGroup against state-of-the-art attack methods. We follow the recommendations on defense evaluation in [8].

**Attack methods.** We consider attack methods in the following two categories: **gradient-based** and **gradient-free**. Gradient-Based Attacks. Within this category, we consider two variants, restricted and unrestricted attacks. The restricted attacks generate adversarial examples by searching an adversarial perturbations within the bound of $L_p$ norm, while unrestrained attacks generate adversarial example by manipulating image-based visual descriptors. Due to restriction on the perturbation the adversarial examples generated by restricted attacks are similar to the clean original image, while unrestrained attacks generate natural-looking adversarial examples, which are far from the clean original image in terms of $L_p$ distance. We consider the following, Restricted attacks: PGD [32], DeepFool [35], C&W [10] and Trust Region [59], and Unrestricted attack: cAdv [5] semantic manipulation attack. An example of cAdv is shown in Fig. 2.

**Gradient-Free Attacks.** The approaches in this category does not have access to the network weights. We consider following attacks: SPSA [51], Boundary [7] and Spatial [17]. Refer supplementary for the attack specific hyperparameters detail.

![cAdv examples](https://download.pytorch.org/models/vgg19-dcbb9e9d.pth)

**Network architectures.** We consider ResNet-50\(^3\) and VGG-19\(^4\) architectures, pre-trained on ImageNet dataset.

**Datasets.** We present our evaluations, comparisons and analysis only on ImageNet [14] dataset. We use the subsets of full ImageNet validation set as described in Tab. 1. Note: V10K, V2K and V10C would be different for ResNet-50 and VGG-19, since an image classified correctly by ResNet-50 need not be classified correctly by the VGG-19.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>V50K</td>
<td>Full ImageNet validation set with 50000 images</td>
</tr>
<tr>
<td>V10K</td>
<td>A subset of 10000 correctly classified images from V50K set. 10 Per class.</td>
</tr>
<tr>
<td>V2K</td>
<td>A subset of 2000 correctly classified images from V50K set. 2 Per class.</td>
</tr>
<tr>
<td>V10C</td>
<td>A subset of correctly classified images of 10 sufficiently different classes.</td>
</tr>
</tbody>
</table>

Table 1. Dataset used for evaluation and analysis.

### 4.1. Performance on Gradient-Based Attacks

**Comparison with adversarial-training/ fine-tuning.** We evaluate REGroup on clean samples as well as adversarial examples generated using PGD ($\epsilon = 16$) from V50K dataset, and compare it with prior state-of-the-art works. The results are reported in Tab. 2, and we see that REGroup outperforms the state-of-the-art input transformation based defense BaRT [41], both in terms of the clean and adversarial samples (except in the case of Top-1 accuracy with $k = 10$, which is the number of input transformations used in BaRT). We see that while our performance on clean samples decreases when compared to adversarial training (Inception v3), it improves significantly on adversarial examples with a high $\epsilon = 16$. While our method is not directly comparable with adversarially trained Inception v3 and ResNet-152, because the base models are different, a similar decrease in the accuracy over clean samples is reported in their paper. The trade-off between robustness and the standard accuracy has been studied in [15] and [50].

\(^3\)https://download.pytorch.org/models/resnet50-19c8e357.pth

\(^4\)https://download.pytorch.org/models/vgg19-dcbb9e9d.pth
An important observation to make with this experiment is, if we set aside the base models of ResNets and compare Top-1 accuracies on clean samples of full ImageNet validation set, our method (REGroup) without any adversarial training/fine-tuning either outperforms or performs similar to the state-of-the-art adversarial training/fine-tuning based methods [41, 57].

<table>
<thead>
<tr>
<th>(Dataset used: V50K),</th>
<th>Clean Images</th>
<th>Attacked Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Top-1</td>
<td>Top-5</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>76</td>
<td>93</td>
</tr>
<tr>
<td>Inception v3</td>
<td>78</td>
<td>94</td>
</tr>
<tr>
<td>ResNet-152</td>
<td>79</td>
<td>94</td>
</tr>
<tr>
<td>Inception v3 w/Adv. Train</td>
<td>78</td>
<td>94</td>
</tr>
<tr>
<td>ResNet-152 w/Adv. Train</td>
<td>63</td>
<td>-</td>
</tr>
<tr>
<td>ResNet-152 w/Adv. Train w/ denoise</td>
<td>66</td>
<td>-</td>
</tr>
<tr>
<td>ResNet-50-BaRT, $\epsilon=5$</td>
<td>65</td>
<td>85</td>
</tr>
<tr>
<td>ResNet-50-BaRT, $\epsilon=10$</td>
<td>65</td>
<td>85</td>
</tr>
<tr>
<td>ResNet-50-REGroup</td>
<td>66</td>
<td>86</td>
</tr>
</tbody>
</table>

Table 2. Comparison with adversarially trained and fine-tuned classification models. Top-1 and Top-5 classification accuracy (%) of adversarial trained (Inception V3 [28] and ResNet-152 [57]) and fine-tuned (ResNet-50 BaRT [41]) classification models. Clean Images are the non-attacked original images. Results are divided into three blocks, the top block include original networks, middle block include defense approaches based on adversarial re-training/fine-tuning of original networks, bottom block is our defense without re-training/fine-tuning. Results of the competing methods are taken from their respective papers. '-' indicate the results were not provided in the respective papers.

Performance w.r.t PGD Adversarial Strength. We evaluate REGroup w.r.t. the maximum perturbation of the adversary. The results are reported in Fig. 3(a). REGroup outperforms both the adversarial training [28] and BaRT [41]. Both adversarial training and BaRT have shown protection against PGD adversarial attacks with a maximum perturbation strength $\epsilon=16$ and $\epsilon=32$ respectively, however, we additionally show the results with $\epsilon=40$ on full ImageNet validation set. We also note that with increasing perturbation strength, our defense’s accuracy is also strictly decreasing. This is in accordance with [8], where transitioning from a clean image to noise should yield a downward slope in accuracy, else there could be some form of gradient masking involved. While it may seem $\epsilon=40$ is a large perturbation budget and it will destroy the object information in the image completely, but we would like to emphasize that it is not the case when using large size images. A comparison of PGD examples generated with $\epsilon=40$ using CIFAR-10 (32 × 32) and ImageNet (224 × 224) images is shown in Fig. 3(b).

Performance on Un-Targeted Attacks. We evaluate REGroup on various untargeted attacks and report results in Tab. 3. The perturbation budgets ($\epsilon$) and dataset used for the respective attacks are listed in the table. With the exception of the maximum perturbation allowed, we used default parameters given by FoolBox [42]. Due to space limitations, the attack specific hyper-parameters detail are included in the supplementary. We observe that the performance of our defense is quite similar for both the models employed. This is due to the attack-agnostic nature of our defense. We achieve 48% accuracy (ResNet-50) for PGD attack using our defense which is significant given that PGD is considered to be one of the strongest attacks.

![Figure 3. Top-1 and Top-5 accuracy(%) w.r.t PGD adversarial strength. Comparison with adversarial training based method [28] and fine-tuning using random input transformations based method (BaRT) [41] with Expectation Over Transformation (EOT) steps 10 and 40, against the PGD perturbation strength ($\epsilon$). The results of the competing methods are taken from their respective papers. Dataset used: V50K.](image-url)
among the class of first order adversaries.

**Performance on Unrestricted, Untargeted Semantic Manipulation Attacks.** We consider V10C dataset for cAdv attack. We use the publicly released source code by the authors. Specifically we use cAdv variant with the parameters suggested by the authors. The results are reported in Tab. 3.

**Performance on Targeted Attacks.** We consider V2K dataset for targeted attacks and report the performance on PGD and C&W targeted attacks in Tab. 3. Target class for such attacks is chosen uniformly at random from the 1000 ImageNet classes apart from the original (ground-truth) class.

**Performance on PGD attack with High Confidence.** We evaluate REGroup on PGD examples on which the network makes highly confident predictions using SoftMax. We generate un-targeted and targeted adversarial examples using PGD attack with a constraint that the network’s confidence of the prediction of adversarial examples is at-least 90%. For this experiment we do not put constraint on the adversarial perturbation i.e $\epsilon$. Results are reported in Tab. 3.

### 4.2. Performance on Gradient-Free Attacks

Several studies [3], [38] have observed a phenomenon called *gradient masking*. This phenomenon occurs when a practitioner unintentionally or intentionally proposes a defense which does not have meaningful gradients, either by reducing them to small values (vanishing gradients), removing them completely (shattered gradients) or adding some noise to it (stochastic gradient). Gradient masking based defenses hinder the gradient computation and in turn inhibit gradient-based attacks, thus providing a false sense of security. Therefore, to establish the robustness of a defense against adversarial attacks in general, it is important to rule out that a defense relies on gradient masking.

To ensure that REGroup is not masking the gradients we follow the standard practice [36] [61] and evaluate on strong gradient-free SPSA [51] attack. In addition to SPSA, we also show results on two more gradient-free attacks, Boundary [7] and Spatial [17] attack. The results are reported in Tab. 4.

<table>
<thead>
<tr>
<th>Data TA / HC</th>
<th>$\epsilon$</th>
<th>ResNet-50 UN / SMax REGroup</th>
<th>VGG-19 UN / SMax REGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>V10K</td>
<td>4 (L_max)</td>
<td>4911 0 71 5789 0 58</td>
<td></td>
</tr>
<tr>
<td>Boundary</td>
<td>V10K</td>
<td>2 (L_2) 10000 0 50 10000 0 50</td>
<td></td>
</tr>
<tr>
<td>Spatial</td>
<td>V10K</td>
<td>2 (L_2) 2624 0 36 2634 0 30</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4. Performance on Gradient-Free Attacks.** Top-1 (%) classification accuracy comparison between SoftMax (SMax) and REGroup. Legends are same as in Tab. 3.

The consistent superior performance on both gradient-based (both restricted and unrestricted) and gradient free attack shows REGroup is not masking the gradients and is attack method agnostic.

### 5. Analysis

#### 5.1. Accuracy vs number of layers ($k$)

We report performance of REGroup on various attacks reported in Tab. 3 for all possible values of $k$. The accuracy of VGG-19 w.r.t. the various values of $k$ is plotted in Fig. 4. We observe a similar accuracy vs $k$ graph for ResNet-50 and note that a reasonable choice of $k$ made based on this graph does not significantly impact REGroup’s performance. Refer Fig. 4, the ‘Agg’ stands for using aggregated Borda count $B^{k_y}$. PGD(V10K, UN), D fool, C&W(V10K, UN) and Trust Region are the same experiments as reported in Tab. 3, but with all possible values of $k$. ‘Per_Layer_V10K’ stands for evaluation using per layer Borda count i.e $\bar{y} = \arg\max_y B^{\ell_y}$ on a separate 10,000 correctly classified subset of validation set. In all our experiments we choose the $k$-highest layers where ‘Per_Layer_V10K’ has at-least 75% accuracy. A reasonable change in this accuracy criteria of 75% would not affect the results on adversarial attacks significantly. However, a substantial change (to say 50%) deteriorates the performance on clean sample significantly.

The phenomenon of decrease in accuracy of clean samples vs robustness has been studied in [15] [50].

#### 5.2. Effect of positive and negative pre-activation responses

We report the impact of using positive, negative and a combination of both pre-activation responses on the performance of REGroup in Fig. 5. We consider three variants of Borda count rank aggregation from later $k$ layers. Pos: $B_{+}^{k_y} = \sum_{\ell = n - k + 1}^{n} B_{+}^{\ell_y}$, Neg: $B_{-}^{k_y} = \sum_{\ell = n - k + 1}^{n} B_{-}^{\ell_y}$, and Pos+Neg: $B^{k_y} = \sum_{\ell = n - k + 1}^{n} B_{+}^{\ell_y} + B_{-}^{\ell_y}$. We report...
the Top-1 accuracy (%) of the attacks experiment as set up in Tab. 3 (DF: DFool, C&W, TR: Trust Region), in Tab. 4 (BD: Boundary, SP: Spatial), and in Fig. 3 (PGD2, PGD4 and PGD8, with $\epsilon = 2, 4$ and 8 respectively). From the bar chart it is evident that in some experiments, Pos performs better than Neg (e.g. UN, TR), while in others Neg is better than Pos only (e.g. UN, DF). It is also evident that Pos+Neg occasionally improve the overall performance, and the improvement seems significant in the targeted C&W attacks for both the ResNet-50 and VGG-19. We leave it to the design choice of the application, if inference time is an important parameter, then one may choose either Pos or Neg to reduce the inference time to approximately half of what is reported in Tab. 6.

5.3. Results on CIFAR-10

While we mainly show results on large-scale dataset (ImageNet), we believe scaling down the datasets to one like CIFAR10 will not have a substantial impact on REGroup’s performance. We evaluate REGroup on CIFAR10 dataset using VGG-19 based classifier. We construct generative classifiers using CIFAR-10 dataset following the same protocol as for the ImageNet case described in the Sec. 3.1. We apply PGD attack with $\epsilon = 4$ and generate adversarial examples. The results are included in the Tab. 5.

<table>
<thead>
<tr>
<th>VGG-19</th>
<th>SMax</th>
<th>REGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean</td>
<td>10000</td>
<td>92</td>
</tr>
<tr>
<td>PGD Untargeted</td>
<td>9243</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5. Performance on CIFAR10. Comparison of Top-1 classification accuracy between SoftMax (SMax) and REGroup based final classification. #S is the number of images for which the attacker is successfully able to generate adversarial examples using PGD attack and the accuracies are reported with respect to the #S samples, hence the 0% accuracies with the SoftMax (SMax).

5.4. Inference time using REGroup

Since we suggest to use REGroup as a test time replacement of SoftMax, we compare the inference time on both CPU and GPU in Tab. 6. We use a machine with an i7-8700 CPU and GTX 1080 GPU.

<table>
<thead>
<tr>
<th>VGG-19</th>
<th>SMax</th>
<th>REGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>0.02</td>
<td>0.06</td>
</tr>
<tr>
<td>GPU</td>
<td>0.13</td>
<td>0.35</td>
</tr>
</tbody>
</table>

Table 6. Inference time comparison. REGroup vs SoftMax

6. Conclusion

In this work, we have presented a simple, scalable, and practical defense strategy that is model agnostic and does not require any re-training or fine-tuning. We suggest to use REGroup at test time to make a pre-trained network robust to adversarial perturbations. There are three aspects of REGroup that justify its success. First, instead of using a maximum likelihood based prediction, REGroup adopts a ranking preference based approach. Second, aggregation of preferences from multiple layers lead to group decision making, unlike SoftMax that relies on the output of the last layer only. Using both positive and negative layerwise responses help contribute to the robustness of REGroup. Third, there exists inherent robustness of Borda count based rank aggregation in the presence of noisy individual voters [43], [24]. Hence, where SoftMax fails to predict the correct class of an adversarial example, REGroup takes ranked preferences from multiple layers and builds a consensus using Borda count to make robust predictions. Our promising empirical results indicate that deeper theoretical analysis of REGroup would be an interesting direction to pursue. One direction of analysis could be inspired from the recently proposed perspective of neurons as cooperating classifiers [13].

Acknowledgement

This work was partially supported by the Infosys Center for Artificial Intelligence, IIIT-Delhi.
References


