

Interpreting Attributions and Interactions of Adversarial Attacks

Xin Wang^{a*}, Shuyun Lin^{a*}, Hao Zhang^a, Yufei Zhu^a, Quanshi Zhang^{a†}
^aShanghai Jiao Tong University

Abstract

This paper aims to explain adversarial attacks in terms of how adversarial perturbations contribute to the attacking task. We estimate attributions of different image regions to the decrease of the attacking cost based on the Shapley value. We define and quantify interactions among adversarial perturbation pixels, and decompose the entire perturbation map into relatively independent perturbation components. The decomposition of the perturbation map shows that adversarially-trained DNNs have more perturbation components in the foreground than normally-trained DNNs. Moreover, compared to the normally-trained DNN, the adversarially-trained DNN have more components which mainly decrease the score of the true category. Above analyses provide new insights into the understanding of adversarial attacks.

1. Introduction

Deep neural networks (DNNs) have shown promise in various tasks, such as image classification [37] and speech recognition [15]. Adversarial robustness of DNNs has received increasing attention in recent years. Previous studies mainly focused on attacking algorithms [48, 4, 28], the detection of adversarial examples for the adversarial defense [29, 12, 26], and adversarial training to learn DNNs robust to adversarial attacks [14, 28].

Unlike previous studies of designing more powerful attacks or learning more robust DNNs, in this research, we aim to explain the signal-processing behavior behind the adversarial attack, *i.e.* how pixel-wise perturbations cooperate with each other to achieve the attack. We develop new methods to explain adversarial attacks from the following perspectives.

1. Given an input image, the regional attribution to the adversarial attack is computed to diagnose the importance of each image region to the decrease of the attacking

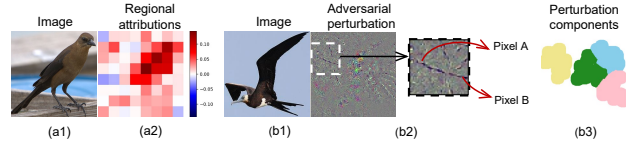


Figure 1. (a) Regional attributions to the adversarial attack. Regions with high attributions are important for the decrease of the attacking cost. (b2) Perturbation pixels A and B interact with each other and form a curve to conduct the adversarial attack; (b3) the entire perturbation can be decomposed into several components. Perturbation pixels within each component have strong interactions, whereas perturbation pixels between different components have relatively weak interactions.

cost, *i.e.* the L_p norm of the adversarial perturbation. As Fig. 1 (a2) shows, regions of the bird’s head and neck have high attributions to the adversarial attack. If these two regions are not allowed to be perturbed, then magnitudes of adversarial perturbations in other regions would be significantly increased for attacking. In this way, the attacking cost may increase significantly.

The regional attribution (importance) provides a new perspective to understand adversarial attacks. We compute such regional attributions as Shapley values *w.r.t.* the attacking cost.

2. Pixel-wise interactions & perturbation components in the adversarial attack: Given a perturbation map of the input image, we further define and quantify interactions among pixel-wise perturbations in the perturbation map, termed *perturbation pixels*. *I.e.* we aim to explore how perturbation pixels cooperate to achieve the attack. According to [46], the adversarial power of a single pixel mainly depends on the context around the pixel, rather than rely on each perturbation pixel independently. For instance, in Fig. 1 (b2), perturbation pixels A and B do not directly contribute to the attack. Instead, they interact with each other to form a curve to fool the DNN.

The interaction among perturbation pixels can be defined based on the game theory. Given a DNN g trained for classification and an adversarial image $x' = x + \delta \in \mathbb{R}^n$, $y = g(x') \in \mathbb{R}$ denotes the scalar output of the DNN (or one dimension of the vectorized network output). Let ϕ_i denote the importance (attribution) of the i -th perturbation pixel of

*Equal contribution

†Quanshi Zhang is the corresponding author. He is with the John Hopcroft Center and the MoE Key Lab of Artificial Intelligence, AI Institute, at the Shanghai Jiao Tong University, China.

δ w.r.t. the output y , which is implemented as the Shapley value. The attribution values of all perturbation pixels satisfy $g(x') - g(x) = \sum_{i=1}^n \phi_i$. Let ϕ_S denote the overall importance of all perturbation pixels in S , when perturbation pixels in S collaborate with each other. Then, the interaction is defined as the change of the importance of S , when we ignore the collaboration between perturbation pixels and simply sum up the importance of each individual-working perturbation pixel in S , i.e. $\phi_S - \sum_{i \in S} \phi_i$ quantifies the interaction. If $\phi_S - \sum_{i \in S} \phi_i > 0$, it indicates that perturbation pixels in S cooperate with each other, and exhibit positive interactions. If $\phi_S - \sum_{i \in S} \phi_i < 0$, it indicates that perturbation pixels in S conflict with each other, and exhibit negative interactions.

Furthermore, based on the pixel-wise interactions among perturbation pixels, as Fig. 1 (b3) shows, we can decompose the effect of the adversarial attack into several *perturbation components*, which provides a new perspective to analyze how perturbation pixels cooperate with each other. To this end, we develop a method to extract groups of perturbation pixels with strong interactions as perturbation components. Perturbation pixels in the same component have strong interactions with each other. Whereas, perturbation pixels in different components have relatively weak interactions.

Using the Shapely value for explanation and its advantages: We define the regional attribution and interactions among perturbation pixels based on the Shapley value [40]. Though explanation methods in previous studies, such as Grad-CAM [39] and GBP [45], can measure the importance of input elements, the Shapley value is proved to be the unique attribution satisfying four desirable properties, i.e. the linearity property, the dummy property, the symmetry property, and the efficiency property [30]. The four properties can be considered the solid theoretic support for the Shapley value. The scientific rigor of the Shapley value makes the attribution analysis and the interaction defined on the Shapley value more trustworthy than other explanation methods. Please see section A in supplementary materials for more discussion.

We have analyzed regional attributions and pixel-wise interactions on different DNNs. The analysis of regional attributions has demonstrated that important image regions for the L_2 attack and those for the L_∞ attack were similar, although L_2 perturbations and L_∞ perturbations were significantly dissimilar. Furthermore, our research has provided new insights into adversarial perturbations by investigating the property of perturbation components.

- Our research has provided a game-theoretic view to explain and verify the phenomenon that adversarially-trained DNNs mainly focus on foreground. For adversarially-trained DNNs, adversarial perturbations are more likely to interact with each other on the foreground.
- Moreover, the adversarial-trained DNN usually had more

components punishing the true category and less components encouraging incorrect categories than the normally-trained DNN.

In fact, our research group led by Dr. Quanshi Zhang has proposed game-theoretic interactions, including interactions of different orders [58] and multivariate interactions [60]. As a basic metric, the interaction can be used to learn baseline values of Shapley values [36] and to explain signal processing in trained DNNs from different perspectives. For example, we have used interactions to build up a tree to explain the hierarchical interactions between words in NLP models [57] and to explain the generalization power of DNNs [59]. The interaction can also explain adversarial transferability [54] and adversarial robustness [35]. As an extension of the system of game-theoretic interactions, in this study, we interpret attributions and interactions of adversarial attacks.

However, the computational cost of the Shapley value is NP-hard, which makes the decomposition of perturbation components is also NP-hard. Thus, we develop an efficient approximation approach to the decomposition problem. Our method has been applied to DNNs with various architectures for different tasks. Preliminary experiments have demonstrated the effectiveness of our method.

Contributions: This study provides new perspectives to understand adversarial attacks, which includes the regional attribution to the adversarial attack and the extraction of perturbation components. We have applied our methods to various benchmark DNNs and datasets. Experimental results provide an insightful understanding of adversarial attacks.

2. Related work

Adversarial attacks and defense: Attacking methods can be roughly divided into two categories, i.e. white-box attacks [48, 4, 14, 22, 28, 33, 46] and black-box attacks [32, 24, 9, 7]. Various methods have been proposed to defend adversarial attacks. Defensive distillation [34] uses knowledge distillation [17] to improve the adversarial robustness of DNNs. Some studies focus on the methods of detecting adversarial examples [12, 29, 26, 49], which can reject adversarial examples in order to protect DNNs. Besides the detection of adversarial examples, some methods are proposed to directly learn robust DNNs. Adversarial training methods have been proposed to train DNNs resistant to adversarial attacks [50, 28, 61], which use adversarial examples as training samples during the training process.

The explanation of adversarial examples: has received increasing attention in recent years. Tsipras *et al.* [52] showed an inherent trade-off between the standard accuracy and adversarial robustness, and found that compared with normally trained DNNs, the adversarial trained DNN tended to be more interpretable. Furthermore, Etman *et al.* theoretically and empirically demonstrated that more

robust DNNs exhibited more interpretable saliency maps. Some studies demonstrated that the existence of adversarial examples was attributed to finite-sample overfitting [3], the presence of well-generalizing but non-robust features in datasets [18], and the geometry property of the high dimensional data [13]. Wen *et al.* [55] proposed the CLEVER score as the lower bound guarantee of the robustness of DNNs. Adversarial saliency map [33] computes the importance of each pixel to the network prediction. Xu *et al.* [56] and Fan *et al.* [11] proposed to generate structured and sparse perturbations to better understand adversarial examples. Unlike previous studies of explaining why the adversarial example exists, our work focuses on the explanation of adversarial examples from perspectives of which region of the input image is important to the attacking cost, and how the adversarial perturbation pixels interact with each other during attacking.

• **Difference between the Shapley-based attribution and the adversarial saliency map [33]:** The gradient-based explanations [41], including the adversarial saliency map [33], represents the marginal attacking utility *w.r.t.* a specific context. In comparison, [30] has proved that the Shapley value is computed on all potential contexts, which ensures more fairness than the gradient-based explanation, and makes the Shapley value satisfy the aforementioned properties. Please see section A in supplementary materials for details.

The interaction has been widely investigated in the field of statistics [2, 23]. Sorokina *et al.* [8] defined the *K-way* interaction for additive models. Lundberg *et al.* [38] quantified interactions between features for tree ensemble methods. Some studies mainly focus on interactions for DNNs. Murdoch *et al.* [31] proposed to disambiguate interactions in LSTMs, and Singh *et al.* [44] extended this method to generic DNNs. Jin *et al.* [20] quantified the contextual independence of words for DNNs in NLP tasks. Tsang *et al.* [51] detected statistical interactions based on neural network weights. Janizek *et al.* [19] extended the method of Integrated Gradients [47] to quantify pairwise interactions of input features. In comparison, in this study, we apply a different type of interactions between perturbation pixels based on Shapley values, in order to extract perturbation components.

The Shapley value is proposed in the game theory [40]. Considering multiple players in a game, each player aims to win a high reward. Some players choose to form a coalition in order to let the coalition win a reward, higher than the sum of rewards of those players when they play individually. The Shapley value is considered as a unique and unbiased approach to fairly allocating the total reward gained by all players to each player [27], which satisfies four desirable properties, *i.e.* *linearity*, *dummy*, *symmetry*, and *efficiency* [30], which will be introduced later. Let $N = \{1, 2, \dots, n\}$ denote the set of all players, and let $r(\cdot)$ denote the reward function. Let us consider a set of players

S , which does not include the player i , *i.e.* $S \subseteq N \setminus \{i\}$. $r(S)$ represents the reward gained by players in S , *i.e.* the reward obtained when only players in S participate in the game. When player i joins the game, the overall reward changes to $r(S \cup \{i\})$. The difference of the reward, *i.e.* $r(S \cup \{i\}) - r(S)$, is considered as the marginal contribution of player i to the reward. The Shapley value $\phi^r(i)$ is formulated as the weighted sum of marginal contributions of player i brought by all possible $S \subseteq N \setminus \{i\}$.

$$\phi^r(i) = \frac{1}{|N|} \sum_{S \subseteq N \setminus \{i\}} \binom{|N|-1}{|S|} (r(S \cup \{i\}) - r(S)) \quad (1)$$

where $\binom{|N|-1}{|S|}$ is the number of all combinations of $|S|$ players among the set without the player i . Note that the Shapley value is the unique function that satisfies all the following desirable properties [30]:

- **Linearity property:** Let there be two games and the corresponding score functions are r and w , *i.e.* $r(S)$ and $w(S)$ measure the score obtained by players in S in these two games. If these two games are combined into a new game, and the score function becomes $r + w$, then the Shapley value comes to be $\phi^{r+w}(i) = \phi^r(i) + \phi^w(i)$ for each player.

- **Dummy property:** A player $i \in N$ is referred to as a dummy player if $r(S \cup \{i\}) = r(S) + r(\{i\})$ for any $S \subseteq N \setminus \{i\}$. In this way, $\phi^r(i) = r(\{i\}) - r(\emptyset)$, which means that player i plays the game independently.

- **Symmetry property:** If $r(S \cup \{i\}) = r(S \cup \{j\})$ holds for any subset $S \subseteq N \setminus \{i, j\}$, then Shapley values of player i and j are equal, *i.e.* $\phi^r(i) = \phi^r(j)$.

- **Efficiency property:** The sum of each player's Shapley value is equal to the score won by the coalition N , *i.e.* $\sum_{i=1}^n \phi^r(i) = r(N) - r(\emptyset)$. This property guarantees the overall score can be allocated to each player in the game.

3. Algorithm

We first introduce basic concepts of adversarial attacks. Let $x \in [0, 1]^n$ denote the input image, and a DNN is learned to predict the class label $c(x) \in \{1, 2, \dots, T\}$. To simplify the story, in this study, we mainly analyze the targeted adversarial attack. The goal is to add a human imperceptible perturbation $\delta \in \mathbb{R}^n$ on x to get a new image $x' = x + \delta$, which makes the DNN mistakenly classify x' as the target category $t \neq c(x)$. The objective of the targeted adversarial attack is defined by [4] as follows.

$$\begin{aligned} \min_{\delta} \|\delta\|_p \quad \text{s.t. } c(x + \delta) = t, x + \delta \in [0, 1]^n \\ \xrightarrow{\text{relax}} \min_{\delta} \|\delta\|_p + \lambda \cdot f(x + \delta) \quad \text{s.t. } x + \delta \in [0, 1]^n \end{aligned} \quad (2)$$

where the value of $f(x) : [0, 1]^n \rightarrow \mathbb{R}$ measures the correctness of the classification, and λ is a scalar constant. For example, in [48] f is defined as the cross entropy loss. In [4], f is chosen as $f(x) = \max\{\max_{i \neq t} Z_i - Z_t, -\text{threshold}\}$, where Z is the output of the DNN before the softmax layer.

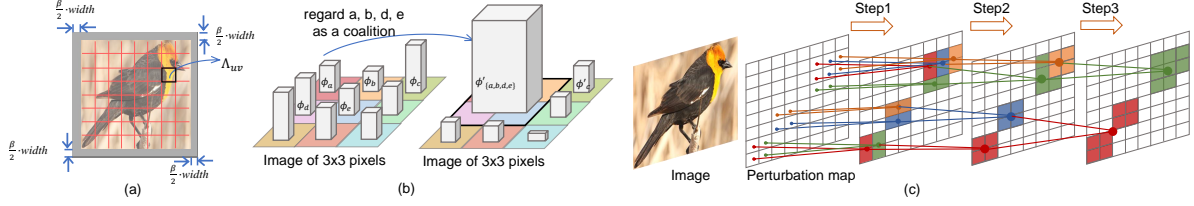


Figure 2. (a) Images are divided into $L \times L$ grids and extended to $(1 + \beta) \cdot width \times (1 + \beta) \cdot height$ to compute regional attributions. (b) A toy example to illustrate the interaction among pixels. If pixels a, b, d, e form a coalition and act as a singleton player, then the total reward of a, b, d, e increases. The additional reward indicates the interaction among a, b, d, e . (c) A toy example to illustrate the extraction of perturbation components.

3.1. Regional attributions to the adversarial attack

Given an input image, the regional attribution measures the importance of each image region to the decrease of the attacking cost $\|\delta\|_p$. [27] has discussed that the commonly used gradient *w.r.t.* the attacking loss [41] cannot objectively reflect the regional attribution due to the highly non-linear representation of the DNN. Please see section A in supplementary materials for more discussions. Considering the high computational burden, we investigate the regional attribution instead of the pixel-wise attribution. We divide the entire image into $L \times L$ grids (regions), denoted by $\Lambda = \{\Lambda_{11}, \Lambda_{12}, \dots, \Lambda_{LL}\}$. Each region $\Lambda_{uv} \in \Lambda$ ($1 \leq u, v \leq L$) is a set of pixels. ϕ_{uv} denotes the attribution of the region Λ_{uv} . The total attribution to the decrease of the attacking cost is allocated to each region:

$$\phi_{11} + \phi_{12} + \dots + \phi_{LL} = \text{cost}(\Lambda) - \text{cost}(\emptyset) \quad (3)$$

where $\text{cost}(\Lambda) = \|\delta^{(\Lambda)}\|_p$ and $\text{cost}(\emptyset) = \|\delta^{(\emptyset)}\|_p$. $\delta^{(\Lambda)}$ represents the adversarial perturbation generated by allowing all pixels to be perturbed. $\delta^{(\emptyset)}$ represents the adversarial perturbation generated without perturbing any pixels in Λ . Note that it is impossible to conduct adversarial attacks, when all pixels in the image are not allowed to be perturbed. Thus, as Fig. 2 (a) shows, we approximate ϕ_{uv} by extending the input image to $(1 + \beta) \cdot width \times (1 + \beta) \cdot height$, where β is a small scalar constant. We regard $\delta^{(\emptyset)}$ as the adversarial perturbation generated when only the extended regions are perturbed. We compute such regional attribution as the Shapley value [40].

$$\phi_{uv} = \frac{1}{L^2} \sum_{S \subseteq \Lambda \setminus \{\Lambda_{uv}\}} \binom{L^2 - 1}{|S|}^{-1} [\text{cost}(S \cup \{\Lambda_{uv}\}) - \text{cost}(S)] \quad (4)$$

where S denotes a set of regions excluding Λ_{uv} . The $\text{cost}(S)$ is the L_p norm ($p = 2$, or $+\infty$) of the adversarial perturbation generated when only regions in S are allowed to be perturbed during attacking. We formulate such an attack using masking operation, $\hat{\delta} = \arg \min_{\delta} \|\delta \circ M^{(S)}\|_p + c \cdot f(x + \delta \circ M^{(S)})$ s.t. $x + \delta \circ M^{(S)} \in [0, 1]^n$, where \circ represents the element-wise multiplication. $M^{(S)}$ is a mask, which satisfies $\forall \Lambda_{uv} \in S, \forall i \in \Lambda_{uv}, M_i^{(S)} = 1$, and for all other pixels $i, M_i^{(S)} = 0$. $\delta^{(S)} = \hat{\delta} \circ M^{(S)}$ is the adversarial perturbation generated when only regions in S are allowed to be perturbed, thereby $\text{cost}(S) = \|\delta^{(S)}\|_p$.

Based on Equation (4), in order to quantify the regional attribution, we sample different $M^{(S)}$ to conduct adversarial attacks. Note that the adversarial attack with masking operation is conducted to compute $\text{cost}(S)$ and $\text{cost}(S \cup \{\Lambda_{uv}\})$, instead of obtaining more robust perturbations.

3.2. Interactions in the attack and the decomposition of perturbation components

In this section, given a perturbation map of an input image, we aim to decompose the perturbation map into several relatively independent image regions (perturbation components), in order to explore the true pixel-wise collaborative behavior behind of the adversarial attack. Note that the collaborative behavior itself may not be adversarial robust. Given the set of all perturbation pixels Ω , we aim to extract m components, *i.e.* $\{C^{(1)}, C^{(2)}, \dots, C^{(m)}\}$, where $\forall i, C^{(i)} \subseteq \Omega; \forall i, j, i \neq j, C^{(i)} \cap C^{(j)} = \emptyset$, and $C^{(1)} \cup C^{(2)} \cup \dots \cup C^{(m)} \subseteq \Omega$. Perturbation pixels within each component are supposed to have strong interactions, while perturbation pixels in different components are supposed to have weak interactions. Thus, each component can be roughly considered independent in the adversarial attack.

The interaction among perturbation pixels reflect cooperative or adversarial relationships of perturbation pixels in attacking. Inspired by [30], we define interactions based on the game theory. Let us consider a game with n players $\Omega' = \{1, 2, \dots, n\}$, where players aim to gain a reward. We use $2^{\Omega'}$ to denote all potential subsets of Ω' . For example, if $\Omega' = \{a, b\}$, then $2^{\Omega'} = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$. The reward of the game $z : 2^{\Omega'} \rightarrow \mathbb{R}$ maps a set of players to a scalar value. Here, given an adversarial example $x' = x + \delta \in \mathbb{R}^n$ and a DNN $g(\cdot)^1 : \mathbb{R}^n \rightarrow \mathbb{R}^T$, where T is the number of categories, we regard each perturbation pixel in δ as a player. The goal of perturbation pixels is to decrease the score of the true category $g_\ell(x')$ and increase the score of the target category $g_t(x')$. Given a set of perturbation pixels $S \subseteq \Omega$, we formulate the reward of perturbation pixels in S as $z(S) = g_t(x + \delta \circ M^{(S)}) - g_\ell(x + \delta \circ M^{(S)})$, by assigning values of perturbation pixels in $\Omega \setminus S$ to zero, where $M^{(S)}$ is a mask $\forall i \in S, M_i^{(S)} = 1; \forall i \in \Omega \setminus S, M_i^{(S)} = 0$.

¹ $g(\cdot)$ denotes the DNN's output scores before the softmax layer.

The total reward in the game is $\Phi = z(\Omega) - z(\emptyset)$, where $z(\Omega) = g_t(x + \delta \circ \mathbf{1}) - g_\ell(x + \delta \circ \mathbf{1})$ is the reward obtained by all perturbation pixels, and $z(\emptyset) = g_t(x + \delta \circ \mathbf{0}) - g_\ell(x + \delta \circ \mathbf{0})$ is the baseline reward *w.r.t.* the original image. The total reward can be allocated to each perturbation pixel, $\Phi = \phi_1 + \phi_2 + \dots + \phi_n$ as the Shapley value. ϕ_i is referred to as the reward of the perturbation pixel i .

We notice that perturbation pixels do not contribute to the attack independently. Instead, some perturbation pixels may form a specific component. In this way, the reward gained by the component is different from the sum of the reward of each perturbation pixel when they contribute to the attack individually. Here, we can consider this component as a singleton player. Thus, the total reward Φ is allocated to $n - |S| + 1$ players in the new game, *i.e.* $\Phi = \phi'_S + \sum_{i \in N \setminus S} \phi'_i$. In this way, the interaction among players in S is defined as

$$I[S] = \phi'_S - \sum_{i \in S} \phi_i \quad (5)$$

where ϕ'_S is the allocated reward when S is taken as a singleton player, and ϕ_i is the reward when we consider the perturbation pixel i contributes to the attack independently.

We compute ϕ_i and ϕ'_S as Shapley values, which will be given in Equation (6). Fig. 2 (b) shows a toy example for the interaction. The total reward is allocated to each perturbation pixel, *i.e.* $\Phi = \phi_a + \phi_b + \dots + \phi_h + \phi_i$. If pixels a, b, d, e are regarded as a singleton player, then the allocation of Φ would change to $\Phi = \phi'_{\{a, b, d, e\}} + \phi'_c + \dots + \phi'_h + \phi'_i$. If $\phi'_{\{a, b, d, e\}} - (\phi_a + \phi_b + \phi_d + \phi_e) \neq 0$, then pixels a, b, d, e are considered to have interactions.

Understanding of the interaction: If $I[S] > 0$, it means perturbation pixels in S cooperate with each other to change prediction scores of the DNN. If $I[S] < 0$, it means perturbation pixels in S conflict with each other. The absolute value $|I[S]|$ indicates the strength of the interaction.

Computation of the reward: According to Equation (1), ϕ_i and ϕ'_S are computed as follows.

$$\begin{aligned} \phi_i &= \frac{1}{n} \sum_{\tilde{S} \subseteq \Omega \setminus \{i\}} \binom{n-1}{|\tilde{S}|}^{-1} [z(\tilde{S} \cup \{i\}) - z(\tilde{S})] \\ \phi'_S &= \frac{1}{n'} \sum_{\tilde{S} \subseteq \Omega \setminus S} \binom{n'-1}{|\tilde{S}|}^{-1} [z(\tilde{S} \cup S) - z(\tilde{S})] \end{aligned} \quad (6)$$

where $n' = n - |S| + 1$. In this study, we propose an efficient method to approximate ϕ_i and ϕ'_S , which will be introduced in Equation (7) and Equation (8).

Extraction of perturbation components: We extract perturbation components via hierarchical clustering, in which perturbation pixels have strong interactions. The pseudo code of extracting perturbation components is shown in section C in supplementary materials. In the first step of clustering, we merge q neighboring perturbation pixels with strong interactions into a q -pixel compo-

nent. In the second step, we merge q neighboring components with strong interactions into a component of q^2 pixels. In this way, we iteratively generate components of q , q^2 , q^3 pixels, *etc.* A toy example is shown in Fig. 2 (c). We use $C^{(i)}$ to denote a component. The finally merged component is selected from a set of component candidates, each of which is a set of q neighboring components, $S^c = C^{(i_1)} \cup C^{(i_2)} \cup \dots \cup C^{(i_q)}$. If $|I[S^c]| > \gamma$, then we merge $C^{(i_1)}, C^{(i_2)}, \dots, C^{(i_q)}$ into a large component. Considering the local property [6], we only compute interactions among neighboring pixels/components.

Efficient approximation of rewards of perturbation pixels: We can consider the value of each perturbation pixel i as the sum of values of K sub-pixels, denoted by (i, k) , $1 \leq k \leq K$. The values of the K sub-pixels are uniformly divided, *i.e.* $\delta/K = \delta_{(i,1)} = \delta_{(i,2)} = \dots = \delta_{(i,K)}$. In this way, the reward ϕ_i can be approximated as the sum of sub-pixels' rewards, *i.e.* $\phi_i \approx \sum_{k=1}^K \phi_{(i,k)}$. Consider the symmetry property of the Shapley value [30], $\phi_{(i,1)} = \phi_{(i,2)} = \dots = \phi_{(i,K)}$. Thus, we can approximate $\phi_i \approx K \cdot \phi_{(i,k)}$. Please see section B.1 in supplementary material for the proof and more discussions. The reward $\phi_{(i,k)}$ can be approximated as follows. Let us consider the marginal reward for computing the Shapley value of the sub-pixel, *i.e.* $z(S \cup \{(i, k)\}) - z(S)$. Given a large value of K , the perturbation magnitude of each sub-pixel $|\delta_{(i,k)}|$ is fairly small. The marginal reward can be approximated as $z(S \cup \{(i, k)\}) - z(S) = \delta_{(i,k)} \cdot \partial z(S) / \partial \delta_{(i,k)} + o(\delta_{(i,k)})$, based on the Taylor expansion. In this way, the Shapley value of each sub-pixel is given as Equation (7).

$$\phi_{(i,k)} \approx \frac{1}{nK} \sum_{S \subseteq \Omega^{\text{pixel}} \setminus \{(i,k)\}} \binom{nK-1}{|S|}^{-1} \underbrace{\left[\frac{\partial z(S)}{\partial \delta_{(i,k)}} \delta_{(i,k)} \right]}_{\text{approximates } z(S \cup \{(i,k)\}) - z(S)} \quad (7)$$

where $\Omega^{\text{pixel}} = \{(i, k) | 1 \leq i \leq n, 1 \leq k \leq K\}$.

Efficient approximation of rewards of components: Let there be m components in a certain clustering step, *i.e.* $\{C^{(1)}, C^{(2)}, \dots, C^{(m)}\}$. Given a component $C^{(u)}$, we approximate $\phi_{C^{(u)}}$ using combinations of components, instead of perturbation pixels. We further reduce the cost of calculating $\phi_{C^{(u)}}$ in the similar way of calculating ϕ_i . Each perturbation pixel i in $C^{(u)}$ is divided into K sub-pixels, *i.e.* $\forall i \in C^{(u)}, \delta_i = \sum_k \delta_{(i,k)}$. In this way, $C^{(u)}$ can be divided into K sub-components. The k -th sub-component is given as $C_k^{(u)} = \bigcup_{i \in C^{(u)}} \{(i, k)\}$. The reward of $C_k^{(u)}$ is approximated as follows. Please see section B.3 in supplementary materials for the proof.

$$\phi_{C_k^{(u)}} \approx \frac{1}{mK} \sum_{S \subseteq \Omega^{\text{comp}} \setminus \{C_k^{(u)}\}} \binom{mK-1}{|S|}^{-1} A \quad (8)$$

where $\Omega^{\text{comp}} = \{C_k^{(u)} | 1 \leq u \leq m, 1 \leq k \leq K\}$, and $A = \sum_{(i,k) \in C_k^{(u)}} \left[\frac{\partial z(S)}{\partial \delta_{(i,k)}} \delta_{(i,k)} \right]$, which approximates $z(S \cup \{C^{(u)}\}) - z(S)$.

Implementation & computational complexity: The computation of Shapley values is NP-hard. The sampling-based method has been widely used for approximation [5]. Thus, we propose to approximate Equation (7) and Equation (8) by a sampling method. The complexity of computing the Shapley value of one pixel is $O(2^n)$. Equation (9) reduces the computational complexity to $O(nKT)$, where n is the pixel number, and T is times of sampling. Because derivatives to sub-pixels *w.r.t.* all pixels $i \in \Omega$ can be computed simultaneously, the computational complexity of computing Shapley values of all pixels remains $O(nKT)$.

$$\begin{aligned} \phi_{(i,k)} &\approx \frac{1}{nKT} \sum_{s=0}^{nK-1} \sum_{t=1}^T \delta_{(i,k)} \left. \frac{\partial z(S)}{\partial \delta_{(i,k)}} \right|_{S=S_{st}^{\text{pixel}}} \text{ s.t. } |S_{st}^{\text{pixel}}| = s \\ \phi_{C_k^{(u)}} &\approx \frac{1}{mKT} \sum_{s=0}^{mK-1} \sum_{(i,k) \in C_k^{(u)}} \sum_{t=1}^T \delta_{(i,k)} \left. \frac{\partial z(S)}{\partial \delta_{(i,k)}} \right|_{S=S_{st}^{\text{comp}}} \quad (9) \\ \text{ s.t. } |S_{st}^{\text{comp}}| &= s \end{aligned}$$

where t represents a sampling step; s controls the size of the set S ; $S_{st}^{\text{pixel}} \subseteq \Omega^{\text{pixel}} \setminus \{(i, k)\}$ denotes a random subset of s sub-pixels excluding (i, k) ; $S_{st}^{\text{comp}} \subseteq \Omega^{\text{comp}} \setminus \{C_k^{(u)}\}$ denotes a random subset of s sub-components excluding $C_k^{(u)}$.

Even with above approximation, the computational cost for computing all component candidates in each step of clustering is still high. Thus, we further approximate rewards of component candidates by simplifying contextual relationships of far-away pixels [6]. We use $S^c = C^{(i_1)} \cup C^{(i_2)} \cup \dots \cup C^{(i_q)}$ to denote a component candidate. If we compute ϕ'_{S^c} in the set $\{S^c, C^{(i_{k+1})}, \dots, C^{(i_m)}\}$, the computational complexity is $O((m-q)KT)$. The complexity of computing rewards of all candidates is $O(m(m-q)KT)$. Here, instead of computing ϕ'_{S^c} in the set $\{S^c, C^{(i_{k+1})}, \dots, C^{(i_m)}\}$, we randomly merge \tilde{m} components to get \tilde{m}/q component candidates, including S^c . In this way, the new set includes \tilde{m}/q component candidates and $m - \tilde{m}$ components, *i.e.* $\{S^c, \bigcup_{a=q+1}^{2q} C^{(i_a)}, \dots, \bigcup_{a=\tilde{m}-q+1}^{\tilde{m}} C^{(i_a)}, C^{(i_{\tilde{m}+1})}, \dots, C^{(i_m)}\}$. We can simultaneously compute rewards of \tilde{m}/q candidates in the new set, and the computational complexity is $O((m - (q-1)\tilde{m}/q)KT)$. To compute rewards of all potential component candidates, we need to sample qm/\tilde{m} different sets. In this way, the overall complexity for the computation of rewards of candidates is reduced from $O(m(m-q)KT)$ to $O(m(qm/\tilde{m} - q)KT)$. Please see section B.4 supplementary materials for details.

4. Experiments

Datasets & DNNs: We tested our methods on tasks of coarse-grained image classification, fine-grained image classification, and face attribute estimation, using four benchmark datasets: the Pascal VOC 2012 dataset [10], the CUB200-2011 dataset [53], the Stanford Dog dataset [21], and the CelebA dataset [25]. For each of these datasets,

we used object images cropped by object bounding boxes for both training and testing. We analyzed regional attributions to the adversarial attack on three benchmark DNNs: AlexNet [42], VGG-16 [43] and ResNet-18 [16]. We computed interactions among perturbation pixels and extracted perturbation components on ResNet-18/34/50 [16].

Furthermore, to analyze the utility of adversarial training, we also extracted perturbation components on adversarially-trained ResNet-18/34/50 [28]. We conducted adversarial training on two datasets: the Pascal VOC 2012 dataset [10] and the CUB200-2011 dataset [53]. For adversarial training on the CUB200-2011 dataset, we only conducted the classification on 10 categories that were uniformly selected from the original 200 categories. For fair comparisons, the normally-trained DNN was also trained using the same 10 categories in the CUB200-2011 dataset. All DNNs were pre-trained on the ImageNet dataset [37], and then fine-tuned using these four datasets, respectively.

Implementation details: We used the C&W attack [4] as the L_2 attacker and the BIM [22] as the L_∞ attacker, which have been widely used. For adversarial attacks on all datasets and DNNs, we conducted the targeted attack. For the Pascal VOC 2012 dataset [10], we set the target class as the *bird*. For the CUB200-2011 dataset [53] and the Stanford Dogs dataset [21], we set the target class as the last category in each dataset, which were *the Common Yellowthroat* and *the African hunting dog*, respectively. For the CelebA dataset [25], we chose two global face attributes (*male*, *young*) and three local face attributes (*wearing glasses*, *smiling*, *wearing lipstick*), and set the target as the opposite attribute, for example, we conducted the adversarial attack on the CelebA dataset to make a DNN mistakenly predict a face image actually with *wearing glasses* to *not wearing glasses*. We set $\beta = 1/6$, $L = 8$ to compute regional attributions. We set $q = 4$ to compute interactions among perturbation pixels and further extract perturbation components.

4.1. Exp. 1: Regional attributions to the adversarial attack

Visualization of regional attributions: Fig. 3 visualizes adversarial perturbations and regional attributions. We compared magnitudes of regional adversarial perturbations with regional attributions. Given a region Λ_{uv} , the magnitude of the regional perturbation was computed as $(\sum_{i \in \Lambda_{uv}} |\delta_i|^2)^{1/2}$. Fig. 3 shows that attributions to the L_2 attack and magnitudes of regional perturbations were similar, but attributions to the L_∞ attack and magnitudes of regional perturbations were usually different. For example, let us focus on the top left girl image in Fig. 3, L_∞ perturbations uniformly distributed over the entire image, while attributions mainly focused on the face region. We have also compared regional attributions with different hyper-parameters

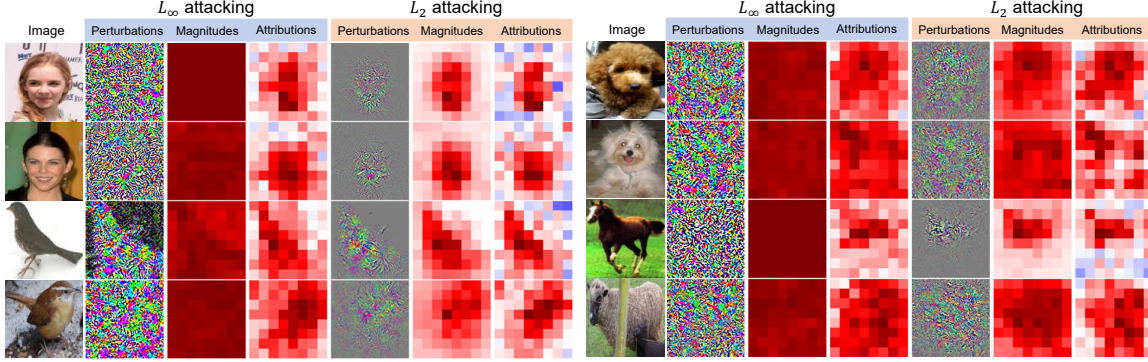


Figure 3. Comparison between adversarial perturbations and regional attributions. For attributions, dark red regions are important regions to the decrease of the attacking cost. Blue regions indicate negative attributions, *i.e.* these regions decrease the effect of attacking and increase the attacking cost. Regional attributions to the L_2 attack and magnitudes of regional perturbations were usually similar, while regional attributions to the L_∞ attack and magnitudes of regional perturbations were usually different.

	CelebA		CUB200-2011		Pascal VOC		Stanford Dogs	
	IoU(attributions)	IoU(magnitudes)	IoU(attributions)	IoU(magnitudes)	IoU(attributions)	IoU(magnitudes)	IoU(attributions)	IoU(magnitudes)
AlexNet	0.749 ± 0.027	0.668 ± 0.154	0.729 ± 0.032	0.526 ± 0.051	0.743 ± 0.046	0.577 ± 0.069	0.739 ± 0.024	0.567 ± 0.061
VGG-16	0.718 ± 0.038	0.534 ± 0.064	0.681 ± 0.057	0.543 ± 0.070	0.685 ± 0.073	0.474 ± 0.096	0.742 ± 0.045	0.488 ± 0.138
ResNet-18	0.762 ± 0.087	0.497 ± 0.190	0.713 ± 0.044	0.552 ± 0.064	0.718 ± 0.072	0.540 ± 0.158	0.707 ± 0.060	0.550 ± 0.110

Table 1. The IoU between regional attributions to the L_2 attack and regional attributions to the L_∞ attack, and the IoU between L_2 adversarial perturbations’ magnitudes and L_∞ adversarial perturbations’ magnitudes². Regional attributions to the L_2 attack and regional attributions to the L_∞ attack were similar, while regional magnitudes of L_2 adversarial perturbations and magnitudes of L_∞ adversarial perturbations were dissimilar.

(β and L), which shows that regional attributions were insensitive to the selection of hyper-parameters. Please see section D.1 supplementary materials for details.

Similarity between regional attributions through different attacks: As Fig. 3 shows, although the distribution of L_2 adversarial perturbations and the distribution of L_∞ adversarial perturbations were dissimilar, their regional attributions were similar to each other. Given regional attributions to the L_2 attack $\phi^{(2)} \in \mathbb{R}^{L \times L}$ and regional attributions to the L_∞ attack $\phi^{(\infty)} \in \mathbb{R}^{L \times L}$, we used the $IoU = \frac{\sum_u \sum_v \min(\phi_{uv}^{(2)}, \phi_{uv}^{(\infty)})}{\sum_u \sum_v \max(\phi_{uv}^{(2)}, \phi_{uv}^{(\infty)})}$ to measure the similarity between regional attributions $\phi^{(2)}$ and $\phi^{(\infty)}$, where we normalized the attribution $\phi'_{uv} = \frac{\phi_{uv} - \min_{u',v'} \phi_{u'v'}}{\max_{u',v'} \phi_{u'v'} - \min_{u',v'} \phi_{u'v'}}$. We also showed the IoU between L_2 magnitudes and L_∞ magnitudes. Please see Table 1 for quantitative results of similarity between regional attributions and similarity between magnitudes of regional perturbations. In Fig. 3, we found that regional attributions to the L_2 attack and regional attributions to the L_∞ attack were similar, while magnitudes of L_2 adversarial perturbations and regional magnitudes of L_∞ adversarial perturbations were dissimilar.

4.2. Exp. 2: Interactions in the attack and the decomposition of perturbation components

Extraction of perturbation components: We extracted interactions between perturbation pixels generated in the L_2 attack, and decomposed the perturbation into compo-

²The mean and standard deviation of IoU were computed on 10 samples.

ponents. To reduce the computation cost, we regarded each group of neighboring 4×4 pixels as a super-pixel. We set $\tilde{m} = 0.5 \cdot m$. In the first step of clustering, γ was set to satisfy $\mathbb{E}_S[\mathbb{1}(|I[S]| > \gamma)] = 0.2$, where $\mathbb{1}(\cdot)$ is the indicator function. In subsequent steps of clustering, γ was set to satisfy $\mathbb{E}_S[\mathbb{1}(|I[S]| > \gamma)] = 0.5$. To obtain stable results, when computing the reward of a component candidate S_c , we kept the nearest component of each component in S_c always present in sampling. In each step, we ended up merging components, when component candidates for which the reward had been computed covered more than 90% components. We used the greedy strategy to merge components, and kept conducting clustering until the size of each component was 64.

We first extracted perturbation components on normally-trained DNNs. Fig. 4 visualizes adversarial perturbations and corresponding perturbation components. Note that we enlarged the size of each super-pixel, to clarify the visualization. Fig. 4 shows that components were not aligned with visual concepts. For example, adversarial perturbations on the top left girl image in the first row of Fig. 4 was segmented into five components, in which the pink component covered the neck, hair, and other regions without semantic meanings.

Effects of adversarial training on perturbation components: We also extracted perturbation components on adversarially-trained DNNs, which are visualized in Fig. 4. We used the method proposed by Madry *et al.* [28] for adversarial training, which was formulated as $\min_{\theta} \mathbb{E}_{x_i \in X} \max_{x'_i: \|x'_i - x_i\|_p \leq \epsilon} \ell(g_{\theta}(x'_i), c_i)$, where c_i repre-

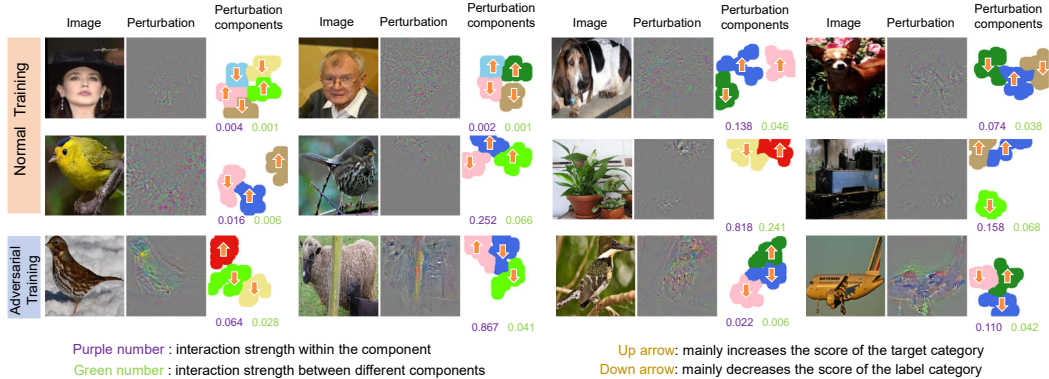


Figure 4. Visualization of perturbation components. Perturbation pixels within each component have strong interactions. Perturbation pixels between different components have relatively weak interactions. The color saturation of the component indicates the average reward (importance) of the perturbation component. Perturbation components were not always aligned with visual concepts.

	CUB200-2011		Pascal VOC	
	Normal	Adv-trained	Normal	Adv-trained
ResNet-18	63.8%	80.8%	55.6%	90.4%
Resnet-34	69.0%	78.6%	66.2%	85.7%
ResNet-50	62.6%	84.9%	60.8%	89.2%

Table 2. The ratio of perturbation components mainly in the foreground.

sents the label of x_i , and ℓ denotes the classification loss.

We investigated whether perturbation components were mainly localized in the foreground or the background. A perturbation component was regarded being in the foreground, if perturbation pixels in the component belonging to the foreground were more than perturbation pixels in the component belonging to the background. Table 2 reports the ratio of components in the foreground for adversarially-trained DNNs and normally-trained DNNs. Adversarially-trained DNNs had more perturbation components in the foreground than normally-trained DNNs.

Moreover, we also explored the utility of perturbation components, *i.e.* whether the component mainly decreased the prediction score of the true category or mainly increased the score of the target category. We classified perturbation components into two types, *i.e.* components mainly decreasing the prediction score of the true category and components mainly increasing the score of the target category. $\Delta y_\ell = |g_\ell(x + \delta) - g_\ell(x + \delta \circ M^{(\Omega \setminus C^{(u)})})|$ and $\Delta y_t = |g_t(x + \delta) - g_t(x + \delta \circ M^{(\Omega \setminus C^{(u)})})|$ measured the decrease of the prediction score of the true category and the increase of the prediction score of the target category caused by the component $C^{(u)}$, respectively. If $\Delta y_\ell > \Delta y_t$, the component $C^{(u)}$ was regarded mainly decreasing the score of the true category; otherwise, mainly increasing the score of the target category. Table 3 reports the ratio of components that mainly decreased the prediction score of the true category. The ratio of components mainly decreasing the score of the true category in adversarially-trained DNNs were usually greater than that in normally-trained DNNs.

	CUB200-2011		Pascal VOC	
	Normal	Adv-trained	Normal	Adv-trained
ResNet-18	62.8%	77.9%	34.9%	55.3%
ResNet-34	44.0%	68.4%	43.8%	63.1%
ResNet-50	58.2%	82.8%	31.6%	48.2%

Table 3. The ratio of perturbation components which mainly decrease the score of the true category.

5. Conclusion

In this paper, we have analyzed adversarial attacks from the attributional perspective. We have computed regional attributions to adversarial attacks. We have further defined and extract interactions among perturbation pixels decomposed the perturbation map into perturbation components based on interactions. We have found regional attributions to the L_2 attack and magnitudes of the L_2 perturbation were similar, while regional attributions to the L_∞ attack and magnitudes of the L_∞ perturbation were dissimilar. The extraction of perturbation components showed that perturbation components were not aligned with visual concepts. We have found that adversarially-trained DNNs had more perturbation components in the foreground than normally-trained DNNs. Moreover, compared to the normally-trained DNN, the adversarially-trained DNN was prone to decrease the score of the true category, instead of increasing the score of the target category. Our methods have been used to analyze different DNNs learned for the image classification and the face attribute estimation.

Acknowledgments

This work is partially supported by the National Nature Science Foundation of China (No. 61906120, U19B2043), Shanghai Natural Science Foundation (21ZR1434600), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), and Huawei Technologies Inc. Xin Wang is supported by Wu Wen Jun Honorary Doctoral Scholarship, AI Institute, Shanghai Jiao Tong University.

References

- [1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *arXiv preprint arXiv:1810.03292*, 2018.
- [2] Jacob Bien, Jonathan Taylor, and Robert Tibshirani. A lasso for hierarchical interactions. *In Annals of statistics*, 41(3):1111, 2013.
- [3] Sébastien Bubeck, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints. *In arXiv:1805.10204*, 2018.
- [4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *In IEEE Symposium on Security and Privacy*, 2017.
- [5] Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *In Computers & Operations Research*, 36(5):1726–1730, 2009.
- [6] Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. L-shapley and c-shapley: Efficient model interpretation for structured data. *In arXiv:1808.02610*, 2018.
- [7] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. *In arXiv:1708.03999*, 2017.
- [8] Mirek Riedewald Daria Sorokina, Rich Caruana. Detecting statistical interactions with additive groves of trees. *In ICML*, 2008.
- [9] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Xiaolin Hu, Jianguo Li, , and Jun Zhu. Boosting adversarial attacks with momentum. *In CVPR*, 2018.
- [10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
- [11] Yanbo Fan, Baoyuan Wu, Tuanhui Li, Yong Zhang, Mingyang Li, Zhifeng Li, and Yujiu Yang. Sparse adversarial attack via perturbation factorization. *In Proceedings of European Conference on Computer Vision*, 2020.
- [12] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. Detecting adversarial samples from artifacts. *In ICML*, 2017.
- [13] Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S. Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *In arXiv:1801.02774*, 2018.
- [14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *In arXiv:1412.6572*, 2014.
- [15] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. *In ICML*, 2014.
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *In CVPR*, 2016.
- [17] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *In NeurIPS Deep Learning Workshop*, 2014.
- [18] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *In NeurIPS*, 2019.
- [19] Joseph D Janizek, Pascal Sturmfels, and Su-In Lee. Explaining explanations: Axiomatic feature interactions for deep networks. *arXiv preprint arXiv:2002.04138*, 2020.
- [20] Xisen Jin, Zhongyu Wei, Junyi Du, Xiangyang Xue, and Xiang Ren. Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models. *In ICLR*, 2020.
- [21] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. *In CVPR Workshop on Fine-Grained Visual Categorization*, 2011.
- [22] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *In arXiv:1607.02533*, 2017.
- [23] Michael Lim and Trevor Hastie. Learning interactions via hierarchical group-lasso regularization. *In Journal of Computational and Graphical Statistics*, 24(3):627–654, 2015.
- [24] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *In arXiv: 1611.02770*, 2016.
- [25] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. *In ICCV*, 2015.
- [26] Jiajun Lu, Theerasit Issaranon, and David Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. *In ICCV*, 2017.
- [27] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *In NeurIPS*, 2017.
- [28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *In ICLR*, 2018.
- [29] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *In ICLR*, 2017.
- [30] Grabisch Michel and Roubens Marc. An axiomatic approach to the concept of interaction among players in cooperative games. *In International Journal of Game Theory*, 1999.
- [31] W. James Murdoch, Peter J. Liu, and Bin Yu. Beyond word importance: Contextual decomposition to extract interactions from lstms. *In ICLR*, 2018.
- [32] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. *In arXiv:1602.02697*, 2017.
- [33] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *In IEEE European Symposium on Security & Privacy*, 2016.
- [34] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *In arXiv:1511.04508*, 2016.
- [35] Jie Ren, Die Zhang, Yisen Wang, Lu Chen, Zhanpeng Zhou, Xu Cheng, Xin Wang, Yiting Chen, Jie Shi, and Quan-

- shi Zhang. Game-theoretic understanding of adversarially learned features. *arXiv preprint arXiv:2103.07364*, 2021.
- [36] Jie Ren, Zhanpeng Zhou, Qirui Chen, and Quanshi Zhang. Learning baseline values for shapley values. *arXiv preprint arXiv:2105.10719*, 2021.
- [37] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *In International Journal of Computer Vision*, 115(3):211–252, 2015.
- [38] Su-In Lee Scott Lundberg. Consistent feature attribution for tree ensembles. *In ICML WHI Workshop*, 2017.
- [39] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *In Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.
- [40] Lloyd S Shapley. A value for n-person games. *In Contributions to the Theory of Games*, 2(28):307–317, 1953.
- [41] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
- [42] Karen Simonyan and Andrew Zisserman. Imagenet classification with deep convolutional neural networks. *In NeurIPS*, 2012.
- [43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *In ICLR*, 2015.
- [44] Chandan Singh, W. James Murdoch, and Bin Yu. Hierarchical interpretations for neural network predictions. *In ICLR*, 2019.
- [45] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.
- [46] Jiawei Su, Danilo Vasconcellos Vargas, and Sakurai Kouichi. One pixel attack for fooling deep neural networks. *In arXiv:1710.08864*, 2017.
- [47] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *In International Conference on Machine Learning*, pages 3319–3328, 2017.
- [48] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *In ICLR*, 2014.
- [49] Guan hong Tao, Shiqing Ma, and Xiangyu Zhang Yingqi Liu. Attacks meet interpretability: Attribute-steered detection of adversarial samples. *In NeurIPS*, 2018.
- [50] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *In ICLR*, 2018.
- [51] Michael Tsang, Dehua Cheng, and Yan Liu. Detecting statistical interactions from neural network weights. *In ICLR*, 2018.
- [52] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *In ICLR*, 2019.
- [53] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.
- [54] Xin Wang, Jie Ren, Shuyun Lin, Xiangming Zhu, Yisen Wang, and Quanshi Zhang. A unified approach to interpreting and boosting adversarial transferability. *In International Conference on Learning Representations*, 2021.
- [55] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. *In ICLR*, 2018.
- [56] Kaidi Xu, Sijia Liu, Pu Zhao, Pin-Yu Chen, Huan Zhang, Quanfu Fan, Deniz Erdogmus, Yanzhi Wang, and Xue Lin. Structured adversarial attack: Towards general implementation and better interpretability. *In International Conference on Learning Representations*, 2019.
- [57] Die Zhang, Huilin Zhou, Hao Zhang, Xiaoyi Bao, Da Huo, Ruizhao Chen, Xu Cheng, Mengyue Wu, and Quanshi Zhang. Building interpretable interaction trees for deep nlp models. *In AAAI*, 2021.
- [58] Hao Zhang, Xu Cheng, Yiting Chen, and Quanshi Zhang. Game-theoretic interactions of different orders. *arXiv preprint arXiv:2010.14978*, 2020.
- [59] Hao Zhang, Sen Li, Yinchao Ma, Mingjie Li, Yichen Xie, and Quanshi Zhang. Interpreting and boosting dropout from a game-theoretic view. *In ICLR*, 2021.
- [60] Hao Zhang, Yichen Xie, Longjie Zheng, Die Zhang, and Quanshi Zhang. Interpreting multivariate interactions in dnns. *In AAAI*, 2021.
- [61] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. *In CVPR*, 2016.

A. Comparisons of Shapley-based attributions and other explanation methods

We define regional attributions and interactions between perturbation pixels based on Shapley values [40]. We compare Shapley-based attributions with other explanation methods from the following perspectives.

- **Theoretical rigor.** A good attribution method must satisfy certain desirable properties. The Shapley value has been proved to be the unique attribution that satisfies four desirable properties, *i.e.* the linearity property, the dummy property, the symmetry property, and the efficiency property [30]. In comparison, some explanation methods like Grad-CAM [39] and GBP [45] do not have theoretic supports for the correctness of these methods.
- **Objectivity.** The attribution of one input element depends on contexts of neighboring pixels. The Shapley value considers all possible contexts to compute the attribution of an input unit, which ensures the objectiveness of the attribution. In contrast, some attention methods, such as the adversarial saliency map [33], only consider the marginal gradient, which is biased to a specific context from this perspective.
- **Trustworthiness.** The theoretic foundation in game theory makes the Shapley values trustworthy. In contrast, some seemingly transparent explanation methods simply do not have clear theoretical support, which hurts the trustworthiness of the explanation. Actually, [1] has shown some explanation methods like GBP [45] can not reflect the true attribution.
- **Broad applicability.** The Shapley values can be extended to measure interactions between two input elements. [57] has proved the theoretical foundation and advantages for defining interactions using the Shapley value in game theory [30]. However, some gradient-based explanation methods assume the model is locally linear, which fails to measure interactions between two input elements.

B. Details of efficient approximation of interactions

We approximate Shapley values to enable efficient computation of interactions.

B.1. Approximation of attributions of perturbation pixels

In Section Algorithm, we introduce the approximation of the Shapley value of a perturbation pixel. In the supplementary material, we give more discussions about the approximation.

The adversarial perturbation is denoted as $\delta \in \mathbb{R}^n$. Each perturbation pixel i is divided into K sub-pixels with equal values, *i.e.* $\delta_i = \delta_{(i,1)} + \delta_{(i,2)} + \dots + \delta_{(i,K)}$ and $\delta_{(i,1)} = \delta_{(i,2)} = \dots = \delta_{(i,K)}$. Instead of directly computing the attribution of each perturbation pixel, we compute the attribution of each sub-pixel. The attribution of the sub-pixel can be efficiently approximated based on the Taylor expansion, which will be discussed later.

Among sub-pixels $(i, 1), (i, 2) \dots (i, K)$, each sub-pixel plays the same role in attacking, thereby $\phi_{(i,1)} = \phi_{(i,2)} = \dots = \phi_{(i,K)}$, which is proved as follows. The attribution of each sub-pixel (i, k) is formulated as the Shapley value. The Shapley value satisfies the four axioms (linearity axiom, dummy axiom, symmetry axiom, and efficiency axiom). According to the symmetry axiom, given two sub-pixels (i, k) and (j, k') , if $z(S \cup \{(i, k)\}) = z(S \cup \{(j, k')\})$ holds for any set $S \subseteq \Omega^{\text{pixel}} \setminus \{(i, k), (j, k')\}$, then $\phi_{(i,k)} = \phi_{(j,k')}$, where $\Omega^{\text{pixel}} = \{(1, 1), (1, 2), \dots, (n, K - 1), (n, K)\}$ denotes the set of all sub-pixels. Because the sub-pixel of the same perturbation pixel i has the equal value, given two sub-pixels (i, k) and (i, k') of the same perturbation pixel i , $z(S \cup \{(i, k)\}) = z(S \cup \{(i, k')\})$ holds for any set $S \subseteq \Omega^{\text{pixel}} \setminus \{(i, k), (i, k')\}$, where $1 \leq k, k' \leq K$, and $k \neq k'$. In this way, $\phi_{(i,1)} = \phi_{(i,2)} = \dots = \phi_{(i,K)}$. Thus, we approximate the attribution of perturbation pixel i as $\phi_i = \sum_{i=1}^K \phi_{(i,k)}$, which equals to $\phi_i = K \cdot \phi_{(i,k)}$.

B.2. Properties of the approximated attribution

In Section Algorithm, we approximate the attribution of perturbation pixel i as $\phi_i = \sum_{i=1}^K \phi_{(i,k)}$. In the supplementary material, we further discuss properties of the approximated attribution.

The approximated attribution still satisfies the linearity axiom and the efficiency axiom.

Proof of the linearity axiom: Given two score functions $v(S)$ and $w(S)$, we use ϕ_i^v and ϕ_i^w to denote the attribution of perturbation pixel i to score v and score w respectively. Let there be a new score function $f'(S) = v(S) + w(S)$. We use ϕ_i^{v+w} to denote the approximated attribution of perturbation pixel i to the new score function. The approximated attribution of perturbation pixel i is the sum of attributions sub-pixels, *i.e.* $\phi_i^{v+w} = \sum_{k=1}^K \phi_{(i,k)}^{v+w}$. The attribution of each sub-pixel is defined as the Shapley value. The Shapley value satisfies the linearity axiom. Then $\sum_{k=1}^K \phi_{(i,k)}^{v+w} = \sum_{k=1}^K (\phi_{(i,k)}^v + \phi_{(i,k)}^w) = \phi_i^v + \phi_i^w$. In this way, the approximated attribution is proved to satisfy the linearity axiom, *i.e.* $\phi_i^{v+w} = \phi_i^v + \phi_i^w$.

Proof of the efficiency axiom: The approximated attribution of each perturbation pixel is the sum of attributions of corresponding sub-pixels. Thus, the sum of approximated attributions of all perturbation pixels is the sum of attribu-

tions of all sub-pixels, *i.e.* $\sum_{i=1}^n \phi_i = \sum_{i=1}^n \sum_{k=1}^K \phi_{(i,k)}$. Attributions of sub-pixels satisfy the efficiency axiom, *i.e.* $\sum_{i=1}^n \sum_{k=1}^K \phi_{(i,k)} = z(\Omega^{\text{pixel}}) - z(\emptyset)$. $z(\Omega^{\text{pixel}})$ is the score gained with all sub-pixels, *i.e.* the score made by the whole adversarial perturbation δ , and $z(\emptyset)$ is the score produced without the adversarial perturbation, *i.e.* the score made by the original image. $z(\Omega)$ also represents the score made by the whole adversarial perturbation δ , where $\Omega = \{1, 2, \dots, n\}$ is the set of all perturbation pixels. Thus, $z(\Omega^{\text{pixel}}) = z(\Omega)$, and $\sum_{i=1}^n \sum_{k=1}^K \phi_{(i,k)} = z(\Omega^{\text{pixel}}) - z(\emptyset) = z(\Omega) - z(\emptyset)$. In this way, the approximated attribution is proved to satisfy the efficiency axiom, *i.e.* $\sum_{i=1}^n \phi_i = z(\Omega) - z(\emptyset)$.

B.3. Approximation for attributions of sub-pixels based on the Taylor expansion

In the paper, we approximate attributions of sub-pixels based on the Taylor expansion as Equation (8). In the supplementary material, we aim to derive the approximation in details.

Given a function $f(x_1, x_2, \dots, x_n) : \mathbb{R}^n \rightarrow \mathbb{R}$, the Taylor expansion at $(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)})$ is

$$\begin{aligned} f(x_1, x_2, \dots, x_n) &= f(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}) \\ &+ \sum_{i=1}^n (x_i - x_i^{(k)}) \frac{\partial f(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)})}{\partial x_i} \\ &+ o((x_1 - x_1^{(k)}, x_2 - x_2^{(k)}, \dots, x_n - x_n^{(k)})) \end{aligned}$$

$z(S \cup \{(i, k)\})$ denotes the change of the prediction score of the DNN made by sub-pixels in $S \cup \{(i, k)\}$, where $S \subseteq \Omega^{\text{pixel}} \setminus \{(i, k)\}$. The Taylor expansion for $z(S \cup \{(i, k)\})$ at S is given as

$$z(S \cup \{(i, k)\}) \approx z(S) + \delta_{(i,k)} \cdot \frac{\partial z(S)}{\partial \delta_{(i,k)}}$$

Thus, the approximation for the Shapley value of the sub-pixel (i, k) is given as

$$\begin{aligned} \phi_{(i,k)} &= \frac{1}{nK} \sum_{S \subseteq \Omega^{\text{pixel}} \setminus \{(i,k)\}} \binom{nK-1}{|S|}^{-1} [z(S \cup \{(i,k)\}) - z(S)] \\ &\approx \frac{1}{nK} \sum_{S \subseteq \Omega^{\text{pixel}} \setminus \{(i,k)\}} \binom{nK-1}{|S|}^{-1} \left(\frac{\partial z(S)}{\partial \delta_{(i,k)}} \delta_{(i,k)} \right) \end{aligned}$$

Let there be m components in a certain clustering step. $C_k^{(u)} = \bigcup_{i \in C^{(u)}} (i, k)$ denotes a sub-component. We use $\Omega^{\text{comp}} = \{C_1^{(1)}, C_2^{(1)}, \dots, C_{K-1}^{(m)}, C_K^{(m)}\}$ to denote the set of all sub-components. The Shapley value of the sub-component $C_k^{(u)}$ is approximated as

$$\begin{aligned} \phi_{C_k^{(u)}} &= \frac{1}{mK} \sum_{S \subseteq \Omega^{\text{comp}} \setminus \{C_k^{(u)}\}} \binom{mK-1}{|S|}^{-1} [z(S \cup \{C_k^{(u)}\}) - z(S)] \\ &\approx \frac{1}{mK} \sum_{S \subseteq \Omega^{\text{comp}} \setminus \{C_k^{(u)}\}} \binom{mK-1}{|S|}^{-1} \sum_{(i,k) \in C_k^{(u)}} [z(S \cup \{(i,k)\}) - z(S)] \\ &\approx \frac{1}{mK} \sum_{S \subseteq \Omega^{\text{comp}} \setminus \{C_k^{(u)}\}} \binom{mK-1}{|S|}^{-1} \sum_{(i,k) \in C_k^{(u)}} \left(\frac{\partial z(S)}{\partial \delta_{(i,k)}} \delta_{(i,k)} \right) \end{aligned}$$

B.4. Implementation & computational complexity:

Clarification: In both the paper and the supplementary material, the computational complexity is quantified as times of network inference, *i.e.* the number of input (masked) images on which we conduct the forward/backward propagation. We do not count the number of detailed operations during the forward/backward propagation *w.r.t.* each specific input image, in order to simplify the analysis. It is because given a specific DNN, the number of detailed operations during the forward/backward propagation is the same for different input images.

In the paper, we introduce the implementation of the approximation of Shapley values and analyze the computational complexity. In the supplementary material, we aim to further explain the computational complexity of our approximation for attributions and how we approximate the attribution of components in detail.

We use a sampling-based method to reduce the complexity of computing Shapley values. The original formulation of the Shapley value considers all combinations of pixels to compute the Shapley value for each pixel. Thus, the computational complexity of the Shapley value of each pixel is $O(2^n)$. We implement the approximation of Shapley values of sub-pixels with a sampling method. In this way, the complexity of computing the Shapley value of one sub-pixel is reduced to $O(nKT)$. Note that $\phi_i \approx K \cdot \phi_{(i,k)}$. Therefore, the complexity of approximating the Shapley value of each pixel is also $O(nKT)$. The derivatives towards all sub-pixels can be computed simultaneously via back-propagation. Thus, the computational complexity of computing Shapley values of all pixels remains $O(nKT)$.

We use hierarchical clustering to iteratively merge several components into a larger component based on interactions. We use the following approximation method, to compute and reduce the complexity of computing the attribution of the pair of components. Let there be m components in a certain clustering step. Given a component $C^{(u)}$, we can use the sampling method to get their attributions $\phi_{C^{(u)}}$. Here, from the perspective of game theory, each component is a player, and there are m players in the game. As mentioned above, the complexity of computing $\phi_{C^{(u)}}$ is $O(mKT)$. We use $S^c = C^{(i_1)} \cup C^{(i_2)} \cup \dots \cup C^{(i_q)}$ to denote a component candidate. To determine the interaction inside S^c , we need to compute ϕ'_{S^c} . The computation of ϕ'_{S^c} regards $C^{(i_1)}, C^{(i_2)}, \dots, C^{(i_q)}$ as a single component. Then, the set of components changes to $\{S^c, C^{(i_{k+1})}, \dots, C^{(i_m)}\}$ with $m-q+1$ players. In this way, the computational complexity of ϕ'_{S^c} is $O((m-q)KT)$. Whereas, considering all potential pairs of components, the computational complexity grows. We only consider the interaction between neighboring components. There are m potential pairs of components, and the complexity of computing all potential pairs of components is $O(m(m-q)KT)$.

Considering the local property [6], we can further approximate $\phi'_{C^{(u)} \cup C^{(v)}}$ by simplifying contextual relationships of far-away pixels. Here, instead of computing ϕ'_{S^c} in the set $\{S^c, C^{(i_{k+1})}, \dots, C^{(i_m)}\}$, we randomly merge \tilde{m} components to get \tilde{m}/q component candidates, including S^c . In this way, the new set includes \tilde{m}/q component candidates and $m - \tilde{m}$ components, *i.e.* $\{S^c, \bigcup_{a=q+1}^{2q} C^{(i_a)}, \dots, \bigcup_{a=\tilde{m}-q+1}^{\tilde{m}} C^{(i_a)}, C^{(i_{\tilde{m}+1})}, \dots, C^{(i_m)}\}$. We can simultaneously compute attributions of \tilde{m}/q candidates in the new set, and the computational complexity is $O((m - (q-1)\tilde{m}/q)KT)$. To compute attributions of all potential component candidates, we need to sample qm/\tilde{m} different sets. In this way, the overall complexity for the computation of attributions of candidates is reduced from $O(m(m-q)KT)$ to $O(m(qm/\tilde{m} - q)KT)$.

C. Pseudo code of extracting perturbation components

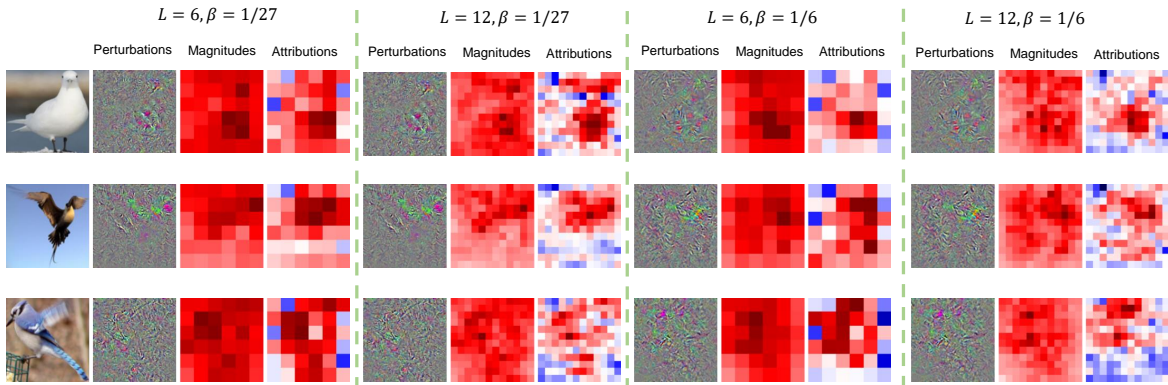
Algorithm 1 Extraction of perturbation components via hierarchical clustering

-
- 1: **Inputs:** pixel set Ω ; reward function $z(\cdot)$; component size q ; iteration times T
 - 2: **Outputs:** Component set Ω' ;
 - 3: **Initialization:** $\Omega' = \Omega$
 - 4: **for** $iter = 1$ to T **do**
 - 5: $\forall C \in \Omega'$, compute ϕ_C with reward function $z(\cdot)$
 - 6: **while** *not* all possible component candidates are considered **do**
 - 7: Get component candidate set $\Omega^{\text{candidate}}$ by randomly merging each group of neighboring q components in Ω'
 - 8: $\forall C_{\text{candidate}} \in \Omega^{\text{candidate}}$, compute $\phi_{C_{\text{candidate}}}$ with reward function $z(\cdot)$
 - 9: Compute interaction in each component candidate: $I = \phi_{C_{\text{candidate}}} - \sum_{C \in C_{\text{candidate}}} \phi_C$
 - 10: **end while**
 - 11: $\Omega' = \emptyset$
 - 12: Update the component set Ω' by greedily adding the component candidate with highest interaction strength $|I|$ to Ω'
 - 13: **end for**
-

D. Additional experimental results of regional attributions

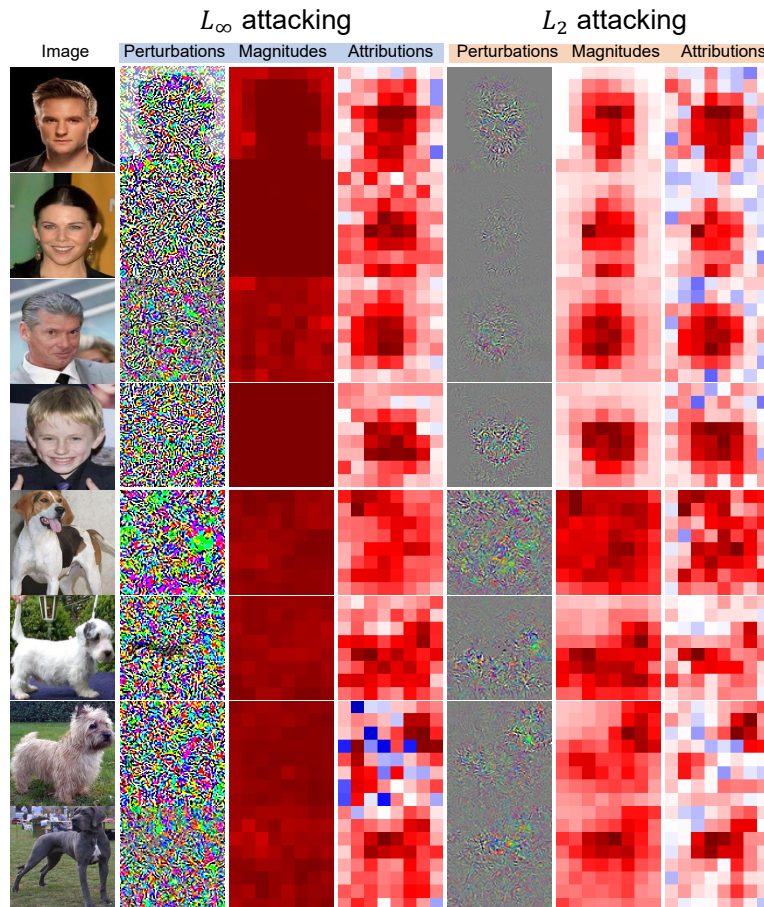
D.1. Regional attributions computed with different hyper-parameters

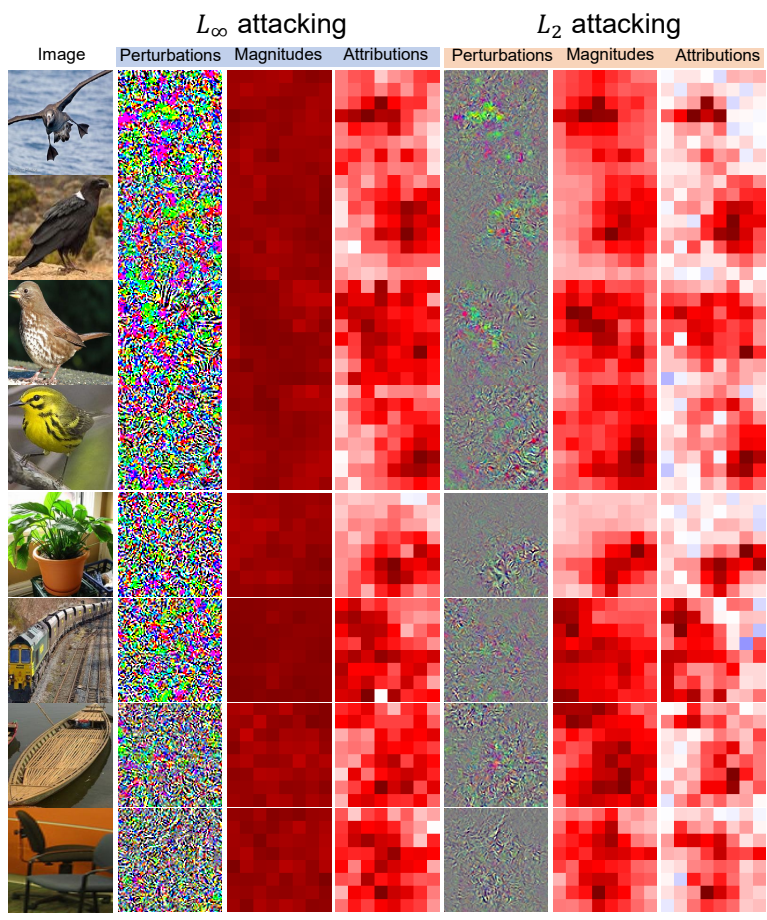
In this section, we have compared regional attributions with different hyper-parameters (β and L). The results are shown as follows. We found that important regions indicated by attributions were similar under the same selection of β , such as the belly region of the pigeon (in the first row) and the wing region of the jaeger (in the second row). Note that when β were different, the generated adversarial perturbations would be different, which led to different regional attributions. However, compared with the difference between the magnitudes and attributions, the difference between regional attributions computed with different hyper-parameters was smaller.



D.2. More experimental results of regional attributions

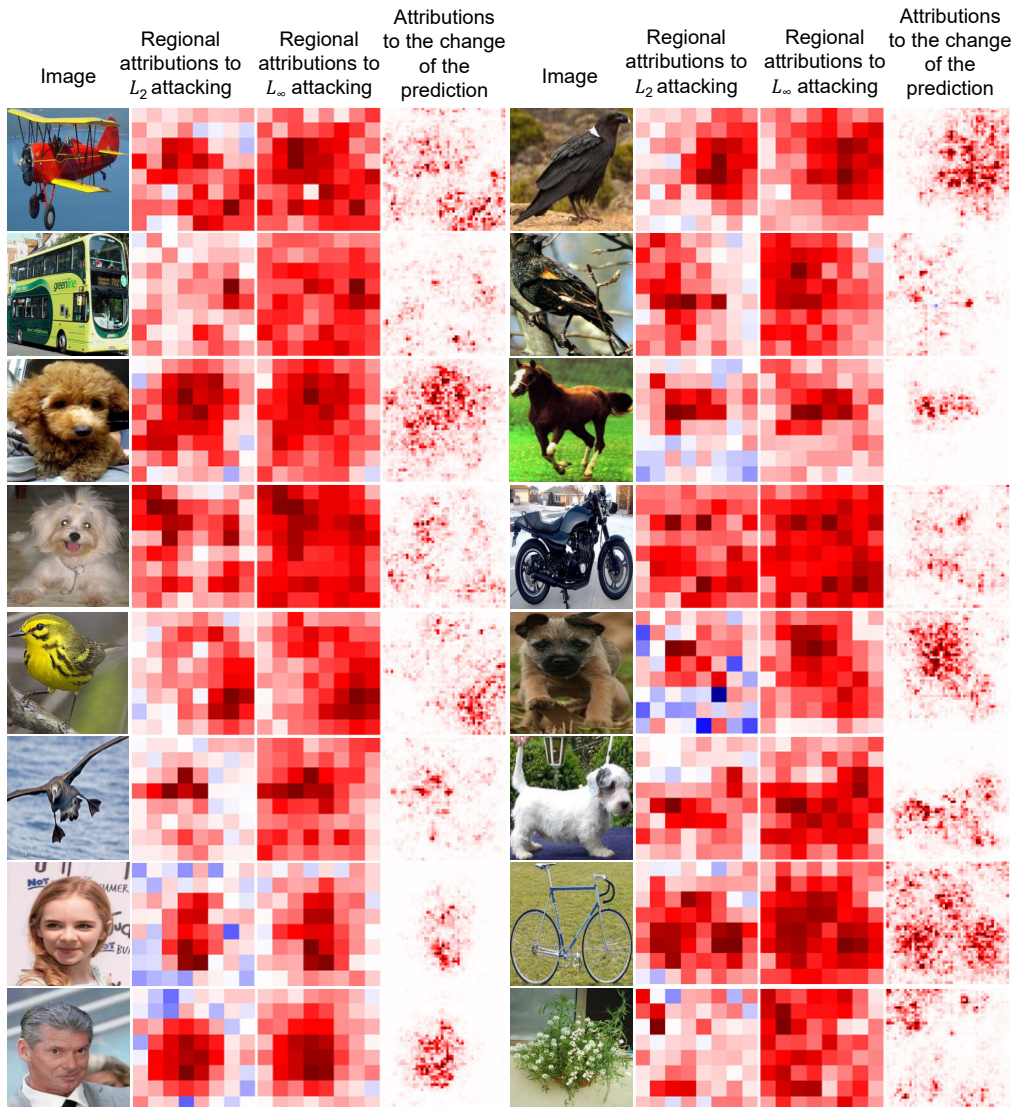
Experimental results of regional attributions have been shown in Fig. 3 in the paper. In the supplementary material, we give additional results of regional attributions. The visualization shows that although the distribution of L_2 adversarial perturbations and the distribution of L_∞ adversarial perturbations were dissimilar, their regional attributions were similar to each other.





E. Comparisons of attributions

There are two types of attributions in the paper, *i.e.* regional attributions to the attacking cost and pixel-level attributions to the change of prediction score (under L_2 attacking)). We visualize regional attributions to the cost of L_2 attacking and L_∞ attacking and pixel-wise attribution to the change of the prediction score. In most cases, important regions indicated by these attributions were similar. For example, in the third row, the dog's head and the horse's body were indicated to be important by all three kinds of attributions. In other cases, important regions indicated by different attributions were different. For example, important regions of the potted plant in the last row indicated by these three kinds of attributions were dissimilar.



F. More experimental results of interactions and perturbation components

Experimental results of interactions and perturbation components have been shown in Fig. 4 in the paper. In the supplementary material, we give more examples of visualizations. Perturbation components usually were not aligned with visual concepts.

