Abdel-Hakim *J Big Data* (2019) 6:51
https://doi.org/10.1186/s40537-019-0213-4

Journal of Big Data

RESEARCH Open Access

Ally patches for spoliation of adversarial patches

Alaa E. Abdel-Hakim[1,2*]

*Correspondence:
alaa.aly@eng.au.edu.eg;
adali@uqu.edu.sa
[1] Electrical Engineering Department, Assiut University, Assiut 71516, Egypt
Full list of author information is available at the end of the article

Abstract
Adversarial attacks represent a serious evolving threat to the operation of deep neural networks. Recently, adversarial algorithms were developed to facilitate hallucination of deep neural networks for ordinary attackers. State-of-the-arts algorithms could generate offline printable adversarial patches that can be interspersed within fields of view of the capturing cameras in an innocently unnoticeable action. In this paper, we propose an algorithm to ravage the operation of these adversarial patches. The proposed algorithm uses intrinsic information contents of the input image to extract a set of ally patches. The extracted patches break the salience of the attacking adversarial patch to the network. To our knowledge, this is the first time to address the defense problem against such kinds of adversarial attacks by counter-processing the input image in order to ravage the effect of any possible adversarial patches. The classification decision is taken according to a late-fusion strategy applied to the independent classifications generated by the extracted patch alliance. Evaluation experiments were conducted on the 1000 classes of the ILSVRC benchmark. Different convolutional neural network models and varying-scale adversarial patches were used in the experimentation. Evaluation results showed the effectiveness of the proposed ally patches in reducing the success rates of adversarial patches.

Keywords: Adversarial patches, Ally patches, CNN, Deep neural networks

Introduction

Vision-based intelligent systems and applications have been increasing rapidly. The continuous growth of dependence on automated systems is a double-edge sword. On the positive side, intelligent systems make human daily life easier and more comfortable. On the other negative side, these systems are vulnerable to manipulation by attackers, either humans or software robot agents. The consequences of successful attacks have diverse degrees of criticality depending on the nature of the underlying application. The consequent troubles may vary from just unpleasant inconvenience in applications like entertainment image and video annotation, passing by security-critical problems like false person identifications, and can turn out to be life-threatening in autonomous navigation and driver support systems.

The widespread and success of convolutional neural networks applications in the fields of object recognition and concept classification attracted more attackers' attention. Adversarial attackers, and deep learning researchers as well, have been attempting to fool convolutional neural networks in order for hallucinating the networks and forcing them to produce false results [3]. Such attempts are conducted for different reasons. While the main goal of adversarial attackers is destruction by causing intelligent systems malfunction, or even for boast purposes, the machine learning researchers are interested in comprehensive investigation of deep network behavior and strength/weak points.

There is a non-stopping race between deep learning systems attackers and defenders. Attackers seek to fool the network such that false outputs are generated with invisible or minimal perceptible changes in the input images [9, 23]. On the other side, defenders continuously rebuild the deep models to fill the gaps through which attackers can sneak into the systems.

Adversarial attacks can be categorized according to their operation on the input side or according to their malicious effect on the network outputs. From the input side, the adversarial methodologies are mainly oriented to find the least observable way for input perturbation. One paradigm is to make imperceptible modifications in some pixels or regions of the input images. DeepFool [16], Projected Gradient Ascent [4], Fast Gradient Sign Method [9], Projected Gradient Descent [13], and L-BFGS [23] are examples of attacking methodologies that attempt to make unremarkable small variations to the input image pixels. These variations are carefully created to lead the network to misclassification. Other approaches perform adversarial modifications on some of the image pixels, either scattered [17] or on the form of a small fixed-location patch in the image, e.g. [21]. Jacobian-Based Saliency Map [17] uses a forward derivative-based approach to change the intensity of some pixels in the image. The classification output of the network is perturbed due to these small changes of only subsets of input pixels. Although pixel-altering-based approaches perform well toward their end goal of DNN's hallucination, most of them suffer from impracticability in real-life applications. In other words, in most of real-time cases, e.g. surveillance and autonomous navigation systems, the direct inputs, right before the network's input layer, are not accessible to the attacking algorithm to perform whatever pixel changes it wants. While this kind of algorithms may be feasible to be applied in such real-time applications for research purposes, it is almost infeasible for practical adversarial purposes.

Therefore, other algorithms were developed to be more realistic for applications with which the digital input is inaccessible to the attacking algorithms. A common strategy of such algorithms is to place a perturbing object in the scene during image capturing. The perturbing objects may be conspicuous like posting a large poster of some traffic sign, e.g. stop sign, or placing some perturbing stickers on the sign itself [8]. In other cases, the perturbing object may not be such explicit. For example, an impersonate attacker may wear an innocent colorful glasses frame to deceive a facial recognition system [21]. Other examples of such attacks are to place printed 2D [3, 12] or 3D objects to cause network misclassification [1].

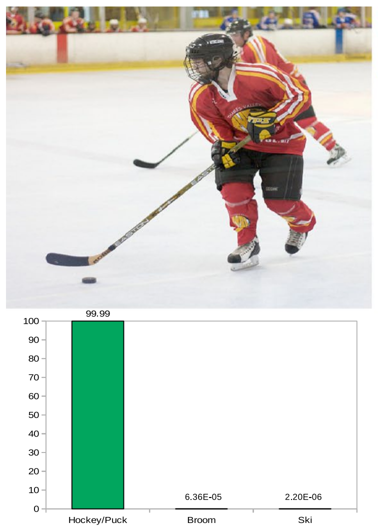
On the output side, or the impact on the perturbed network, an attacking algorithm aims at destruction of the network integrity. In this context, attacking algorithms can be categorized according to their goals into four categories [17]: confidence reduction, misclassification, targeted misclassification, and source/target misclassification. Furthermore, we re-categorize these kinds of threats into three levels of severity: low, medium, and high. The first level, the low severity level, includes attacks that aim at reducing classification confidence. This kind of attacks force the victim network to look more “hesitant” in terms of the classification output. Although this kind of threat affects the robustness of the network, it can be overcome as long as the top classification orders are not disturbed. The second severity level is the medium level. This level is assigned to algorithms that attempt to cause non-targeted misclassification. The reliability of networks that are subjected to this kind of threats is ruined. However, it is hard to exploit such attacks for system hacking in applications like surveillance, since the goal of these attacks is to cause malfunctions only not to drive the network to a specific targeted output. The last severity level is the high severity one. Targeted and source/targeted misclassification go under this level. In targeted misclassification, a specific object is detected regardless the contents of the input image [3, 21]. In source/target misclassification, a specific target is paired with a specific input. These kinds of threats could be catastrophic to many systems. For example, in surveillance systems, unauthorized person may be allowed in restricted areas. In security applications, terrorists can pass weapons, bombs, and restricted materials in front of automated surveillance cameras. In driver support systems, stop signs, red lights, or emergency lanes can be positively or negatively misclassified causing severe accidents.

So, the latter kind of techniques is the most dangerous, especially if a simple convenient attacking approaches of placing perturbing 2D or 3D objects is adopted. This is because of its ease-to-use by the attackers and the difficulty to suspect or accuse the attacker with attempts of adversarial actions.

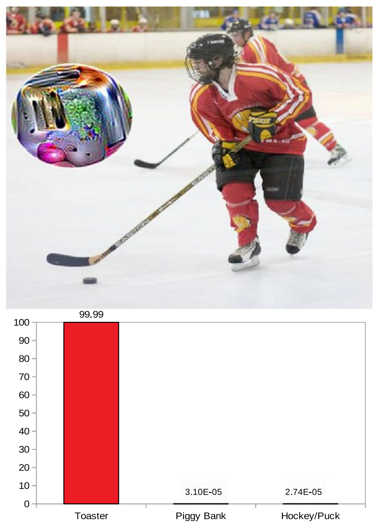
So far, the research studies that present defenses against adversarial examples are limited [2, 11, 14, 15, 17, 24]. Carlini and Wagner [5–7] have pointed out that these algorithms suffer from vulnerability to optimization-based attacks. Obfuscated gradients have been used to develop robust defenses against optimization-based attacks [2]. However, Athalye et al. [2] have shown that even these defenses provide illusive sense of security against adversarial examples.

In this paper, we propose ally patches in order to limit the effect of threats of the third severity level. Ally patches are generated intrinsically from the input image in a blind manner. In other words, regardless attacked or not, the contents of the input image are used to generate a set of ally patches. These patches form an alliance to face an unknown possible adversarial patch. The image entropy is used to assess the information contents of the candidate image region for joining the ally patch collection. The nature of ally patch extraction guarantees its robustness against known optimization attacks, since it intrinsically uses the image contents in prior to exposition to the input layer of the neural network. A glimpse of the effect of the proposed ally patch in ravage of adversarial patches is shown in Fig. 1.

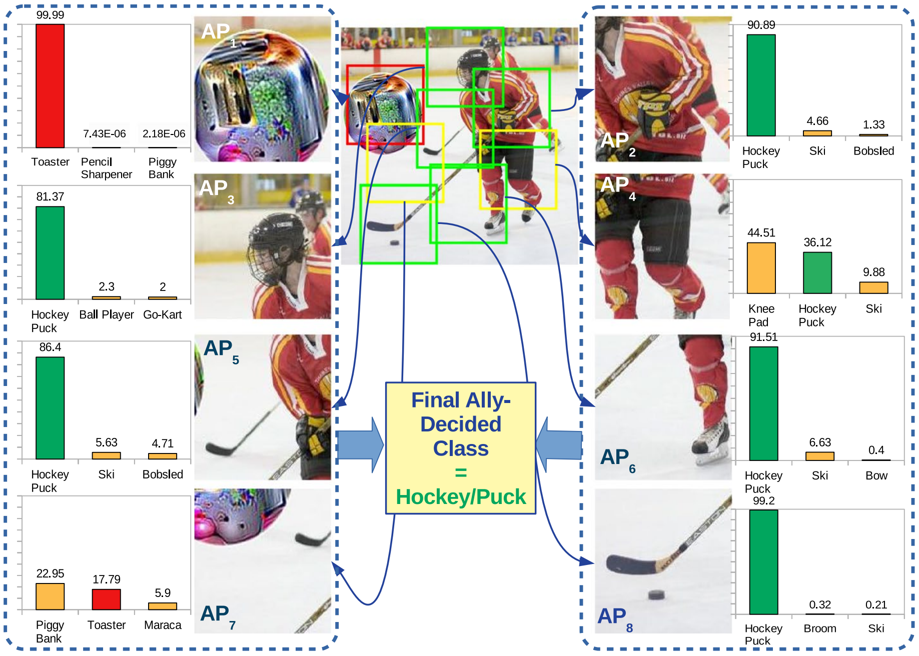
The rest of this paper is organized as follows: in "Adversarial patches" section, adversarial techniques are presented. "Ally patches" section details the proposed ally patch algorithm. The details of conducted experiments setup are illustrated in "Experimental setup" section. Evaluation results are discussed in "Results and discussions" section. Finally, in "Conclusions" section, the paper is concluded and directions for future work are highlighted.



(a) Attack-free input image



(b) Attacked image



(c) Ally patching and classification

Fig. 1 An exemplar image to illustrate the effect of the proposed method. **a** An attack-free input image. The convolutional neural network succeeded in classifying the image in compliance to the ground truth with confidence close to 100%. **b** The input image after having been attacked by a toaster-targeted adversarial patch generated by [3]. The adversarial patch succeeded in driving the network to hallucination and to classify the input image to the adversarial target with a confidence value of almost 100%. **c** Classification using the proposed ally patches. The generated patch alliance could ravage the operation of the adversarial patch and flee from the fake saliency caused by it. For illustration clarity purposes, the confidence values of the top-3 labels only are shown. The used CNN model for this example is VGG19 [22]

Adversarial patches

The general supervised classification problem in neural networks can be summarized as finding a set of weight values, $\hat{W} \in \mathbb{R}^{m\times n}$, that achieves the following constraint:

$$\hat{W} = \arg\min_{W}(|Wx - y_{gt}|) \tag{1}$$

where $x \in \mathbb{R}^n$ is the input vector and $y_{gt} \in \mathbb{R}^m$ is the ground truth output vector. Once trained, neural network weights are fixed; hence, practically, the only fully-accessible values to user are those of the input vector, x. The output vector $y \in \mathbb{R}^m$ is a read-only vector that cannot be modified directly by the user.

The main goal of an adversarial attack is to change the output vector y according to the kind of attack as follows:

$$\hat{y} = \begin{cases} y', & \text{s.t. } y' \neq y_{gt}, \\ y_t, & \forall x \in \mathbb{R}^n, \\ y_t^i, & \text{for } x^i = x_v^i, i = \{1, 2, .., n\}. \end{cases} \tag{2}$$

where y_t is a specific target label and x_v is a victim input. The first label value in Eq. (2) is used in the case of a non-targeted attack. The second assignment is used for a blind targeted attack, where a specific target label y_t is desired to be assigned to the output vector regardless what the input vector is. The last assignment goes to the case when a source/target attack is considered. In this case, a specific target label y_t^i is set to a desired value $\hat{y}$ for the i^{th} item in the input space. Since the output is not changeable directly, the general adversarial strategy is to find a perturbed input, $\hat{x}$, that lead to maximum likelihood $P(\hat{y}|\hat{x})$ subject to the constraint that the perturbation does not exceed a certain threshold, ϵ [3]. Formally, this procedure can be formulated by the following equation:

$$\hat{x} = \arg\max_{\hat{x}} \log([P(\hat{y}|\hat{x})]) \text{ s.t. } ||x - \hat{x}||_\infty \leq \epsilon \tag{3}$$

While the attacks produced by this approach are sufficiently camouflaged, the required comprehensive modification of the input image is not always feasible [3].

To overcome this crucial practical limitation, the attack strategies have been modified to change small regions of the image, or even to generate small adversarial patches from scratch [3]. The most dangerous threat caused by such strategy is resulted from the usage convenience to attackers. Particularly, an attacker can generate or download digital versions of these adversarial patches. Then, he/she prints colored stickers of these patches. Finally, he/she shows it up in the field of view of operating cameras in an innocent way varying according to the attacked system. For example, the attacker can through dozens of these patches from the window of his/her car while traveling on a high way to cause autonomous driver support systems, (DSSs), malfunction by detecting stop-signs or red-light in the middle of the highway. The damage caused by a sudden stop or slowing down of a vehicle on a highway is obvious. Accordingly, Brown et al. [3] have recently developed their adversarial patches. They used a learning approach to learn the shape, location, and 2D transformation of their generated adversarial patches. The training is performed for adversarial patches to produce a specific target label regardless the existing background in the original image. In other words, the adversarial patch contents force the network to perceive it as the most salient component of any input image.

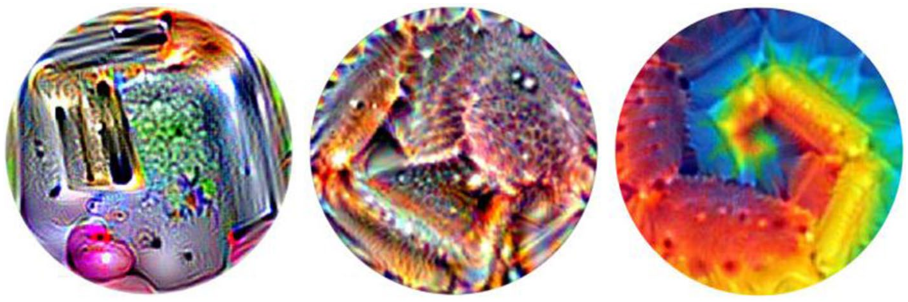


Fig. 2 The used adversarial patches for evaluation, as generated by [3]. Whereas the patches are imperceptible by humans, they are perceptible and salient to the CNN models

Figure 2 shows exemplar adversarial patches which are learned to produce a number of misclassified labels.

An additional feature in that approach is its capability to produce camouflaged patches with minimum changes from a starting patch image. The patch ability to be covert is implemented by adding the following constraint to the used objective function: $||p - p_{orig}||_\infty < \epsilon$.

Ally patches

As discussed in "Adversarial patches" section, the main working strategy of an adversarial patch is to cause network "distraction" by changing saliency characteristics of the input image. While doing this, the adversarial attacker is keen to minimize the perceived changes of the input image by a human observer. Fortunately, attackers utilize a major distinguishing feature between DNN and human perception, which is the saliency perceived by humans versus that perceived by networks. We make use of this specific weak point to generate our proposed ally patches. Our main hypothesis is: if we could break or ravage the adversarial saliency perceived by the network, while preserving the original characterizing features of the input image, we will succeed in stopping the harmful effect of the attacking adversarial patch. Our approach intrinsically utilizes the input image contents without any prior knowledge about the nature of the possible adversarial patches, or even knowledge about probability of existence of any adversarial patch in the input image.

Figure 3 shows a block diagram that outlines the extraction and the operation of the proposed ally patches. The following subsections explain the details of each stage in this process.

Ally patch extraction

The first step is to extract a comprehensive set of ally patch candidates from the input image. Assume the size of the input image is $h \times w$ pixels. The input image is cropped using a sliding window whose fixed size of $h_p \times w_p$. The original image is scanned by this sliding window with a stride, ST. This step generates a total number of $\frac{(h-h_p).(w-w_p)}{ST^2}$ candidate ally patches.

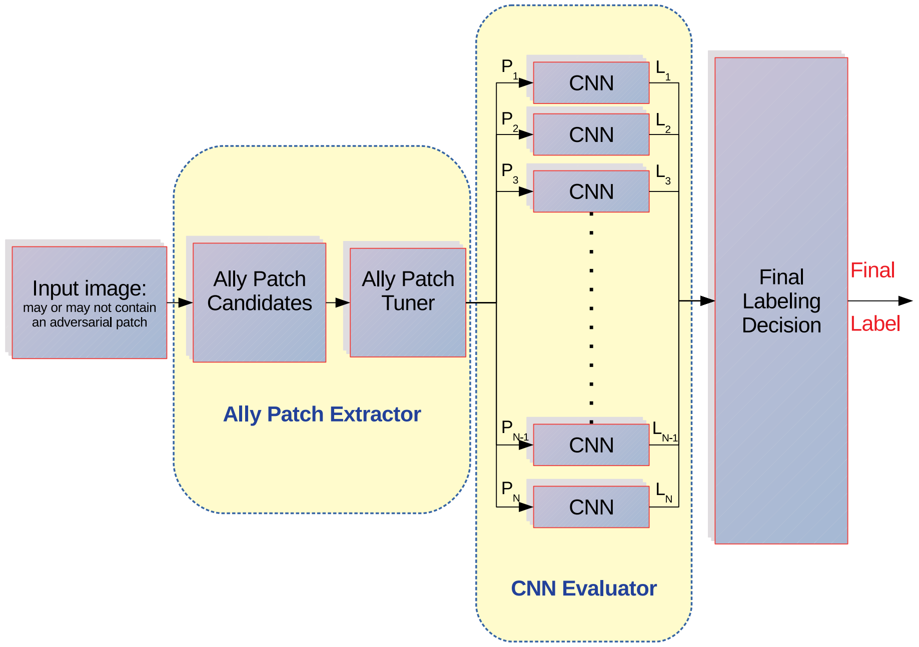


Fig. 3 The block diagram of the entire process of generating ally patches and readjustment of the classification output. The ally patch extractor intrinsically extracts 'promising' ally patches from the input image. Then identical, yet independent, copies of a pre-trained convolutional deep network are used to evaluate the classification of each of the ally patches. Finally, the fusion stage decides the most appropriate final label of the entire image. Any possible adversarial patch is counter-attacked during the ally patch extraction stage. Fled adversarial patches from the evaluator are taken care of in the final stage

The main criterion of selecting the final set of ally patches is their information contents. We use Shannon's entropy [20] to measure the information contents of each candidate ally patch, AP, as shown in Eq. (4).

$$H(AP_i) = \sum_{k=0}^{255} p_k . \log\left(\frac{1}{p_k}\right) \tag{4}$$

where p_k is the probability of an intensity value, k, to appear in any of the three RGB channels of the candidate patch $AP_i, 1 \leq i \leq \frac{(h-h_p).(w-w_p)}{ST^2}$.

Using entropy as a measure of the information contents may suffer from a limitation, which can be considered minor in most of underlying applications. Particularly, object localization is missing when considering patch entropy. However, it is obvious that object localization is not critical in most of classification tasks.

The patches in the initially-generated dense set are filtered to preserve the most non-redundant informative ones. A patch AP_i is included in the final ally patch set, if two constraints are achieved. The first is the information contents constraint, Eq. (5). It makes sure that the selected patch has a minimum amount of information contents compared to the parent image.

$$H(AP_i) \geq Th_{ent}.H(I) \tag{5}$$

where $H(AP_i)$ is the entropy of the selected patch, $H(I)$ is the entropy of the entire input image, Th_{ent} is a threshold value for the minimum patch entropy relative to the image's total entropy. In monotonous background images, e.g. sky, sea, and desert scenes, the adversarial patch will be dominant. This means that most of non-adversarial patches will fail to fulfill this constraint. Fortunately, scenes in most sensitive applications usually do not have such backgrounds.

The second constraint is a similarity constraint. Under this constraint the extracted patches are guaranteed to be sufficiently dissimilar to assure diversity of the training set. The constraint can be achieved by limiting the maximum allowed overlap area between extracted patches, Eq. (6), or by limiting the allowed mutual information [18] of each of the extracted patch-pairs, Eq. (7).

$$area(AP_i \cap AP_j) \leq Th_s.h_p.w_p \ \forall H(AP_j) > H(AP_i) \tag{6}$$

$$\left[H(AP_i) + H(AP_j) - H(AP_i, AP_j)\right] \leq Th_s \ \forall H(AP_j) > H(AP_i) \tag{7}$$

where Th_s is the threshold of the maximum allowed similarity.

All patches, which achieve these two constraints, collectively build up the fine-tuned final ally patch set, $\mathbb{AP}$.

Figure 1 illustrates the effect of application of these stages on an exemplar image. In this figure, the effect of these two constraints can be easily noticed. The cardinality of $\mathbb{AP}$ equals eight. The elements of $\mathbb{AP}$ are ordered from AP_1 to AP_8. They are sorted according to the order of achieving the constraints of Eqs. (5) and (6), respectively. AP_1 is the richest in terms of information contents. Naturally, as the adversarial patches are usually highly-textured, they start to show up among the top ally patches. Fortunately, classifications obtained by the extracted patches have equal weights regardless the order in the entropy-sorted list.

CNN evaluation

In this stage, each element of the final fine-tuned ally patch set, $\mathbb{AP}$, is exposed to an instance of a specific pre-trained convolutional neural network model. Each patch is treated as a completely independent input image and is assigned to an independent label, $L_i, 1 \leq i \leq |\mathbb{AP}|$. In this work, the used CNN models are identical. However, different models might be used for a diverse classification. Moreover, the conceptual operation of ally patches guarantees its validity with any whole-image-based classifier.

The attacking adversarial patch may face one of three scenarios. The first occurs with highly-textured input images. In this case, the adversarial patch will fail to achieve the information content constraint, Eq. (5). Thus, it will not be included in $\mathbb{AP}$, i.e. the risk is eliminated completely. Practically, the probability of complete exclusion of the attacking adversarial patch from $\mathbb{AP}$ is not high. This is because the created attacking adversarial patches are usually rich in texture, which means that their entropy values are usually high. So, they often fulfill the information content constraint.

The second scenario, whose the highest probability, is to have the adversarial patch partially appears in one or more of the elements of $\mathbb{AP}$. In this case, like the case of Fig. 1, and depending on the used CNN model, there is a high chance that the model will not classify the input patch as desired by the adversarial patch, since the adversarial patch is a "broken" one.

In the last scenario, the adversarial patch intended attack succeeds. In other words, the desired misclassification label is assigned to the patch. This happens in two cases. The first is with the patches that passed through the second scenario, but the damage to the adversarial patch was not sufficient to cause attacking failure. The second case occurs when an ally patch in $\mathbb{AP}$ contains the entire adversarial patch. This scenario is taken care of in the final stage, which is explained in the following subsection.

These scenarios are represented, to some extent, in the example of Fig. 1. In this exemplar image, the third scenario, which is the most dangerous, is represented in patch AP_1. Most of the adversarial patch is isolated in this ally patch. Therefore, the targeted adversarial classification succeeded for this patch. Patch AP_7 gives an exemplar case for the second scenario. It contains about 20–25% of the adversarial patch. This partial existence of the adversarial patch causes misclassification. Nonetheless, the network was not deceived by the adversarial target label, *'Toaster"*. In this case, although the harmful effect of the adversarial attack could not be eliminated completely, its severity degree was reduced. Patching caused AP_4 to isolate an image portion that is more relevant to another class, "*Knee Pad*", which is different from the ground truth label, "*Hockey Puck*". Although AP_4 is misclassified, the confidence levels of the ground truth label and the false label are close.

However, as shown in the next subsection, the fusion stage shall take care of such misclassifications, which are caused either by patching significant portions of the attacking adversarial patch or by patching an irrelevant portion of the input image. So, we can say that the adversarial patch was successfully broken and isolated in few patches whose corrupted classifications have to face correct classifications come from the alliance of the other patches.

Fusion and final classification decision

The inputs to this stage are N labels: $L_i, 1 \leq i \leq N$, where $N = |\mathbb{AP}|$. Each label is associated with a confidence value, c_i, which represents the confidence score of the output from the CNN model. This label set represents the candidate labels to the final image classification. The function of this stage is to fuse these labels together and reach a final decision of the input image class.

The main advantage of ally patch extraction appears in this stage. Specifically, even if the attacking adversarial patch succeeded in deceiving the network, as discussed in the third scenario of the previous subsection, its influence to the final labeling decision will be equal to other "clean" patches. Usually, the number of these clean ally patches is larger than the number of adversarial patches that succeeded to flee in the first stage. This gives an advantage to ally patches.

The fusion process is performed according to one of the following four strategies:

Majority voting

The input patches contribute in a fair voting pool. If a tie occurs between the top labels, the label whose larger average confidence score wins.

Total confidence

The confidence scores of the input patches whose same labels are added to vote for their corresponding label. This gives an advantage to the confident patches, even if they are few when compared to other patches. This is a risky strategy because confidence values of the fled adversarial patches in the third evaluation scenario of "CNN evaluation" section are often high.

Average confidence-weighted

This metric is calculated in a similar way of the total confidence metric. The only difference is taking the average values of the confidence scores of the same-label patches rather than taking the summation of these scores.

Spanning measure

This measure is designed to give advantage to ally patches extracted at attack-clear regions over the patches extracted on or around the adversarial patch. The metric is inspired by the fact that suspicious patches tend to gather within a connected compact zone. Therefore, this measure weights the average confidence values by the ratio between the area of the inclusive square that encapsulates all the same-label patches and the patch area. So, the patches that are spread over a larger area in the image will have more weight than those which are condensed around small area and more probable to include parts of the adversarial patch.

Experimental setup

The main goal of the proposed ally patches is to stop threats created by targeted and source/target adversarial patches. Therefore, the main objective of experimentation in this work is to prove the effectiveness of the proposed ally patches in repelling the targeted attacks. Adversarial patches used to be evaluated using the success rate metric [3]. This metric evaluates the percentage of times in which an adversarial patch succeeded in forcing the network output to have a desired target attack label. We adopt the same metric to evaluate our approach by showing that using ally patches could significantly decrease the success rate of adversarial patches.

We use ILSVRC benchmark dataset [19] for input images. The dataset contains more than 1.2M images for training, 50K images for validation, and 150K images for testing. The images are collected from 1000 different distinct classes. We exposed the input images to three different adversarial patches generated in [3], as shown in Fig. 2. Each of these adversarial patches targets a specific false label. Specifically, they aim at "*Toaster*" and "*Crab*" labels. Three different pre-trained CNN's are used for white-box evaluation: VGG16 [22], VGG19 [22], and ResNet50 [10]. Comprehensive evaluation using ILSVRC benchmark was carried out on a single Nvidia GTX 1080-ti GPU. The host machine processor is Intel i7 with 32GB RAM. Tesnsorflow was used for CNN models evaluation. Cuda and C++ were used for dataset patching. A Matlab code for image ally patching is available on GitHub[1].

Input images are overlaid by each of the adversarial patches with different scales ranging from 10 to 50% of the input image areas and placed at random locations. The attacking algorithms which adopt adversarial patches attempt to make these patches visually unnoticeable as much as possible. This means that using patches larger than 50% of the

[1] https://github.com/aaaly/Ally-Patches.

input image is highly improbable. Therefore, we limited our conducted experiments to adversarial patch scales up to 50%. Ally patches are extracted, as explained in "Ally patch extraction" section. The extracted patches are classified using the considered three CNN models. Finally, the late fusion stage, using the aforementioned four strategies, is applied to decide a final image label.

Results and discussions

For evaluation purposes, we investigated the effect of ally patches on the main performance metric of adversarial patches, which is the success rate. Figure 4 shows the success rates obtained using the aforementioned setup. Ally patches have a close performance pattern for the used three models. Generally, ally patches succeeded in decreasing the success rates of adversarial patches to lie within the middle third of the scale instead of the top one. With small-scaled adversarial patches, the performance of adversarial patches is obviously low. So, the success rate reduction achieved by ally patches is not explicitly exhibited in such scales. Another factor that supports such behavior in small scales is that during the extraction stage of ally patches, small adversarial patches are most likely contained completely in one or more ally patch. So, a collection of ally patches will vote for the adversarial targeted label. Thus, the final decision taken by the fusion stage is affected. Fortunately, as the performance of the adversarial patches is low in such scales, they are not preferred by the attackers.

For fusion strategies, the performance of the used four fusion methods is close. The average confidence metric slightly shows better performance in large scales. In small scales, the majority voting metric preforms the best. This behavior is expected since in large scales the probability of partial appearance of the adversarial patch within fine-tuned patch set is higher. Consequently, the confidence of misclassifications will be decreased. Therefore, the average confidence gives an advantage to the patches that contain adversarial-clear areas from the original image. On the other side, in small scales, the probability of complete appearance of the adversarial patch in some of the fine-tuned ally patches set is higher. Hence, adversarial labels will dominate some patches with high confidence, which gives less preference to the average confidence metric in favor to other metrics like majority voting.

Conclusions

We presented ally patches as a defense approach against attacks to deep neural networks, which are caused by adversarial patches. Ally patches are extracted from the input image intrinsically based on the information contents of candidate patches. Entropy is utilized as a measure of the patch information contents. It is highly probable that the adversarial patch gets damaged during ally patch extraction causing either failure to deceive the network or at least missing its adversarial target label. Even if it was not damaged, the other extracted ally patches form an alliance which withstand the adversarial patch. Late fusion is performed to take a final classification decision according to the independent classifications of the ally patches. The evaluation results proved the ability of the proposed ally patches in bringing down the success rates of the adversarial patches by about one-third.

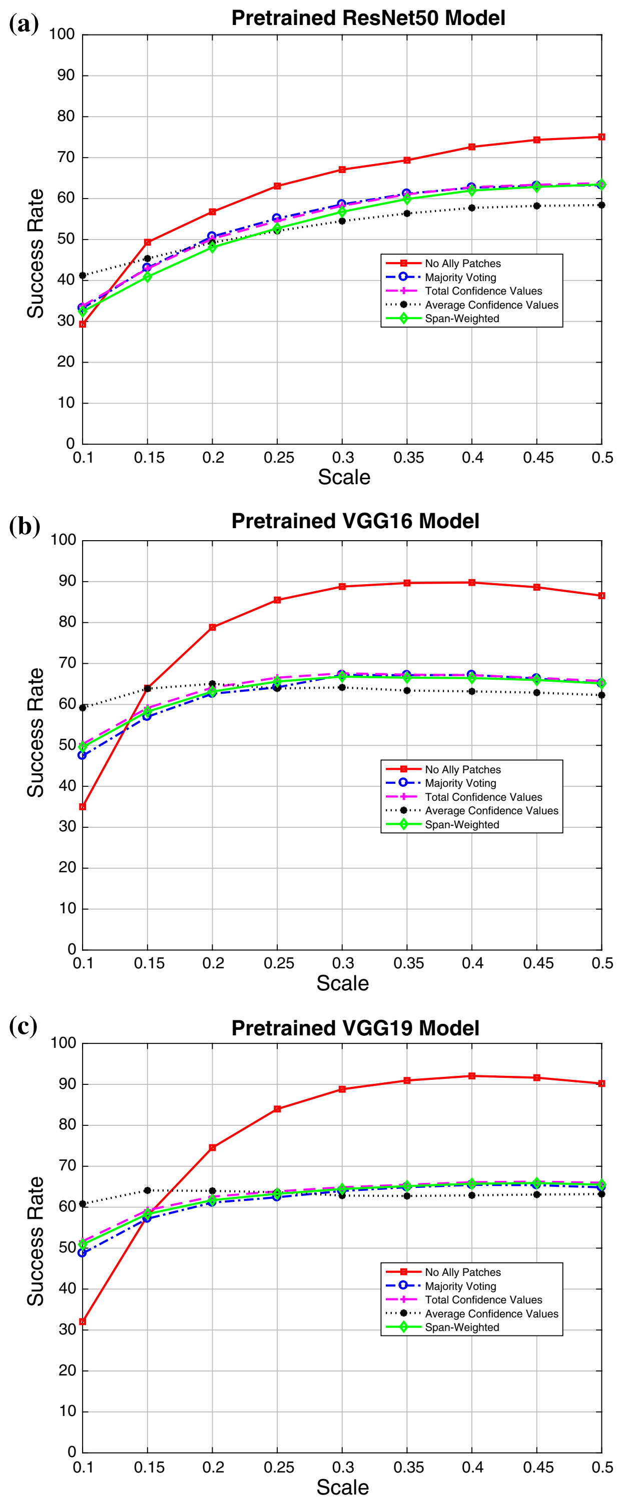


Fig. 4 The success rates before and after using the proposed ally patches with three different pre-trained CNN models: **a** ResNet50 [10], **b** VGG16 [22], and **c** VGG19 [22]

As a future direction of research, the number of degrees of freedom of ally patches can be increased for better inclusion of informative image parts. In this work, we used two degrees of freedom, which are the horizontal and vertical displacements of the candidate patches. This number can be increased to five by including up to three more degrees of freedom: horizontal scale, vertical scale, and 2D patch rotation angle. We expect better performance with more degrees of freedom, since the encapsulation of relevant image parts will be more precise.

Abbreviations

AP: ally patch; $\mathbb{AP}$: ally patch set; CNN: convolutional neural network; DNN: deep neural network; DSS: driver support system; ILSVRC: ImageNet Large Scale Visual Recognition Competition.

Acknowledgements
The author would like to thank the Deanship of Scientific Research at Umm Al-Qura University for the continuous support. This work was supported financially by the Deanship of Scientific Research at Umm Al-Qura University to Dr. Alaa E. Abdel-Hakim—(Grant Code: 17-CRW-1-01-0003).

Authors' contributions

This is a single author work. Therefore, the contribution of the author is 100% of the work. The author read and approved the final manuscript.

Funding
This work was supported financially by the Deanship of Scientific Research at Umm Al-Qura University to Dr. Alaa E. Abdel-Hakim—(Grant Code: 17-CRW-1-01-0003).

Availability of data and materials

Source code is available on: https://github.com/aaaly/Ally-Patches.

Competing interests

The author declares no competing interests.

Author details
[1] Electrical Engineering Department, Assiut University, Assiut 71516, Egypt. [2] Computer Science Department, Umm Al-Qura University, Jamoum, Saudi Arabia.

Received: 25 September 2018 Accepted: 29 May 2019
Published online: 07 June 2019

References

1. Athalye A, Sutskever I. Synthesizing robust adversarial examples; 2017. arXiv preprint arXiv:170707397.
2. Athalye A, Carlini N, Wagner D. Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples; 2018. arXiv preprint arXiv:180200420.
3. Brown TB, Mané D, Roy A, Abadi M, Gilmer J. Adversarial patch; 2017. arXiv preprint arXiv:171209665.
4. Buckman J, Roy A, Raffel C, Goodfellow I. Thermometer encoding: one hot way to resist adversarial examples. In: Submissions to international conference on learning representations; 2018.
5. Carlini N, Wagner D. Adversarial examples are not easily detected: bypassing ten detection methods. In: Proceedings of the 10th ACM workshop on artificial intelligence and security. New York: ACM; 2017. p. 3–14.
6. Carlini N, Wagner D. Magnet and efficient defenses against adversarial attacks are not robust to adversarial examples; 2017a. arXiv preprint arXiv:171108478.
7. Carlini N, Wagner D. Towards evaluating the robustness of neural networks. In: IEEE symposium on security and privacy (SP), 2017b. New York: IEEE; 2017. p. 39–57.
8. Evtimov I, Eykholt K, Fernandes E, Kohno T, Li B, Prakash A, Rahmati A, Song D. Robust physical-world attacks on deep learning models; 2017c. arXiv preprint arXiv:1707089451.
9. Goodfellow IJ, Shlens J, Szegedy C, Explaining and harnessing adversarial examples; 2014. arXiv preprint arXiv:14126572.
10. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 770–8.
11. Hendrycks D, Gimpel K. Early methods for detecting adversarial images; 2017. arXiv preprint arXiv:1608.00530.
12. Kurakin A, Goodfellow I, Bengio S. Adversarial examples in the physical world; 2016. arXiv preprint arXiv:160702533.
13. Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A. Towards deep learning models resistant to adversarial attacks; 2017. arXiv preprint arXiv:170606083.
14. Meng D, Chen H. Magnet: a two-pronged defense against adversarial examples. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. New York: ACM; 2017. p. 135–47.
15. Metzen JH, Genewein T, Fischer V, Bischoff B. On detecting adversarial perturbations; 2017. arXiv preprint arXiv:170204267.

16. Moosavi Dezfooli SM, Fawzi A, Frossard P. Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of 2016 IEEE conference on computer vision and pattern recognition (CVPR), EPFL-CONF-218057; 2016.
17. Papernot N, McDaniel P, Jha S, Fredrikson M, Celik ZB, Swami A. The limitations of deep learning in adversarial settings. In: IEEE European symposium on security and privacy (EuroS&P), 2016. New York: IEEE; 2016. p. 372–87.
18. Russakoff DB, Tomasi C, Rohlfing T, Maurer CR. Image similarity using mutual information of regions. In: European conference on computer vision. New York: Springer; 2004. p. 596–607.
19. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L. ImageNet large scale visual recognition challenge. Int J Comput Vis. 2015;115(3):211–52. https://doi.org/10.1007/s11263-015-0816-y.
20. Shannon CE, Wyner A, Sloane NJ. Claude E. Shannon: collected papers. New York: Wiley; 1993.
21. Sharif M, Bhagavatula S, Bauer L, Reiter MK. Accessorize to a crime: real and stealthy attacks on state-of-the-art face recognition. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. New York: ACM; 2016. p. 1528–40.
22. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition; 2014. arXiv preprint arXiv:14091556.
23. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R. Intriguing properties of neural networks. In: International conference on learning representations; 2014.
24. Zantedeschi V, Nicolae MI, Rawat A. Efficient defenses against adversarial attacks. In: Proceedings of the 10th ACM workshop on artificial intelligence and security. New York: ACM; 2017. p. 39–49.

Publisher's Note