Appendix for Adversarial Eigen Attack on Black-Box Models

1. Proof

1.1. Proof for Theorem 1

Theorem 1 *The optimal solutions for problem given by (P1) and (P2) are that $\delta_1, \delta_2, \cdots, \delta_m$ are just the eigenvectors corresponding to the top-m eigenvalues of $J^T J$.*

Proof 1 *For the first optimization problem given by (P1), as $J^T J$ is a real symmetric matrix, considering the eigenvalue decomposition $J^T J = U \Sigma U^T$. Hence, we have $||J\delta_1||_2^2 = \delta_1^T J^T J \delta_1 = \delta_1^T U \Sigma U^T \delta_1$. Let $q = U^T \delta_1$, the original optimization problem could be written as:*

$$\max_q \ q^T \Sigma q = \sum_{k=1}^{m} \lambda_k q_k^2, \qquad s.t. \qquad \sum_{k=1}^{m} q_k^2 \leq 1$$

As $\sum \lambda_k q_k^2 \leq \lambda_1 \cdot \sum q_k^2 \leq \lambda_1$, and the condition of equality is reached when $q = [1, 0, \cdots, 0]^T$. Therefore, easy to show that the unique solution for δ_1 is given by the first column of U. Using the similar techique, and noticing that $\delta_j^T J^T J \delta_i = \delta_j^T \cdot \lambda_i \delta_i = 0$ when δ_i and δ_j are two different eigenvectors of $J^T J$. The constraint of the second recursive problems is satisfied.

1.2. Proof for Theorem 2

Theorem 2 *(Property of Eigen Perturbations) Assume there is no prior information about the gradient of $\tilde{g}$ (the direction of the actual gradient is uniformly distributed on the surface of an m-dimensional ball with unit radius). Given a query budget K for each iteration, the perturbations $\vec{l_1}, \vec{l_2}, \cdots, \vec{l_K}$ on representation space and the corresponding perturbations $\delta_1, \delta_2, \cdots, \delta_K$ on input space solved by Problem 5 (in original paper) is most efficient among any choice of exploring K orthogonal perturbation vectors on the representation space. Specifically, the final one-step gradient for $\nabla_z[\tilde{g}(z;\tilde{\theta})_y]$ is estimated by:*

$$\nabla_z[\tilde{g}(z;\tilde{\theta})_y] = \sum_{i=1}^{K} \left(\left. \frac{\partial \tilde{g}(z;\tilde{\theta})_y}{\partial \vec{l_i}} \right|_z \cdot \vec{l_i} \right)$$

and the expected change of the output probability $dp_F(y|x)$ reaches the largest with the same l_2-norm of perturbation on input space for all cases.

Proof 2 *Consider the relative change of $p_{\tilde{g} \circ h}(y|x)$ with respect to x:*

$$dy = dx^T \cdot \nabla_x[F(x;\theta)_y] = dx^T \cdot (J_h(x)^T \nabla_z[\tilde{g}(z;\tilde{\theta})_y])$$

For simplicity, define $J = J_h(x)$, and $\tilde{g}(z;\tilde{\theta})_y = Q\beta$, where $Q = [q_1, q_2, \cdots, q_K]$, a group of orthogonal basis with unit length, and β is $K \times 1$ vector representing K directional derivatives on Q. Thus we have:

$$dy = dx^T J^T Q \beta$$

To maximize $dy/||dx||_2$, we set $dx = \eta J^T Q \beta$ and then:

$$max_{||dx||_2 \leq \epsilon} \frac{dy}{||dx||} = \frac{\eta \beta^T Q^T J J^T Q \beta}{||\eta J^T Q \beta||_2} = ||J^T Q \beta||_2$$

As there is no prior information about the direction of the gradient of $\tilde{g}$, β could be viewed as a random vector with a fixed length uniformly distributed on the surface of an m-dimensional ball. Thus, the optimization problem is converted to:

$$max_Q \ \mathbb{E}_\beta[\beta^T Q^T J J^T Q \beta]$$

where Q is column orthogonal.

Next, consider the space formed by all eigenvectors of JJ^T, i.e. $l_1, l_2, \cdots, l_m$, sorted by eigenvalues in descending order. Define $L = [l_1, l_2, \cdots, l_m]^T$. As Q is formed by choosing K columns from a certain orthogonal matrix, L is also an orthogonal matrix, immediately we have $Q = L^T C$ for certain $m \times K$ matrix C. Noticing that $JJ^T l_i = \lambda_i l_i$, hence,

$$\mathbb{E}_\beta[\beta^T Q^T J J^T Q \beta] = \mathbb{E}_\beta[\beta^T C^T L J J^T L^T C \beta] = \mathbb{E}_\beta[\beta^T C^T \Sigma C \beta]$$

where $\Sigma = diag(\lambda_1, \lambda_2, \cdots, \lambda_m)$.

The final step is to demonstrate $C = I_{m \times K}$ will be optimal. Consider a new random vector $\gamma = C\beta$, we have:

$$\begin{aligned}
\mathbb{E}_\beta[\beta^T C^T \Sigma C \beta] &= \mathbb{E}_\beta[\gamma^T \Sigma \gamma] \\
&= \mathbb{E}_\beta \left[\sum_{j=1}^{m} \lambda_j \gamma_j^2 \right] \\
&= \mathbb{E}_u \left\{ \mathbb{E}_\beta \left[\sum_{j=1}^{m} \lambda_j \gamma_j^2 \middle| \sum_{i=1}^{K} \beta_i^2 = u \right] \right\} \\
&= \mathbb{E}_u \left\{ \sum_{j=1}^{m} \lambda_j \mathbb{E}_\beta \left[\gamma_j^2 \middle| \sum_{i=1}^{K} \beta_i^2 = u \right] \right\} \\
&\leq \mathbb{E}_u \left\{ \sum_{j=1}^{K} \lambda_j \mathbb{E}_\beta \left[\beta_j^2 \middle| \sum_{i=1}^{K} \beta_i^2 = u \right] \right\} \\
&= \mathbb{E}_\beta \left[\sum_{j=1}^{K} \lambda_j \beta_j^2 \right] = \mathbb{E}_\beta[\beta^T I_{m \times K}^T \Sigma I_{m \times K} \beta]
\end{aligned} \tag{1}$$

The key step is the inequality from 4th row to 5th row, which is not obvious. To demonstrate this, we first notice that

$$\sum_{j=1}^{m} \mathbb{E}_\beta \left[\gamma_j^2 \middle| \sum_{i=1}^{K} \beta_i^2 = u \right] = \sum_{j=1}^{K} \mathbb{E}_\beta \left[\beta_j^2 \middle| \sum_{i=1}^{K} \beta_i^2 = u \right] = u \tag{2}$$

This is because C is an orthogonal transformation. Also, we have for $j = 1, 2, \cdots, K$:

$$\mathbb{E}_\beta \left[\gamma_j^2 \middle| \sum_{i=1}^{K} \beta_i^2 = u \right] \leq \mathbb{E}_\beta \left[\beta_j^2 \middle| \sum_{i=1}^{K} \beta_i^2 = u \right] \tag{3}$$

To simplify the notation, we directly use $\mathbb{E}_\beta[\gamma_j^2]$ and $\mathbb{E}_\beta[\beta_j^2]$. We prove this conclusion as follows:

$$\begin{aligned}
\mathbb{E}_\beta[\gamma_j^2] &= \mathbb{E}_\beta[(c_j^T \beta)^2] = \mathbb{E}_\beta[((c_{j,S} + c_{j,S^\perp})^T \beta)^2] \\
&= \mathbb{E}_\beta[((c_{j,S}^T \beta)^2] = ||c_{j,S}||_2^2 \mathbb{E}_\beta[\beta_j^2] \leq \mathbb{E}_\beta[\beta_j^2]
\end{aligned}$$

The idea is that, let S be the subspace constructed by top-K columns of L, we decompose the jth column of the orthogonal transformation C, i.e. c_j as $c_{j,S} \in S$ and $c_{j,S^\perp} \in S^\perp$. $c_{j,S^\perp}$ has no contribution to the expectation, and only $c_{j,S}$ contributes to the expectation. Due to the symmetry characteristic of β on S, we have $\mathbb{E}_\beta[((c_{j,S}^T \beta)^2] = ||c_{j,S}||_2^2 \mathbb{E}_\beta[\beta_j^2]$. And as the maximum length of c_j is 1, the conclusion is proved.

By Equation 2 and 3, the inequality in 1 is obvious for the eigenvalues λ is sorted in descending order.

We further note that $l_1, l_2, \cdots, l_K$ and the $\delta_1, \delta_2, \cdots, \delta_K$ are just the top-K eigenvectors of JJ^T and $J^T J$, completing the proof.

2. Implementation Details

2.1. Taking Advantage of Image Continuity

In practical experiments, the operation of truncated SVD is time-consuming. Consider a network h with input size $n = H \times W \times C$ and the output size m (*i.e.* dimension of representation), the Jacobian matrix will be $m \times n$, and the complexity of SVD operation is $O(m^2 n)$, which may be slow when the image size is large.

A simple way to decrease the complexity is to aggregate adjacent pixels together, taking advantage of image continuity. Specifically, suppose the image size is $H_0 \times W_0$, and we would like to decrease the input to $H_1 \times W_1$. First, we define the scale to be $s_h = \lfloor H_0/H_1 \rfloor$ and $s_w = \lfloor W_0/W_1 \rfloor$. Then, while processing EigenBA, we only change the pixel in the center area $(s_h \cdot H_1) \times (s_w \cdot W_1)$ of the original image.

The forward propagation remains unchanged, where the input is still $H_1 \times W_1$. As to the backward propagation, we only need to calculate the Jacobian matrix of representation z with respect to the center area $(s_h \cdot H_1) \times (s_w \cdot W_1)$. And then, we process average pooling with scale (s_h, s_w) and stride (s_h, s_w) for each row of Jacobian matrix J. (It is noteworthy that each row of J is an $(s_h \cdot H_1 \cdot s_w \cdot W_1)$ vector, hence, before average pooling operation we need to restore the vector to $(s_h \cdot H_1) \times (s_w \cdot W_1)$ matrix.) Finally, the new Jacobian matrix J' should be a $m \times (H_1 \times W_1)$ matrix. After processing SVD to J', the right singular vector should be a $H_1 \cdot W_1$ vector, which represents for the perturbation, naming δ'. Finally, the actual δ related to the center area $(s_h \cdot H_1) \times (s_w \cdot W_1)$ pixels should be obtained by applying nearest upsampling method to δ' with scale (s_h, s_w).

Through this simple method, we reduce the computation of SVD to about $1/(s_h \cdot s_w)$ of the original method.

2.2. Rounding Technique

For all experiments, we follow the setting of SimBA and ParsiBA for fair comparison, where the input pixel could be any real number on $[0, 1]$. However, in practical use, the value of pixel is discrete in image classfication. In this section, we introduce a simple rounding technique.

The method is rather simple: after each renewal, for each pixel value v, we find the integer N, such that $N/255 \leq v < (N+1)/255$. Then, we round the value v to the nearer one, either $N/255$ or $(N+1)/255$. Table 1 shows the difference whether using the rounding technique corresponding to our experiment on attacking ResNet-50 trained on ImageNet.

The results show that, the rounding technique will only slightly increase average l_2 and average query numbers, which is acceptable in practical use.

2.3. Hyperparameters

In this section, we mainly describe the hyperparameters for all settings.

For attack on ResNet-50 trained on ImageNet, the maximum query number of each attacked image is limited to 10,000. For SimBA, the stepsize of gradient, α is set to 0.2. For SimBA-DCT, the stepsize is set to 0.2, the dimensionality of 2D frequency space is set to 28 and the stride for block order is set to 7. For ParsiBA, the maximum l_∞ norm is set to 0.01. For Trans-FGSM , the stepsize is set to 0.4 for untargeted attack and 0.3 for targeted attack. For Trans-FGM, the step size is set to 0.4 for both cases. For EigenBA, the stepsize is set to 0.4 for both cases, we also decrease the dimension of Jacobian matrix from $512 \times (224 * 224 * 3)$ to $512 \times (112 * 112 * 3)$ by using the method described in Appendix 2.1. For processing SVD once, we extract top 100 right singular vectors.

For attack on Inception-v3 trained on ImageNet, the size of original image is $299 * 299$, which is different from the input of the pre-trained model (ResNet-18), our solution is to crop the center $224 * 224$ pixedls as the input of the pre-trained model. Also, the purturbations are limited to these $224 * 224$ pixels, keeping the other pixedls unchanged during attacking. The hyperparameters are the same as attacking ResNet-50 trained on ImageNet except for SimBA-DCT, we expand the dimensionality of 2D frequency to 38 as original paper of SimBA.

For attack on ResNet-18 trained on Cifar-10, the maximum query number of each attacked image is limited to 2,000. For SimBA, the stepsize of gradient, α is set to 0.04. For SimBA-DCT, the stepsize is set to 0.04, the dimensionality of 2D frequency space is set to 32 (the same to original image size, which is optimal value in Cifar-10 experiment). For Trans-FGSM the stepsize is set to 0.08, and for Trans-FGM, the stepsize is set to 0.06. For EigenBA, the stepsize is set to 0.08 for both untargeted attack and targeted attack, the dimension of Jacobian matrix is $512 \times (32 * 32 * 3)$, we do not use method in Appendix 2.1 for Cifar-10 experiment. For processing SVD once, we extract top 50 right singular vectors.

For attack on ResNet-50 trained on WebVision, the maximum query number of each attacked image is limited to 5,000. For SimBA, the stepsize of gradient, α is set to 0.2. For SimBA-DCT, the stepsize is set to 0.4, the dimensionality of 2D frequency space is set to 28 and the stride for block order is set to 7. For Trans-FGSM the stepsize is set to 0.5, and for Trans-FGM,

Table 1. Untargeted attack on ResNet-50 trained on ImageNet, with or without rounding technique.

Methods	Avg. queries (success)	Avg. queries (all)	Success Rate	Avg. l_2
EigenBA (No rounding)	383	518	0.986	3.622
EigenBA (Rounding)	503	617	0.988	3.797

the stepsize is set to 0.4. For EigenBA, the stepsize is set to 0.4, we also decrease the dimension of Jacobian matrix from $512 \times (224 * 224 * 3)$ to $512 \times (112 * 112 * 3)$ by using the method described in Appendix 2.1. For processing SVD once, we extract top 100 right singular vectors.

For ablation study on Cifar-10 in Section 4.4 in original paper, the stepsize for reserve rate 1.0, 0.9, 0.8, 0.7, 0.6, 0.5 experiment is 0.08, 0.07, 0.06, 0.05, 0.04, 0.03.

2.4. Complexity Analysis

We run all experiments on a single Ubuntu 16.04 server, with two 6-core 12-thread CPU Intel Xeon E5-2630. Totally there are 24 threads. We use a single Nvidia Tesla P40 GPU.

The bottleneck of our algorithm is the SVD operation, which is an $O(m^2 n)$ algorithm, as Section Appendix 2.1 shows. For experiment on ImageNet, we decrease the Jacobian matrix to $m = 512, n = 112 * 112 * 3 = 37,632$. And processing SVD once costs about 2 seconds.

We use a batchsize of 5. For each loop starting with an SVD operation and ending with a number of perturbation renewals, at the beginning of each batch the SVD operation will take about $2 * 5 = 10$ seconds, the several steps of perturbation renewals (between 100 to 200 steps) cost about 5 seconds. With the number of unsuccessful attack samples decreasing in each batch, the time for SVD operation will decrease. For untargeted attack on ImageNet, EigenBA will cost about 16 hours to finish 1,000 attacked images, and for targeted attack on ImageNet, the execution time will be about 100 hours for 1,000 attacked images.

To compare the time complexity of our algorithm to baselines fairly, we test the execution time of different algorithms on attacking ResNet-50 trained on WebVision dataset in a closed test environment. The result in shown in Table 2, including attacking 1,000 images. From the results, our EigenBA algorithm is comparable in time complexity to baselines.

Table 2. Execution time for attacking 1,000 images from WebVision dataset.

Methods	SimBA	SimBA-DCT	Trans-FGSM	Trans-FGM	EigenBA (Ours)
Time	6h15min	9h49min	15h15min	14h31min	19h6min

3. Visualization of the Results

We randomly choose 4 attacked images from all attacked images for experiment on attacking ResNet-50 trained on ImageNet. Figure 1 visualizes the original images and the adversarial images generated by our EigenBA. From Figure 1, the difference between the original image and the adversarial image is barely visible to the naked eye. However, the adversarial image is incorrectly classified by the deep neural network, which demonstrates the advantages of our algorithm. One interesting finding is that, in the setting of untargeted attack, the misclassified label tends to be similar to the original label in semantic meaning, for example, although the staffordshire bullterrier is different with the bull mastiff, they are both under the meta class of Dog.

4. Pseudocode of SimBA Algorithm

Figure 1. Showcases of untargeted attack on ImageNet.

Algorithm 1 The SimBA Algorithm for untargeted attack

PROCEDURE SimBA($\mathbf{x}, y, Q, \epsilon$): // Q represents for a group of orthogonal basis
$\delta = \mathbf{0}$
$\mathbf{p} = p_h(y \mid \mathbf{x})$
while $\mathbf{p}_y = \max_{y'} \mathbf{p}_{y'}$ **do**
  Pick randomly without replacement: $\mathbf{q} \in Q$
  for $\alpha \in \{\epsilon, -\epsilon\}$ **do**
    $\mathbf{p}' = p_h(y \mid \mathbf{x} + \delta + \alpha\mathbf{q})$
    if $\mathbf{p}'_y < \mathbf{p}_y$ **then**
      $\delta = \delta + \alpha\mathbf{q}$
      $\mathbf{p} = \mathbf{p}'$
      break
    end if
  end for
end while
Return δ