Retrieval-Augmented Convolutional Neural Networks against Adversarial Examples

Jake Zhao (Junbo) *
New York University
j.zhao@nyu.edu

Kyunghyun Cho
New York University & Facebook AI Research
CIFAR Azrieli Global Scholar
kyunghyun.cho@nyu.edu

Abstract

We propose a retrieval-augmented convolutional network (RaCNN) and propose to train it with local mixup, a novel variant of the recently proposed mixup algorithm. The proposed hybrid architecture combining a convolutional network and an off-the-shelf retrieval engine was designed to mitigate the adverse effect of off-manifold adversarial examples, while the proposed local mixup addresses on-manifold ones by explicitly encouraging the classifier to locally behave linearly on the data manifold. Our evaluation of the proposed approach against seven readily-available adversarial attacks on three datasets–CIFAR-10, SVHN and ImageNet–demonstrate the improved robustness compared to a vanilla convolutional network, and comparable performance with the state-of-the-art reactive defense approaches.

1. Introduction

Since the initial investigation in [35], adversarial examples have drawn a large interest. Various methods for both generating adversarial examples as well as protecting a classifier from them have been proposed. According to [9], adversarial examples can be categorized into those off the data manifold, which is defined as a manifold on which training examples lie, and those on the data manifold. Off-manifold adversarial examples occur as the classifier does not have a chance to observe any off-manifold examples during training, which is a natural consequence from the very definition of the data manifold. On-manifold adversarial examples however exist between training examples on the data manifold. There are two causes behind this phenomenon; (1) the sparsity of training examples and (2) the non-smooth behavior of the classifier on the data manifold.

We propose to tackle both off- and on-manifold adversarial examples by incorporating an off-the-shelf retrieval mechanism which indexes a large set of examples and training this combination of a deep neural network classifier and the retrieval engine to behave linearly on the data manifold using a novel variant of the recently proposed mixup algorithm [39], to which we refer as "local mixup."

*Work done while the author worked at Facebook AI Research.

The retrieval mechanism efficiently selects a subset of neighboring examples from a candidate set near the input. These neighboring examples are used as a local approximation to the data manifold in the form of a feature-space convex hull onto which the input is projected. The classifier then makes a decision based on this projected input. This addresses off-manifold adversarial examples. Within this feature-space convex hull, we encourage the classifier to behave linearly by using local mixup to further address on-manifold adversarial examples.

We evaluate the proposed approach, called a retrieval-augmented classifier, with a deep convolutional network [22] on object recognition. We extensively test the retrieval-augmented convolutional network (RaCNN) on datasets with varying scales; CIFAR-10 [20], SVHN [26] as well as ImageNet [7], against seven readily-available adversarial attacks including both white-box (FGSM, iFGSM, DeepFool, L-BFGS, CW and PGD) and black-box attacks (Boundary). Our experiments reveal that the RaCNN is more robust to these attacks than the vanilla convolutional network, and also achieve comparable robustness with other *reactive* state-of-the-art defense methods. To the best of our knowledge, RaCNN by far achieves the best results among the *proactive* defense methods.

2. Retrieval-Augmented CNN

In [9], it was recently demonstrated that adversarial examples exist both on and off the data manifold. This result suggests that it is necessary to tackle both types of adversarial examples to improve the robustness of a deep neural network based classifier to adversarial examples. In this section, we describe our approach toward building a more robust classifier by combining an off-the-shelf retrieval engine and a variant of the recently proposed mix-up learning strategy.

Let $D' = \{(x'_1, y'_1), \ldots, (x'_M, y'_M)\}$ be a candidate set of examples. This set may be created as a subset from a training set $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ or may be an entire separate set. We use D' as a proxy to the underlying data manifold. $k(x, x')$ is a distance function that measures the dissimilarity between two inputs x and x'. In order to facilitate the use of an off-the-shelf retrieval engine, we use $k(x, x') = \|\phi'(x) - \phi'(x')\|^2$, where ϕ' is a predefined, or pretrained, feature extractor. We assume the existence of a readily-available retrieval engine F that takes x as input and returns the K nearest neighbors in D' according to $k(x, x')$. We then have a deep neural network classifier composed of a feature extraction ϕ and a classifier g. This classifier is trained on a training set D, taking into account the extra set D' and the retrieval engine.

2.1. Inference

Local Characterization of Data Manifold Given a new input x, we use the retrieval engine F to retrieve the examples x'_k's from D' that are closest to x: $F(x) = \{x'_1, \ldots, x'_K\}$. We then build a feature-space convex hull

$$\mathcal{C}(F(x)) = \left\{ \sum_{k=1}^{K} \alpha_k \phi(x'_k) \middle| \sum_{k=1}^{K} \alpha_k = 1 \wedge \forall k : \alpha_k \geq 0 \right\}.$$

As observed earlier, linear interpolation of two input vectors in the feature space of a deep neural network often corresponds to a plausible input vector [2, 19, 27]. Based on this, we consider the feature-space convex hull $\mathcal{C}(F(x))$ as a reasonable local approximation to the data manifold.

Trainable Projection Exact projection of the input x onto this convex hull $\mathcal{C}(F(x))$ requires expensive optimization, especially in the high-dimensional space. As we consider a deep neural network classifier, the dimension of the feature space ϕ' could be hundreds or more, making this exact projection computationally unfeasible. Instead, we propose to learn a goal-driven projection procedure based on the attention mechanism [1].

We compare each input $x'_k \in F(x)$ against x and compute a score: $\beta_k = \phi(x'_k)^\top U \phi(x)$, where U is a trainable weight matrix [24]. These scores are then normalized to form a set of coefficients: $\alpha_k = \frac{\exp(\beta_k)}{\sum_{k'=1}^{K} \exp(\beta_{k'})}$. These coefficients α_k's are then used to form a projection point of x in the feature-space convex hull

$$\mathcal{C}(F(x)) : \mathcal{P}(x) = \mathcal{P}_{\mathcal{C}(F(x))}(x) = \sum_{k=1}^{K} \alpha_k \phi(x'_k).$$

This trainable projection could be thought of as learning to project an off-manifold example on the locally-approximated manifold to maximize the classification accuracy.

Classification The projected feature $\mathcal{P}_{\mathcal{C}(F(x))}(x)$ now represents the original input x and is fed to a final classifier g. In other words, we constrain the final classifier to work only with a point inside a feature-space convex hull of neighboring training examples. This constraint alleviates the issue of the classifier's misbehaviors in the region outside the data manifold up to a certain degree.[1]

Randomization At the expense of computational overhead, we can improve the robustness further. We retrieve $K' = cK$ examples, where $c \gg 1$, and uniformly select K examples at random to form a feature-space convex hull. We evaluate this approach later against the strongest attack [5].

2.2. Training

The output of the classifier $g(\mathcal{P}(x))$ is almost fully differentiable w.r.t. the classifier g, both of the features extractors (ϕ' and ϕ) and the attention weight matrix U, except for the retrieval engine F.[2] This allows us to train the entire pipeline using backpropagation [31] and gradient-based optimization.

Local Mixup This is however not enough to ensure the robustness of the proposed approach to on-manifold adversarial examples. During training, the classifier g only observes a very small subset of any feature-space convex hull. Especially in a high-dimensional space, this greatly increases the chance of the classifier's misbehaviors within these feature-space convex hulls, as also noted in [9]. In order to address this issue, we propose to augment learning with a local variant of the recently proposed mix-up algorithm [39].

The goal of the original mixup is to encourage a classifier to act linearly between any pair of training examples. This is done by linearly mixing in two randomly-drawn training examples and creating a new linearly-interpolated example pair during training. Let two randomly-drawn pairs be (x_i, y_i) and (x_j, y_j), where y_i and y_j are one-hot vectors in the case of classification. Mixup creates a new pair $(\lambda x_i + (1 - \lambda) x_j, \lambda y_i + (1 - \lambda) y_j)$ and uses it as a training example, where $\lambda \in [0, 1]$ is a random sample from a beta distribution. We call this original version *global* mixup, as it increases the linearity of the classifier between any pair of training examples.

[1] The quality of the local approximation may not be uniformly high across the input space.

[2] The introduction of this non-differentiable retrieval engine further contributes to the increased robustness.

It is however unnecessary for our purpose to use global mixup, as our goal is to make the classifier better behave (i.e., linearly behave) within a feature-space convex hull $\mathcal{C}(F(x))$. Thus, we use a *local* mixup in which we uniformly sample the convex coefficients α_k's at random to create a new mixed example pair $(\sum_{k=1}^{K} \alpha_k \phi(x'_k), \sum_{k=1}^{K} \alpha_k y'_k)$. We use the Kraemer Algorithm [33].

Overall We use stochastic gradient descent (SGD) for training. At each update, we perform N_{CE} descent steps for the usual classification loss, and N_{MU} descent steps for the proposed local mixup.

2.3. Retrieval Engine F

The proposed approach does not depend on the specifics of a retrieval engine F. Any off-the-shelf retrieval engine that supports dense vector lookup could be used, enabling the use of a very large-scale D' with latest fast dense vector lookup algorithms [16]. In this work, we used a more rudimentary retrieval engine based on locality-sensitive hashing [6] with a reduced feature dimension using random projection [3], as the sizes of candidate sets D' contain approximately 1M or less examples. The key $\phi'(x)$ was chosen to be a pretrained deep neural network without the final fully-connected classifier layers [21, 14].

3. Attack Scenarios

S1 (Direct Attack) In this work, we consider the candidate set D' and the retrieval engine which indexes it to be "hidden" from the outside world. This property makes a usual white-box attack more of a *gray-box* attack in which the attacker has access to the entire system except for the retrieval part.

S2 (Retrieval Attack) Despite the hidden nature of the retrieval engine and the candidate set, it is possible for the attacker to confuse the retrieval engine, if they could access the feature extractor ϕ'. We furthermore give the attacker the access not only to ϕ' but the original classifier g' which was tuned together with ϕ'. This allows the attacker to create an adversarial example on $g'(\phi'(x))$ that could potentially disrupt the retrieval process. Although unlikely in practice, we test this second scenario to investigate the possibility of compromising the retrieval engine.

4. Attack Methods

Fast gradient sign method (FGSM) creates an adversarial example by adding the scaled sign of the gradient of the loss function L computed using a target class $\hat{y}$ to the input. **Iterative FGSM (iFGSM)** improves upon the FGSM by iteratively modifying the input x for a fixed number of steps. **DeepFool** was proposed in [25] to create an adversarial example by finding a residual vector $r \in \mathbb{R}^{\dim(x)}$ with the minimum L_p-norm with the constraint that the output of a classifier must flip. **L-BFGS** from [36] more explicitly constrains the input to lie inside a tight box defined by training examples using L-BFGS-B [40]. **CW** from [5] directly solves the original formulation of adversarial examples [35] and has been found recently to defeat most of the recently proposed defense mechanisms [4]. We consider this the strongest attack and test the self-ensemble approach against it.

Boundary brendel2017decision proposed a powerful black-box attack, or more specifically decision-based attack, that requires neither the gradient nor the predictive distribution. It only requires the final decision of the classifier. Starting from an adversarial example, potentially far away from the original input, it iteratively searches for a next adversarial example closer to the original input.

5. Related Work

Data-independent transformation aims at minimizing regions that are off the data manifold. dziugaite2016study demonstrated that JPEG-compressed images suffer less from adversarial attacks. lu2017no suggest that trying various scaling of an image size could overcome adversarial attacks. guo2017countering use compressed sensing to transform an input image by reconstructing it from its lower-resolution version while minimizing the total variation [30]. More recently, buckman2018thermometer proposed to discretize each input dimension using thermometer coding. These approaches are attractive due to their simplicity, but there have some work showing that it is not enough to defend against sophisticated adversarial examples [32].

Data-Dependent Transformation relies on data-dependent transformation. gu2014towards use a denoising autoencoder [38]. samangouei2018defensegan and song2017pixeldefend respectively use a pixelCNN [37] and generative adversarial network [10] to replace an input image with a nearby image. Instead of using a separately trained generative model, guo2017countering uses a technique of image quilting [8]. These approaches are similar to our use of a retrieval engine. They however do not address the issue of misbehaviors of a classifier on the data manifold.

Adversarial training trains a classifier on both training examples and adversarial examples generated on-the-fly [11]. lee2017generative extended this procedure by introducing a GAN that learns to generate adversarial examples. Instead, **robust optimization** directly modifies a learning algorithm to induce robustness. cisse2017parseval proposed Parseval training that encourages the Lipschitz constant to be less than one. More

	Clean	FGSM			iFGSM			DeepFool		
$\overline{L_2}$	0	1e-04	2e-04	4e-04	1e-05	2e-05	8e-05	1e-05	2e-05	8e-05
Baseline	**85.15**	14.05	7.5	4.22	55.2	26.17	2.59	26.04	11.72	0.34
RC-K5	72.57	42.97	34.29	24.55	72.57	72.48	45.46	64.34	61.34	60.96
RC-K5-MU	75.6	46.37	37.9	28.11	74.89	74.89	48.12	66.96	63.84	63.55
RC-K10	79.52	52.95	43.9	**33.77**	79.12	79	**55.27**	72.89	71.81	71.14
RC-K10-MU	80.80	**53**	**44.01**	33.47	**79.87**	**79.72**	54.36	**73.63**	**72.35**	**71.26**

Table 1. The CIFAR-10 classifiers' robustness to the adversarial attacks in the S2 (Retrieval Attack)

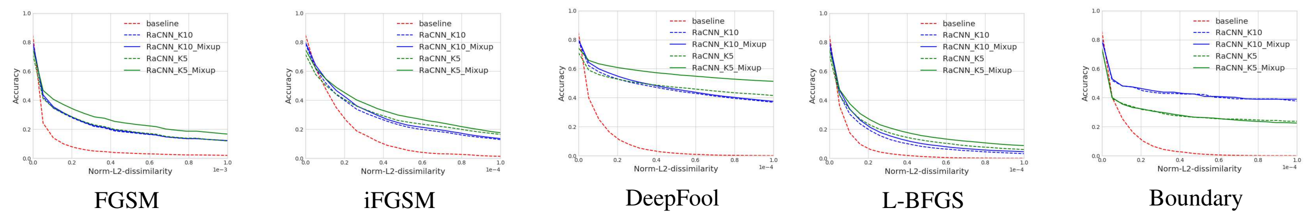


Figure 1. The CIFAR-10 classifiers' robustness to the adversarial attacks in the Scenario 1 (Direct Attack). The x-axis indicates the strength of attack in terms of the normalized L_2 distance. The y-axis corresponds to the accuracy.

recently, sinha2018certifiable proposed a tractable robust optimization algorithm It ensures that the classifier well-behaves in the neighborhood of each training point.

Retrieval-Augmented Neural Networks The proposed approach tightly integrates an off-the-shelf retrieval engine into a deep neural network. A similar approach has recently gained popularity. gu2017search use a text-based retrieval engine to build a non-parametric neural machine translation system. wang2017k proposed a similar approach to text classification, and guu2017generating to language modeling. It was also applied to online learning [23, 34].

6. Experiments

Datasets We test the proposed approach (RaCNN or **RC**) on three datasets of different scales. CIFAR-10 has 50k training and 10k test examples, with 10 classes. SVHN has 73k training and 26k test examples, with 10 classes. ImageNet has 1.3M training and 50k validation examples with 1,000 classes. For CIFAR-10 and ImageNet, we use the original training set as a candidate set, i.e., $D' = D$, while we use the extra set of 531k examples as a candidate set in the case of SVHN. The overall training process involves data augmentation on D but not D'.

Architectures We train a deep convolutional network for each dataset, remove the final fully-connected layers and use the remaining stack as a fixed feature extractor ϕ for retrieval. We use the same convolutional network from above for the RaCNN (RC) as well (separated into ϕ and g by the final average pooling) for each dataset. For CIFAR-10 and SVHN, we train ϕ and g from scratch. For ImageNet, on the other hand, we fix $\phi = \phi'$ and train g from the pre-trained ResNet-18 above. The latter was done, as we observed it greatly reduced training time in the preliminary experiments.

Training and Evaluation We use Adam [18] as an optimizer and investigate the influence of the newly introduced components–retrieval and local mixup (**MU**)– by varying $K \in \{5, 10\}$ and $N_{\text{MU}} \in \{0, 5\}$.

In addition to the accuracy on the clean test set, we look at the accuracy per the amount of perturbation used to create adversarial examples. We use the default `MeanSquaredDistance` from the Foolbox library; this amount is computed as a normalized L_2 distance between the original example x and its perturbed version $\tilde{x}$: $\overline{L_2}(x, \tilde{x}) = \frac{\|x-\tilde{x}\|_2^2}{\dim(x)*\left(\max(x)-\min(x)\right)^2}$. We use Foolbox [29, 28] for all the attacks other than CW for which we use the implementation from [12, 13]. [3] Our attacks are generally performed with clipping the outbounded pixel values at each step.

6.1. CIFAR-10

In the CIFAR-10 experiments, our model contains 6 convolutional layers and 2 fully-connected layers. For each layer we employ batch normalization [15] and ReLU following convolution.

[3]Noted that the Foolbox implementation searches over many attack hyperparameters to obtain the most optimized perturbation distance while achieving fooling. The implementation from [12, 13] differs in that it uses one empirically suitable set of hyperparameters to save computation. That being said, it might not be fair to compare attack strengths if they are implemented in different libraries.

S1 (Direct Attack) We present in Fig. 1 the effect of adversarial attacks with varying strengths (measured in the normalized L_2 distance) on both the vanilla convolutional network (Baseline) and the proposed RaCNN's with various settings. Across all the adversarial attacks, it is clear that the proposed RaCNN is more robust to adversarial examples than the vanilla classifier is. The proposed local mixup improves the robustness further, especially when the number of retrieved examples is small, i.e., $K = 5$. We conjecture that this is due to the quadratically increasing number of pairs, i.e., $\frac{K(K-1)}{2}$, for which local mixup must take care of, with respect to K.

S2 (Retrieval Attack) In Table 1, we present the accuracy of both the baseline and RaCNN's with varying strengths of white-box attacks, when the feature extractor ϕ' for the retrieval engine is attacked. We observe that it is indeed possible to fool the proposed RaCNN by attacking the retrieval process. Comparing Fig. 1 and Table 1, we however notice that the performance degradation is much less severe in this second scenario.

6.2. SVHN

We use the same architecture and hyper-parameter setting as in the CIFAR-10 experiments.

S1 (Direct Attack) On SVHN, we observe a similar trend from CIFAR-10. The proposed RaCNN is more robust against all the adversarial attacks compared to the vanilla convolutional network. Similarly to CIFAR-10, the proposed approach is most robust to DeepFool and Boundary, while it is most susceptible to L-BFGS. We however notice that the impact of local mixup is larger with SVHN than was with CIFAR-10.

Another noticeable difference is the impact of the number of retrieved examples on the classification accuracy. In the case of CIFAR-10, the accuracies on the clean test examples (the first column in Table 1) between using 5 and 10 retrieved examples differ significantly, while it is much less so with SVHN (the first column in Table 2.) We conjecture that this is due to a lower level of variation in input examples in SVHN, which are pictures of house numbers taken from streets, compared to those in CIFAR-10, which are pictures of general objects.

S2 (Retrieval Attack) We observe a similar trend between CIFAR-10 and SVHN, when the feature extractor ϕ' for retrieval was attacked, as shown in Tables 1–2.

6.3. ImageNet

We use ResNet-18 [14]. We pretrain it as a standalone classifier on ImageNet and use the feature extractor part ϕ' for retrieval. We use the same feature extractor $\phi = \phi'$ for the RaCNN without updating it. The classifier g is initialized with g' and tuned during training. In the case of ImageNet, we only try $K = 10$ retrieved examples with local mixup. Due to the high computational cost of the L-BFGS and Boundary attacks, we evaluate both the vanilla classifier and RaCNN against these two attacks on 200 images drawn uniformly at random from the validation set. We use Accuracy@5 which is a standard metric with ImageNet.

S1 (Direct Attack) A general trend with ImageNet is similar to that with either CIFAR-10 or SVHN, as can be seen in Fig. 3. The proposed RaCNN is more robust to adversarial attacks. We however do observe some differences. First, iFGSM is better at compromising both the baseline and RaCNN than L-BFGS is, in this case. Second, DeepFool is much more successful at fooling the baseline convolutional network on ImageNet than on the other two datasets, but is much less so at fooling the proposed RaCNN.

S2 (Retrieval Attack) Unlike CIFAR-10 and SVHN, we have observed that the retrieval attack is sometimes more effective than the direct attack in the case of ImageNet. For instance, FGSM can compromise the retrieval feature extractor ϕ' to decrease the accuracy from 77.68 down to 0.20 at $\overline{L_2} = 10^{-4}$. We observed a similar behavior with DeepFool, but not with iFGSM.

6.4. CW and Randomization: S1 – Direct Attack

As the CW attack and its variants are the state-of-the-art attacks available to date [5], we consider it separately from the other attacks. We pick the proposed RaCNN with $K = 10$ and local mixup and evaluate the randomization strategy's robustness to this attack. On CIFAR-10 and SVHN, we use the second-most-likely prediction as a target and the strength $\kappa = 0$. On the other hand, we use the sixth most-likely prediction and try both $\kappa = 0$ and 10 with ImageNet, as we report Accuracy@5.

As demonstrated in Fig. 4, the CW attack is effective at compromising the baseline. On CIFAR-10 and SVHN, CW is further effective at compromising the proposed RaCNN, although randomized inference greatly improves the robustness. We however observe an opposite trend on ImageNet, where randomized inference suffers when the strength of the CW attack is increased. Nevertheless, we observe that the proposed approach is more robust than the baseline is.

6.5. Comparison with other SOTA defense approaches

In Table 4, we compare RaCNN with other best published defense approaches. We adopt another strong attack, the projected gradient descent (PGD), as our benchmarking attack protocol. We can see from these results that all the

	Clean	FGSM			iFGSM			DeepFool		
$\overline{L_2}$	0	2e-04	4e-04	8e-04	2e-05	8e-05	2e-04	2e-05	8e-05	2e-04
Baseline	**95.48**	42.09	30.95	21.61	70.41	35.53	11.17	51.10	16.00	4.28
RC-K5	90.78	64.87	53.31	39.44	90.73	75.80	63.41	84.62	81.30	80.55
RC-K5-MU	91.64	68.31	57.20	**43.73**	91.55	77.74	**65.75**	86.18	83.20	82.43
RC-K10	92.19	64.94	52.24	37.73	92.10	76.41	62.70	86.18	84.25	82.21
RC-K10-MU	92.49	**68.72**	**57.30**	43.49	**92.45**	**78.26**	65.50	**87.33**	**84.73**	**84.10**

Table 2. The SVHN classifiers' robustness to the adversarial attacks in the S2 (Retrieval Attack)

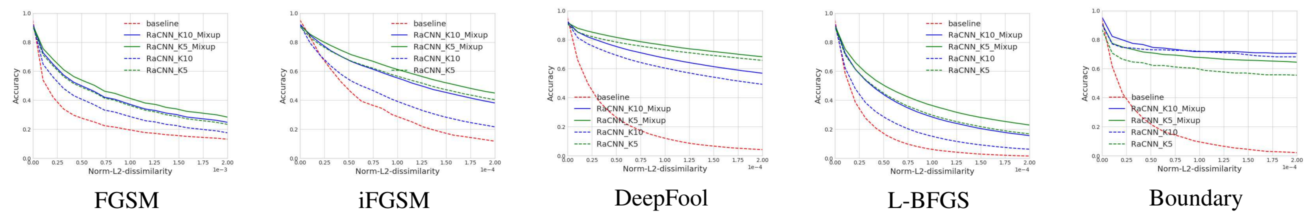


Figure 2. The SVHN classifiers' robustness to the adversarial attacks in the Scenario 1 (Direct Attack). The x-axis indicates the strength of attack in terms of the normalized L_2 distance. The y-axis corresponds to the accuracy.

reported defense methods have degraded the clean classification performance to some extent.

Although RaCNN does not outperform the SOTA *reactive* approaches proposed by [12], we argue that RaCNN as a *proactive* defense approach can easily be combined with any *reactive* defense method. We want to leave this combination to future work.

6.6. Summary Discussion

We have observed that the proposed RaCNN, when trained with the local mixup, is more robust to adversarial attacks, at least those seven considered in the experiments, than the vanilla convolutional network. More specifically, the RaCNN was most robust to the black-box, decision-based attack, while it was more easily compromised by white-box attacks, especially by the CW and L-BFGS attacks. This suggests that the RaCNN could be an attractive alternative to the vanilla convolutional network when deployed , for instance, in a cloud-based environment.

In Fig. 5, we show retrieval results given a query image from ImageNet. Although adversarial attack altered the retrieval engine's behavior, we see that the semantics of the query image could still be maintained in retrieved images, suggesting two insights. First, the robustness of the RaCNN is largely due to the robustness of the retrieval engine to small perturbation. Even when the retrieval quality degrades, a majority of retrieved examples are of a similar class. Second, we could further improve the robustness by designing the feature extractor ϕ' more carefully. For instance, an identity function $\phi'(x) = x$ would correspond to retrieval based on the raw pixels, which would make the retrieval engine robust to any adversarial attack imperceptible to humans. This may however results in a lower accuracy on clean examples, which is a trade-off that needs to be determined per task.

The robustness of the proposed RaCNN comes at the expense of the generalization performance on clean input examples. We have observed however that this degradation could be controlled at the expense of computational overhead by varying the number of retrieved examples per input. This controllability could be an important feature when deploying such a model in production.

7. Conclusion

In this paper, we proposed a novel retrieval-augmented convolutional network classifier (RaCNN) that integrates an off-the-shelf retrieval engine to counter adversarial attacks. The RaCNN was designed to tackle both off- and on-manifold adversarial examples, and to do so, we use a retrieval engine to locally characterize the data manifold as a feature-space convex hull and the attention mechanism to project the input onto this convex hull. The entire model, composed of the retrieval engine and a deep convolutional network, is trained jointly by the novel local mixup learning strategy which encourages the classifier to behave linearly within the feature-space convex hull.

We have evaluated the proposed approach on three standard benchmarks–CIFAR-10, SVHN and ImageNet–against six white-box attacks and one black-box, decision-based attack. The experiments have revealed that the proposed approach is more robust than the vanilla convolutional network in all the cases. The RaCNN was found to be especially robust to the black-box, decision-based attack.

The proposed approach consists of three major components; (1) local characterization of data manifold, (2) data manifold projection and (3) regularized learning on the

	Clean	FGSM			iFGSM			DeepFool		
$\overline{L_2}$	0	1e-04	2e-04	4e-04	1e-05	2e-05	4e-05	1e-05	2e-05	4e-05
Baseline	**88.98**	15	13.12	11.65	9.59	3.57	1.82	0.29	0.17	0.16
RC-K10-MU	77.68	**20.17**	**17.40**	**14.70**	**77.28**	**64.97**	**17.67**	**35.74**	**35.72**	**35.71**

Table 3. The ImageNet classifiers' robustness to the adversarial attacks in the S2 (Retrieval Attack).

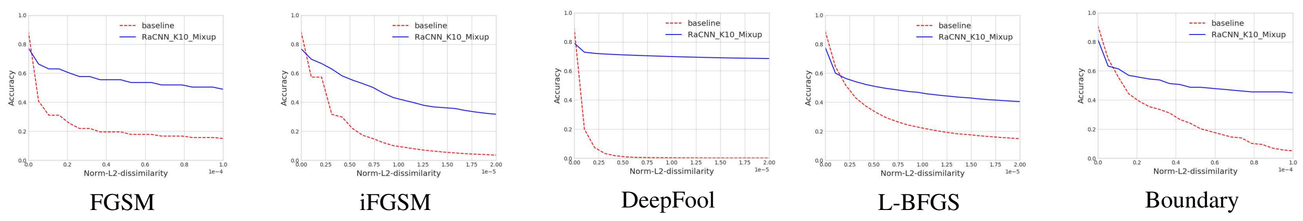


Figure 3. The ImageNet classifiers' robustness to the adversarial attacks in the Scenario 1 (Direct Attack). The x-axis indicates the strength of attack in terms of the normalized L_2 distance. The y-axis corresponds to the accuracy. The adversary utilizes top-5 accuracies for attacks.

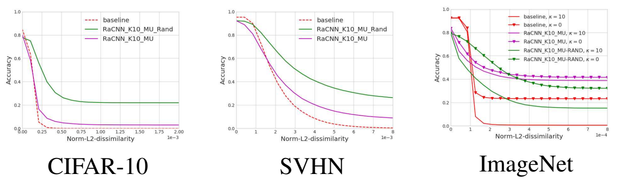
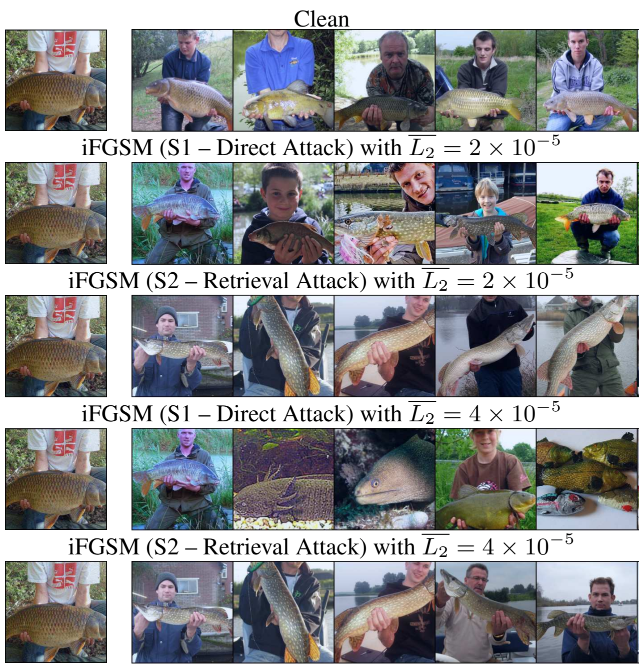


Figure 4. The robustness of the classifiers to the CW attack under the Scenario 1 (Direct Attack). The x-axis indicates the strength of attack in terms of the normalized L_2 distance. The y-axis corresponds to the accuracy. "_Rand" indicates that inference was done with the randomization strategy.

Name	Clean Acc	S1 / Gray-box Acc
Vanilla CNN	69.5	3.1
Crop Ensemble [12]	63.2	41.5
TV Minimization [12]	60.9	32.5
Image Quilting [12]	39.8	35.4
Adversarial Logit Pairing [17]	52.8	24.5
RaCNN	60.2	25.9
RaCNN + Rand	60.1	28.4

Table 4. The comparison of RaCNN with other SOTA defense approaches. The results are obtained subject to $\frac{\|x-\tilde{x}\|_2}{\|x\|_2} \leq 0.06$.


Clean
iFGSM (S1 – Direct Attack) with $\overline{L_2} = 2 \times 10^{-5}$
iFGSM (S2 – Retrieval Attack) with $\overline{L_2} = 2 \times 10^{-5}$
iFGSM (S1 – Direct Attack) with $\overline{L_2} = 4 \times 10^{-5}$
iFGSM (S2 – Retrieval Attack) with $\overline{L_2} = 4 \times 10^{-5}$
Figure 5. The left-most column of each panel shows the query image, and the next five images have been retrieved by F. We show the retrieval results using the original image (Clean) and the adversarial images one panel at a time. Even with noise largely enough to fool any vanilla convolutional network, the retrieval engine changes largely maintains the semantics of the query image.

manifold. There is a large room for improvement in each of these components. For instance, a feature-space convex hull may be replaced with a more sophisticated kernel estimator. Projection onto the convex hull could be done better, and a better learning algorithm could further improve the robustness against on-manifold adversarial examples. We leave these possibilities as future work.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 2

[2] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai. Better mixing via deep representations. In *Proceedings of the 30th*

International Conference on Machine Learning (ICML-13), 2013. 2

[3] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *KDD*. ACM, 2001. 3

[4] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017. 3

[5] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017. 2, 3, 5

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, 2004. 3

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2009. 1

[8] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001. 3

[9] J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018. 1, 2

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, 2014. 3

[11] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 3

[12] C. Guo, M. Rana, M. Cisse, and L. van der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017. 4, 6, 7

[13] C. Guo, M. Rana, M. Cisse, and L. van der Maaten. Countering adversarial images using input transformations. https://github.com/facebookresearch/adversarial_image_defenses, 2017. [Online]. 4

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. 3, 5

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 2015. 4

[16] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017. 3

[17] H. Kannan, A. Kurakin, and I. Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018. 7

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[19] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[20] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, 2009. 1

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012. 3

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. 1

[23] X. Li, J. Zhang, and C. Zong. One sentence one model for neural machine translation. *arXiv preprint arXiv:1609.06490*, 2016. 4

[24] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015. 2

[25] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 3

[26] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011. 1

[27] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 2

[28] J. Rauber and W. Brendel. Foolbox. http://foolbox.readthedocs.io/en/latest/, 2017. [Online; Revision 2d468cb6]. 4

[29] J. Rauber, W. Brendel, and M. Bethge. Foolbox v0. 8.0: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017. 4

[30] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 1992. 3

[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 1986. 2

[32] R. Shin and D. Song. Jpeg-resistant adversarial images. In *MAchine LEarning and Computer Security Workshop*, 2017. 3

[33] N. A. Smith and R. W. Tromble. Sampling uniformly from the unit simplex. *Johns Hopkins University, Tech. Rep*, 2004. 3

[34] P. Sprechmann, S. Jayakumar, J. Rae, A. Pritzel, A. P. Badia, B. Uria, O. Vinyals, D. Hassabis, R. Pascanu, and C. Blundell. Memory-based parameter adaptation. *International Conference on Learning Representations*, 2018. 4

[35] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1, 3

[36] P. Tabacof and E. Valle. Exploring the space of adversarial images. In *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 2016. 3

[37] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, 2016. 3

[38] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 2010. 3

[39] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 1, 2

[40] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 1997. 3