IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 21 November 2022, accepted 27 January 2023, date of publication 1 February 2023, date of current version 7 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3241360

RESEARCH ARTICLE

Rearranging Pixels Is a Powerful Black-Box Attack for RGB and Infrared Deep Learning Models

JARY POMPONI[1], **(Graduate Student Member, IEEE), DANIELE DÁNTONI**[1], **NICOLOSI ALESSANDRO**[2], **AND SIMONE SCARDAPANE**[1]

[1]Department of Information Engineering, Sapienza University of Rome, 00185 Rome, Italy
[2]Leonardo Labs, 00195 Rome, Italy

Corresponding author: Jary Pomponi (jary.pomponi@uniroma1.it)

ABSTRACT Recent research has found that neural networks for computer vision are vulnerable to several types of external attacks that modify the input of the model, with the malicious intent of producing a misclassification. With the increase in the number of feasible attacks, many defence approaches have been proposed to mitigate the effect of these attacks and protect the models. Mainly, the research on both attack and defence has focused on RGB images, while other domains, such as the infrared domain, are currently underexplored. In this paper, we propose two attacks, and we evaluate them on multiple datasets and neural network models, showing that the results outperform others established attacks, on both RGB as well as infrared domains. In addition, we show that our proposal can be used in an adversarial training protocol to produce more robust models, with respect to both adversarial attacks and natural perturbations that can be applied to input images. Lastly, we study if a successful attack in a domain can be transferred to an aligned image in another domain, without any further tuning. The code, containing all the files and the configurations used to run the experiments, is available https://github.com/jaryP/IR-RGB-domain-attackonline.

INDEX TERMS Deep learning, infrared, adversarial attack, black-box, adversarial training, robustness.

I. INTRODUCTION

Neural networks (NNs) based systems have become state of the art in multiple fields, and therefore have become targets for adversarial attacks that try to break the system by modifying the input [1]. This goal can be achieved simply by modifying only a small portion of the NN input (usually an image), even if it represents a real-world object [2]. This vulnerability must be addressed if we want to build agents that operate in real-world scenarios.

Most of the approaches used to attack NNs generate attacks by exploiting a constrained optimization problem, in which the NN must be fooled while keeping the distance between the original image and the adversarial one minimized, in order to keep the adversarial image as close as possible to the original one, while fooling the model. The constraint is applied to the distance between the original images and the modified version used to fool the model, called adversarial image, and it is calculated using the L_p norm of the distance between these images; the choice of the norm highly influences the overall algorithm, by changing how many pixels, and how, are modified by the attack. In addition to the norm used, we can group the attacks into two sets: if the attack needs to access the internal state of a NN in order to produce an adversarial image, or not. In the first case, the attack is called white-box, while in the second case it is called black-box [3].

With the increase in the number of feasible attacks, it is crucial to study methods to defend NN-based systems, especially if they operate in a real-world scenario. An approach to do that is by training the NN in a way that the resulting model is robust to external attacks, by injecting attacked images into the training dataset [4]. Usually, this approach makes the model more robust only to certain types of attacks, which are similar to the attack used during the training phase. Another

The associate editor coordinating the review of this manuscript and approving it for publication was Nuno Garcia.

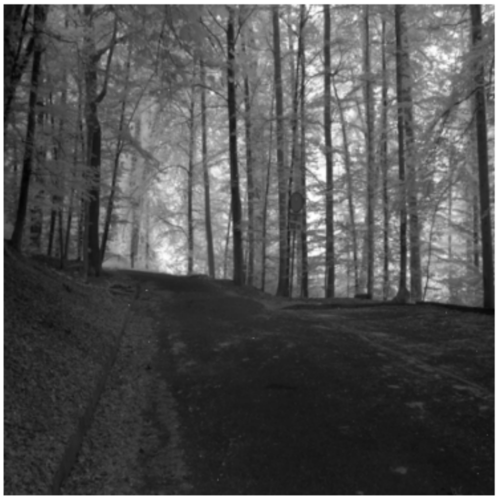

FIGURE 1. Successful adversarial image from the NIR Scene Dataset using the proposed Wixle attack. The L_0 distance between the original image and the adversarial one is just 32 pixels (corresponding to ∼ 0.02% of the pixels in the image).

way of protecting a system is by implementing a mechanism that can detect and discard malicious inputs before classifying them [5], [6].

Most of the attacks and defences focus on computer vision systems for RGB images in the visible spectrum, due to the high number of benchmarks and pre-trained models available. In this paper, instead, our focus goes beyond the standard RGB domain, and we study how the proposed attacks and defences perform on infrared images, which are commonly used in practice (e.g., for monitoring applications), but severely underexplored in the advarsarial attack literature. An infrared image differs from an RGB one because it has a single channel instead of three, it contains no information about the texture of the subject, and it is grey-scaled; an example of infrared image attacked by our method is shown in Figure 1. The infrared is part of the electromagnetic spectrum and it is imperceptible to the human eye, and thus can be used to extract hidden and highly informative features. This latter aspect is used in many applications in which working with visible light is not enough, such as satellite monitoring [7] and image classification and segmentation [8], [9].

In this paper, we propose two L_0 attacks: Pixle, which is a black-box attack based on random search, and a white-box version of Pixle, called Wixle.[1] We evaluate these attacks on different combinations of benchmarks, both on RGB and infrared domains, and architectures. The latter attack can also be used to build more robust models, which can resist better to black-box attacks as well as natural perturbations. In the experimental section, we also study the transferability of an attack from one domain (e.g., RGB) to another (e.g., infrared), which is carried out by attacking an image from a source domain, and then using the same attack to try to fool also the aligned image in the other domain, called target domain, without performing further searches for the best adversarial image.

In this paper, we show that black-box attacks are capable of attacking also models trained on infrared images. We show also that, in general, models trained on such images are more robust to attacks that create adversarial images by injecting colored pixels into the image. We also prove that, by using our white-box proposal in an adversarial training schema, we are capable of creating more robust models, both in terms of black-box attacks as well as natural corruptions of the images. In the end, we study if an attack can be transfer from one domain to another, without having any information about the destination one.

The paper is organized as follows. In Section II we discuss related researches that have been done in the attack and defense domain, both for infrared and RGB domains. Section III presents the definitions of adversarial attacks and natural attacks, as well as the exposition of the proposed attacks. In the end, Section IV contains all the results. Finally, in Section V general conclusions are drawn.

II. RELATED WORKS

The first papers about adversarial attacks were introduced in the context of data mining and spam filtering [11], [12], [13], while the first machine learning model that was successfully attacked was the Support Vector Machine [14]. Later on, the authors of [1] and [15] showed that NNs are also prone to such attacks. After these studies, the security of machine learning based agents became a crucial aspect to study in order to create more robust models.

Over the years, many methods to fool NNs and defend them have been proposed, mostly operating in the RGB domain. The attacks can be categorized based on how the images are modified, by analyzing the norm of the difference between the image and the adversarial counterpart, and also based on which information the approach needs to correctly attack an image. If access to the internal state of the network is needed we have a white-box attack, otherwise, we have a black-box attack. Regarding how the images are modified, the most studied approaches are based on L_∞ or L_2 norms, which usually modify all the pixels in the image using a small noise, with L_0 norm being the less studied set of attacks. In this paper, we study attacks based on the latter norm.

One of the first approaches to attack an image following a L_0 approach is OnePixel [16], which aims to find the best pixel to overwrite using the Differential Evolution search algorithm [17]. This approach works well on small images but struggles to attack bigger ones because it requires thousands of iterations, and the number of pixels to attack must be selected before the attack and cannot be tuned while searching for the adversarial image. Following the same principle, in [18] the authors proposed an approach that tries to place patches on the images using a reinforcement learning approach; the main drawback of it is that the

[1] A preliminary version of the black-box variant appeared in [10]. Compared to [10], we significantly extend the treatment by considering white-box variants of Pixle, adversarial defences and mitigations, and the efficiency of these attacks in the infrared domain.

patches are clearly visible, hence easily detected. Lastly, ScratchThat [19], also based on differential evolution search, is an approach that aims to attack an image by literally scratching it by applying lines and curves of different colours on the pixels.

The attacks on images in the infrared domains are more practical but less studied. In [20] the authors proposed an approach to fool face recognition systems by placing an infrared light on around the subject. In the same context, the authors of [21] proposed an approach to fool thermal images and near-infrared images, based on 3D masks. In [22], the authors studied how attacks perform on infrared aerial images, captured by drones, and how to build more robust models. To the best of our knowledge, no work in the literature has explored general black-box attacks for infrared models, or studied the transferability of these attacks from the visible light spectrum to the infrared domain.

For a complete review of Adversarial Attacks and robustness refer to [23], [24] and [25].

III. METHODOLOGY

A. PRELIMINARIES

1) FOOLING NEURAL NETWORKS

The goal of fooling a neural network is to take an image that the model correctly classifies and modify it so that the model miss-classifies it. This problem can be seen as an optimization problem with constraints, where the constraints depend on how the images are corrupted.

Let $f : x \rightarrow \mathbf{p} \in \mathbb{R}^Y$ be a function that takes as input an image and return the associated probability for each possible class (such that $\sum_i^Y \mathbf{p}_i = 1$). In our case, the function f is a trained neural network, and the classification is carried out by taking the class with the highest associated probability: $c(x) = \arg\max_i p_i(x)$, where $p_i(x)$ returns the probability associated to the class i.

Given an image x, with its associated label y, that is correctly classified by the model, our goal is to produce an adversarial image $\overline{x}$ such that $c(\overline{x}) \neq y$. In order to be credible, the adversarial image can not be completely different from the original one, otherwise, the artefacts injected would be too visible and easily avoided by a defence algorithm. To this end, the distance between the original image and the adversarial one must be constrained:

$$c(\overline{x}) \neq y \ \text{ s.t. } \ ||x - \overline{x}||_l \leq \epsilon \tag{1}$$

where the choice of l determines the attack typology. In our case we consider $l = 0$, i.e., the number of different pixels between the original image x and the adversarial attack, meaning that $\epsilon \in \mathbb{N}_+$ is the number of maximum pixels that can be modified by the algorithm. The task of finding the perturbed image $\overline{x}$, associated to x, can be viewed as a minimization problem:

$$\min_{\overline{x}} L(\overline{x}, y) \ \text{ s.t. } \ ||x - \overline{x}||_l \leq \epsilon\,, \tag{2}$$

for a proper loss function L. In this paper, since we want to minimize the confidence associated to the correct label, we use $L(\overline{x}, y) = p_y(\overline{x})$. We note that the loss function L is agnostic with respect to the state of the model, and only the input and the output of the model are required to calculate it, hence it is a valid loss also for black-box attacks.

2) ROBUSTNESS

Neural networks can be easily fooled, and thus having robust models is a desirable property to have for agents operating in a real-world scenario. Realistically, it is unlikely that an attack method can access the internal state of a model and use it to perturb the input image, hence in this paper we focus mostly on robustness with respect to a set of attacks that are considered natural, as well as black-box attacks.

Regarding adversarial attacks, we aim to make models more robust by injecting adversarial samples, generated using a L_0 attack, into the training set. By doing so, we expect the model to be more robust to attacks which operate on a pixel level and that try to change pixels in the image, instead of attacks that modify all the pixels by injecting noise.

Regarding the natural corruptions, in [26] the authors proposed a set of corruptions that are called natural and can be applied to the image before interacting with the model, such as Gaussian Noise, blur, and contrast. We formalize the robustness with respect to these attacks following the same formulation proposed in [26]. As before, we have a trained neural network defined as a function f, and we also have a set of corruption functions C, in which each function approximates the real-world frequency of the same corruption. Using this setting, we measure the robustness of a model on a sample (x, y) as:

$$\mathbb{E}_{\zeta \sim C}\left[\mathbb{P}_{(x,y)\sim D}\left[c(\zeta(x)) = y\right]\right] \tag{3}$$

where D is the dataset from which the samples are drawn. This is in contrast with the concept of adversarial robustness introduced before, because corruption robustness measures the classifier's average-case performance on a set of corruptions C, while adversarial robustness measures the worst-case performance on a small perturbation generated for the current image.

B. PIXLE

In this Section we propose Pixle, a black-box attack based on random search, that does not depend on gradient information or the internal state of the model.

Given an image x, the attack samples a patch of adjacent pixels from it and rearranges them into the image, by copying the values into other random positions. A generic patch is a 4-tuple $p = (o_x, o_y, w_p, h_p)$, where $0 < o_x \leq w$ and $0 < o_y \leq h$ are the coordinates on the image used as the origin point of the patch, and w and h are, respectively, the width and the height of the image. The set of coordinates of the pixels in the patch is defined as $P = \left[(o_x + i, o_y + j)\right]_{\forall i \in \{0,\dots,w_p\}, j \in \{0,\dots,h_p\}}$, which has size $|P| = w_p \cdot h_p$ (if a position exceeds the dimension of the image it is discarded).

The proposed Pixle algorithm is composed of a fixed number of restarts $R \geq 1$, and within each restart, a maximum

Algorithm 1 Pixle Algorithm

```
Require: input image $x$ with its associated label $y$. Maximum and minimum dimensions for source patch $w_p$ and $h_p$. The number of restarts $R$ and the iterations to perform for each restart step $I$. A function $m(x)$ that returns a random position in the image.
$\overline{x} \leftarrow x$
$l \leftarrow L(x, y)$
for $r = 1$ to $R$ do
    $x^r \leftarrow none$
    for $i = 0$ to $I$ do
        Sample $p = (o_x, o_y, w_p, h_p)$.
        Calculate the set $P$
        $x^i \leftarrow \overline{x}$
        for $\forall(i, j) \in P$ do
            $(z, k) \leftarrow m(x)$
            $x^i_{z,k} \leftarrow x_{i,j}$
        end for
        if $L(x^i, y) < l$ then
            $l \leftarrow L(x^i, y)$
            $x^r \leftarrow x^i$
        end if
    end for
    if $x^r$ is none then
        return $\overline{x}$
    end if
    $\overline{x} \leftarrow x^r$
end for
return $\overline{x}$
```

number of iterations I are performed. At every iteration, it samples a source patch p and the set P is calculated, then the pixels in the set are copied into random positions of a proxy image which is equal to the image at the beginning of the restart step, to avoid sub-optimal attacks. If this rearrangement of pixels produces a loss value which is lower than the best one obtained so far, the image is the new adversarial candidate and the associated loss becomes the new loss to beat. After the last iteration step, if an image decreased the loss, then it becomes the adversarial image and it is used in the next restart step, otherwise, the last adversarial image is returned. When the number of restart steps is reached, the algorithm returns the last adversarial image found. The algorithm is summarized in Alg. 1.

C. WIXLE

Wixle is a white-box version of Pixle, also based on random search, in which the gradient values of $f(x)$ are used to sample the pixels to attack according to the gradient value associated with each one, which is considered directly proportional to the importance of the pixel itself. We introduce this attack mainly to perform faster and more impacting attacks while training the models in an adversarial training scenario, i.e., we use the white-box variant Wixle as a proxy for the true black-box attack Pixle, which is generally unfeasible for adversarial training to the need of performing a random search over possible pixel rearrangements. The use of the gradients in Wixle associated with the pixels reduces the number of necessary iterations needed to attack an image, thus allowing for a faster and better selection of the pixels to move, which is crucial in order to decrease the time required to find the adversarial image.

Given an image x and the gradient value $|g|$ for each pixel of x, averaged over the channels, the attack randomly samples a subset of source pixels, giving more importance to pixels with a higher gradient value, and copies them into the location of the destination pixels, sampled using the same approach but using as sampling probability the inverse of the gradient value. The gradient values are calculated using the cross entropy classification loss, as proposed in [15]. In order to sample the positions of the pixels, we define two different distributions, where the probability of the position (i, j) to be sampled is given by:

$$P^s(X = (i, j)) = \frac{g_{i,j}}{\sum g} \tag{4}$$

$$P^d(X = (i, j)) = \frac{g^{-1}_{i,j}}{\sum g^{-1}} \tag{5}$$

where P^s gives the probability for the source pixels and P^d for the destination ones, and $g_{i,j}$ is the gradient value associated to the pixel in position (i, j). The distributions used to sample, respectively, source and destination positions, are $S(g)$ and $D(g)$.

The algorithm is composed of a fixed number of restarts R, and within each one, a maximum number of iterations I are performed. At the beginning of each restart step, the attack calculates the gradients associated with each pixel in the current proxy image x^r, using the cross-entropy loss, as $g = \frac{1}{ch} \sum_{ch} |\nabla_{x^r} \text{CrossEntropy}(f(\overline{x}), y)|$, where ch is the number of channels in the image. For every iteration step i in the current restart, the source position $(s_i, s_j) \sim S(g)$ and the destination position $(d_i, d_j) \sim D(g)$ are randomly sampled, then, the pixel in the destination position is overwritten with the one in the source position. This is done to a proxy image x^i associated with the current iteration step, to avoid changing the images with sub-optimal attacks, as done also in Pixle. The rest of the algorithm is the same used for Pixle: if the loss associated with x^i is lower than the one calculated at the beginning of the current restart, the image is saved and the new loss to beat is the current one, otherwise, it is discarded. After the last iteration, if a proxy image is saved, it becomes the new image to attack at the next restart step, otherwise, the current image is the final adversarial image $\overline{x}$ and it is returned.

By changing one pixel per restart the distance norm between the adversarial image and the attacked one is minimized, but it can be expensive in the number of times that the function f is called. To mitigate this aspect, multiple pixels can be changed at each iteration step, by sampling a set of source pixels and a set of destination pixels, having the same size, and using the same approach described above by iterating both sets at the same time (the positions from

Algorithm 2 Wixle Algorithm

```
Require: input image $x$ with its associated label $y$. The number of restarts $R$ and the iterations to perform for each restart step $I$.
Ensure: $y = x^n$
  $\bar{x} \leftarrow x$
  $l \leftarrow L(x, y)$
  for $r = 0$ to $R$ do
      $x^r \leftarrow$ none
      $g = \mathbb{E}_c|\nabla_x \text{CrossEntropy}(f(\bar{x}), y)|$
      for $i = 0$ to $I$ do
          $x^i \leftarrow \bar{x}$
          Sample $(s_i, s_j) \sim S(g)$.
          Sample $(d_i, d_j) \sim D(g)$.
          $x^i_{d_i,d_j} \leftarrow x_{s_i,s_j}$
          if $L(x^i, y) < l$ then
              $l \leftarrow L(x^i, y)$
              $x^r \leftarrow x^i$
          end if
      end for
      if $x^r$ is none then
        return $\bar{x}$
      end if
      $\bar{x} \leftarrow x^r$
  end for
  return $\bar{x}$
```

each distribution must be sampled without replacement). The complete algorithm is summarized in Alg. 2.

Adversarial Training: we introduce Wixle as a faster version of Pixle, which is capable of attacking an image using more precise pixel rearrangements, and thus it is more suitable to be used in an adversarial training schema. The procedure is the following: when a batch is collected, a subset of it is attacked but, since the attack requires many inferences in order to find the correct adversarial image, we relax the problem, by using our approach as a one-shot attack. For each image to attack, the percentage of pixels to move is randomly sampled using $p = \text{unif}(l, h)$, where l and h are, respectively, the lowest and the higher percentage of pixels that can be moved in an image. The sampled value is converted to an integer using the actual dimension of the image, and then the pixels' positions are sampled as before and moved at the same time. The idea is to attack an image in a different way each time it is encountered during the training process, forcing the model to learn how to classify multiple attacked versions of the same image. To the best of our knowledge, Wixle is the first L_0 attack to be used in an adversarial training schema.

IV. EXPERIMENTAL EVALUATION

A. SETUP

1) ADVERSARIAL EXPERIMENTAL SETUP

The evaluation of the proposed attacks is carried out on CIFAR10 [27] and ImageNet [28]. Regarding the first we attack, using SGD with learning rate equals to 0.01 and 0.9 as momentum, VGG11 [29] and ResNet-20 [30], while regarding the latter we use ResNeXt-50 [31] and ConvNeXt-Tiny [32], both using pre-trained weights without fine-tuning.

We also use the RGB-NIR Scene dataset proposed in [33], which is composed of 477 images in 9 categories, and each image has both RGB and Near-infrared (NIR) versions. We train the same models used for ImageNet on both RGB and NIR sets of images, but we resize the images to have a size of 420 per side. The training procedure is the same used for CIFAR10, and the test subset is composed of 10% of the images in the dataset.

For each experiment, we attack only correctly classified images from the test set of the dataset. We attack 100 images for each class present in CIFAR100, and 1 for each class in ImageNet. In this way, we have the same number of attacked images. Regarding RGB-NIR Scene, the number of images per class is lower, so we attack them all.

We compare our proposals with two other L_0 attacks: ScratchThat [19], which attacks the images by drawing lines and curves on the images, and OnePixel [16], which overwrites a variable number of pixels with randomly coloured pixels. Both ScratchThat and OnePixel are based on the Differential Evolution search algorithm [17].

To compare the attacks, we use the following metrics:

- Success Rate: the percentage of images that are correctly attacked (the ones that are miss-classified after the attack).
- Iterations: the number of times that the model is interrogated while attacking a given image.
- L_0 norm: the distance between the original image and the adversarial one.

Regarding the parameters of each attack, we performed a grid search, following the results from the respective papers, over ResNet-20 trained on CIFAR10. For OnePixel we set to 5 the number of pixels to attack. We use a Bézier curves approach for ScratchThat, drawing 1 curve for CIFAR10, and 2 curves to attack the other datasets; the differential evolution parameters are the same as in the respective papers. Regarding Pixle, at each iteration, we randomly sample a patch having size 3 for CIFAR10 and 1% of the attacked image size for the others. The number of restarts is set to 100, and for each restart step, we perform up to 20 iterations. In the end, we attack one single pixel per restart iteration when attacking using Wixle, and we perform up to 100 restarts and up to 50 iterations per restart step. For each method, a callback is used to interrupt the attack when an adversarial image that correctly fools the model is found. Except for Wixle, we use the attacks as implemented in TorchAttack [34] package.

2) ADVERSARIAL TRAINING

To evaluate the efficiency of Wixle used in the context of adversarial training we test it on CIFAR10, classified using ResNet-20 [30], and Scene dataset, classified using ResNeXt-50 [31]. For each experiment, we perform a pre-training step in which the model is trained on the

TABLE 1. Results obtained when attacking multiple datasets trained on ResNeXt and ConvNeXt architectures. For each score, we show the mean and the variance, if present, calculated over all the images on that dataset.

Dataset	Model	Test score ↑	Method	Success rate % ↑	Iterations ↓	L_0 norm ↓
CIFAR10	ResNet-20	85.7	OnePixel	59.0	5100	5
			ScratchThat	93.5	$176_{\pm 282}$	$27_{\pm 5}$
			Pixle	**100**	$45_{\pm 52}$	$19_{\pm 14}$
			Wixle	100	$30_{\pm 24}$	$18_{\pm 14}$
	VGG11	80.7	OnePixel	67.4	5100	5
			ScratchThat	92.7	$157_{\pm 282}$	$26_{\pm 5}$
			Pixle	100	$49_{\pm 86}$	$27_{\pm 33}$
			Wixle	96.6	$410_{\pm 1818}$	$19_{\pm 25}$
ImageNet	ResNeXt-50	77.6	OnePixel	8.10	5100	5
			ScratchThat	38.1	$1400_{\pm 854}$	$95_{\pm 4}$
			Pixle	89.1	$663_{\pm 528}$	$538_{\pm 395}$
			Wixle	95.0	$650_{\pm 674}$	$249_{\pm 218}$
	ConvNeXt-T	82.5	OnePixel	0	5100	5
			ScratchThat	38.2	$1404_{\pm 854}$	$96_{\pm 3}$
			Pixle	62.5	$1309_{\pm 659}$	$1095_{\pm 543}$
			Wixle	100	$1512_{\pm 1182}$	$744_{\pm 555}$
NIR Scene	ResNeXt-50	78.7	OnePixel	8.1	5100	5
			ScratchThat	10.8	$1855_{\pm 539}$	$98_{\pm 3}$
			Pixle	89.1	$663_{\pm 528}$	$528_{\pm 395}$
			Wixle	100	$650_{\pm 674}$	$249_{\pm 218}$
	ConvNeXt-T	68.1	OnePixel	0	5100	5
			ScratchThat	15.6	$1825_{\pm 518}$	$97_{\pm 5}$
			Pixle	62.5	$1309_{\pm 659}$	$1095_{\pm 543}$
			Wixle	100	$1512_{\pm 1182}$	$744_{\pm 555}$
RGB Scene	ResNeXt-50	89.4	OnePixel	7.1	5100	5
			ScratchThat	16.6	$1742_{\pm 643}$	$98_{\pm 4}$
			Pixle	78.5	$910_{\pm 657}$	$689_{\pm 479}$
			Wixle	100	$1099_{\pm 1300}$	$354_{\pm 368}$
	ConvNeXt-T	74.4	OnePixel	0	5100	5
			ScratchThat	5.71	$1932_{\pm 372}$	$98_{\pm 3}$
			Pixle	25.7	$1676_{\pm 428}$	$1388_{\pm 352}$
			Wixle	100	$3544_{\pm 2610}$	$1510_{\pm 1143}$

dataset, and then we perform the adversarial training step, in which the pre-trained model is trained for additional epochs using Wixle to attack the images in the dataset, as exposed in Section III-C. We trained all the models during the pre-training step for 50 epochs, and then we perform 20 epochs using adversarial training. For each training step, we use SGD with a learning rate equal to 0.01 and 0.9 as momentum, as before.

Regarding Wixle parameters, we attack each image in a batch with a probability of 0.5%, and for each image, a random number of pixels, that varies from 5% to 40% of the total number of pixels in the image, is moved just once, without searching for the best attack (we use one restart and one iteration for each image).

To evaluate the robustness against Black-box attacks, we test the model before the adversarial training and after, using the same metrics exposed before.

We also test if this approach is suitable to improve the robustness with respect to natural corruptions. To this end, we test the model trained on CIFAR10 using the Corrupted CIFAR10 (C-CIFAR10) [35] dataset, which is the test split of CIFAR10, but 15 different corruptions are applied to each image, and each corruption has 5 levels of severity.

To evaluate the performances on a specific corruption c, we use the Corruption Error (CE) metric (proposed in [35]), computed using the formula:

$$CE(x)^f_c = \frac{\sum_{s=1}^{5} E^f_{c,s}(x)}{\sum_{s=1}^{5} E^b_{c,s}(x)} \tag{6}$$

where f is the neural network trained using the adversarial training as proposed before, g is a neural network pre-trained on CIFAR10 (in our case it is the same network as f but before the adversarial training), and $E^i_{c,s}(x)$ is the top-1 error achieved by the model i on images corrupted using corruption c, having severity s. The metric tells us how much a model is fooled by corruption, and thus lower is better: if the score is lower that 1, then the model robustness is improved with respect to the one achieved using the pre-trained model, otherwise it is worse or the same (if it is precisely 1). In addition to this metric, we also study how the accuracy metric evolves during the adversarial approach.

3) TRANSFERABILITY

To perform the transferability experiments we train two ResNeXt-50 [31] on RGB and NIR images of Scene Dataset [33]. Each model is randomly initialized, and the training procedure is the same as exposed before, but the training epochs are 100.

The approach is the following: we have a source domain and a target domain, each one with its trained model; once the models are trained, we attack the image from the source domain using Wixle, then, if the attack is successful and the model is fooled, we use the same sequence of pixels movements also in the image from the target domain. To study the feasibility of the approach, we simply use the Success rate metric.

B. ATTACKS RESULTS

The main results are presented in Table 1, which shows that our proposals achieve a higher success rate across all the combinations of datasets and networks. In particular, Wixle achieves a perfect success rate by modifying a contained number of pixels. It is interesting to be noted that ConvNeXt-T is a much more robust model when it comes to black-box attacks. In fact, OnePixel fails each time this model is used, and ScratchThat achieves a lower score if compared to the one obtained on the ResNeXt trained on the same dataset. By analyzing the number of iterations we observe that our approaches require a lower number of iterations if compared to other attacks that rely on the DE algorithm. Also, Wixle results suggest that there are images that are easily classified by moving a single pixel, and images which require more iterations, as shown by the standard deviation of the results. In the end, the L_0 norm tells us that our proposals are competitive since a small percentage of the attacked images is corrupted. Also here, we have a high standard deviation, telling us that some images are more difficult to attack than others.

We conclude our analysis by hypothesizing that our approaches are capable of finding a suitable adversarial image because the research space is not constrained or bounded in the research of the best pixels to attack, because of the double iteration performed by the random search, which allows the attacks to explore a wider attack space.

1) RANDOM SEARCH PARAMETERS

Table 2 contains the results obtained when changing the parameters of Pixle and Wixle. The results, obtained by attacking ResNet-20 trained on CIFAR10, with a fixed number of restarts equal to 100, gives us some interesting insight into the two approaches. First of all, Pixle is capable of achieving a success rate equal to 100% using each combination of parameters, while Wixle fails when the number of moved pixels grows. This probably happens because gradient values are computed after each restart step, and by moving a large number of pixels some movements can nullify past changes. in support of this claim, we have that also the L_0 values and iterations required are much higher than the ones achieved using the same parameters in Pixle. Furthermore, Wixle is capable of achieving complete success when it moves just one pixel in each restart step. Also, by moving just one pixel, the number of required iterations is lower than the corresponding number achieved by Pixle when the same amount of pixels are moved.

TABLE 2. The results obtained while varying the parameters of Pixle and Wixle attacks. The attacked model is ResNet-20, trained on CIFAR10. For each experiment, we use a number of restarts equal to 100. The pixel percentage is the percentage of pixels with respect to the image size (32 × 32) that are moved at each iteration step.

Attack	Pixels per iteration	Iterations	Success rate % ↑	Iterations ↓	L_0 norm ↓
Wixle	1	20	100	$41_{\pm 38}$	$18_{\pm 14}$
		100	100	$51_{\pm 314}$	$19_{\pm 15}$
	1%	20	100	$17_{\pm 38}$	$59_{\pm 49}$
		100	100	$18_{\pm 40}$	$59_{\pm 51}$
	10%	20	95.8	$20_{\pm 78}$	$189_{\pm 146}$
		100	99.6	$115_{\pm 757}$	$197_{\pm 163}$
	20%	20	94	$23_{\pm 88}$	$272_{\pm 165}$
		100	97.6	$327_{\pm 1611}$	$282_{\pm 184}$
Pixle	1	20	100	$126_{\pm 109}$	$21_{\pm 5}$
		100	100	$392_{\pm 347}$	$19_{\pm 3}$
	1%	20	100	$126_{\pm 108}$	$23_{\pm 5}$
		100	100	$393_{\pm 354}$	$24_{\pm 3}$
	10%	20	100	$48_{\pm 68}$	$43_{\pm 25}$
		100	100	$142_{\pm 176}$	$48_{\pm 14}$
	20%	20	100	$34_{\pm 84}$	$66_{\pm 64}$
		100	100	$82_{\pm 220}$	$50_{\pm 39}$

We can conclude that, if we want an attack which is very precise, we can use Wixle with a number of pixels set to 1, while if we want an attack which moves more pixels at the same time, and thus with a higher L_0 score, we can use Pixle with a patch dimension which contains 1% of the total pixels in the image or more.

C. ADVERSARIAL TRAINING RESULTS

In this section, we analyze both the robustness results obtained on black-box attacks as well as natural corruptions.

Table 3 shows the results obtained on CIFAR10 and NIR Scene, both before and after the adversarial training. We can see that most of the approaches are capable of attacking the pre-trained model on CIFAR10, but after the adversarial training, the success rate decreases for both OnePixel and ScratchThat. More importantly, the number of iterations required to attack an image and the final L_0 norm are both worse with respect to the results associated with the pre-trained model: even if the success rate is unchanged, Pixle requires, on average, five times the number of iterations and more than the triple of pixels are modified, while ScratchThat requires three times the number of iterations to find the adversarial image, with a standard deviation that is also double. This means that the model is more robust and can resist better to various black-box attacks which are different from the one used in the adversarial procedure. Regarding NIR Scene, the attacks struggle to successfully attack the model (as exposed before), and after the adversarial training, all metrics are worse: Pixel is not able to reach the same success rate, it requires more iterations, and the norm is higher, while OnePixel loses the ability to attack any image, and also the success rate of ScratchThat decreases.

The results associated with the natural corruptions robustness are shown in Figure 2, which shows the results associated with the metric in Eq. 6, and Figure 3, which shows the accuracy results obtained on each severity level. By analyzing the first one we can see that adversarial training improves

TABLE 3. The results obtained using Wixle attack to perform adversarial training. For CIFAR10 we use ResNet-20, while for NIR scene we use ResNeXt-50.

Dataset	Training step	Method	Success rate % ↑	Iterations ↓	L_0 norm ↓
CIFAR10	Pre-training	OnePixel	68.4	5100	5
		ScratchThat	96.7	$116_{\pm 220}$	$26_{\pm 5}$
		Pixle	**100**	$31_{\pm 34}$	$20_{\pm 14}$
	Adv. training	OnePixel	14.2	5100	5
		ScratchThat	71.3	$439_{\pm 416}$	$27_{\pm 5}$
		Pixle	100	$143_{\pm 107}$	$68_{\pm 44}$
NIR Scene	Pre-training	OnePixel	5.26	5100	5
		ScratchThat	21.0	$1774_{\pm 595}$	$97_{\pm 5}$
		Pixle	63.1	$1205_{\pm 671}$	$945_{\pm 522}$
	Adv. training	OnePixel	0.0	5100	5
		ScratchThat	5.26	$1985_{\pm 147}$	$94_{\pm 6}$
		Pixle	10.5	$1865_{\pm 112}$	$1550_{\pm 90}$

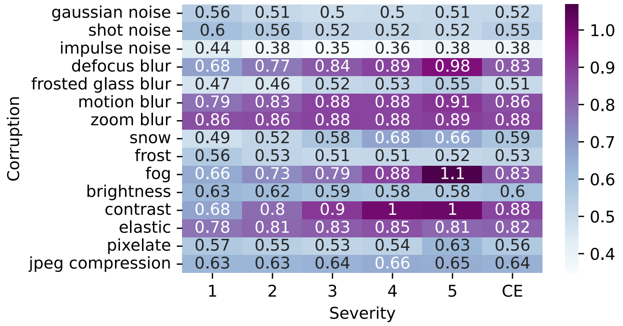

FIGURE 2. Visualization of the corruption errors (CE_c) for each combination of corruption and severity, obtained when training ResNet-20 on CIFAR10 using an adversarial approach based on Wixle. The last column of each corruption contains the average score obtained across all the severity. Lower is better.

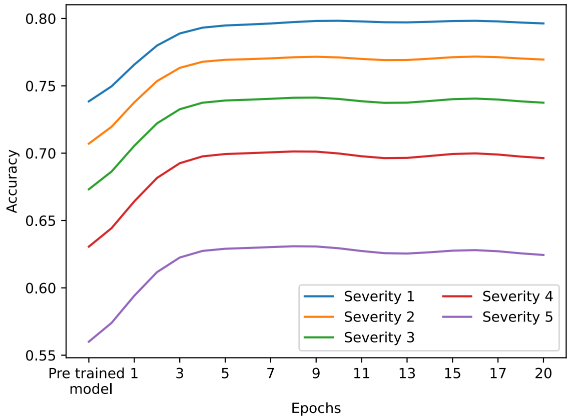

FIGURE 3. Visualization of accuracy score for each corruption, averaged over the severities, obtained when training ResNet-20 on CIFAR10 using an adversarial approach based on Wixle. The first value is the accuracy score obtained at the end of the pre-training phase.

TABLE 4. The transferability results obtained on ResNeXt-50 trained on both RGB and infrared (IR) domains the of Scene dataset. Both models are trained separately, and then a successful attack using Wixle from a domain is used also in the corresponding aligned image in the other. SR stands for Success rate.

Source → Target	Pixels per iteration	Source SR % ↑	Source → Target SR % ↑
RGB → IR	1	100	4
	1%	52.3	9.52
	5%	85.7	42.8
	10%	90.4	71.4
IR → RGB	1	100	0
	1%	71.4	4.7
	5%	90.4	23.8
	10%	90.4	52.3

the accuracy obtained on almost all the corruptions and severity combinations, with the exception of fog-5, which is the only result which is worse (having an error ratio of 1.1). Looking at the CE column, which contains corruption averaged values, we can split the corruptions into two sets by setting a threshold value of 0.7: the corruptions that have a lower CE are the ones that operate on a pixel level (e.g. Gaussian noise, pixelated), while in the second one we have corruptions that modify a pixel by taking into consideration also its neighbour pixels (e.g. blur corruptions, fog). In the end, the averaged results are better for each corruption. By looking at the second figure, we see that the accuracy results increase for each severity while using the adversarial training, and the best results are achieved after 3 training epochs.

D. TRANSFERABILITY RESULTS

Here we study the viability of transferring an attack from one domain to another. The results are shown in Table 4, which tells us that the more the attack is general (a higher number of pixels are moved), the more it is probable that it can also fool the target model. In fact, when a single pixel is attacked at each iteration, the attack achieves a higher success rate because it attacks only pixels that are highly important in the current image, but the same attack has no effect in the second domain. On the other hand, when we attack 10 percent of the pixels, the same attack is also capable of achieving a higher success rate in the target domain. This happens not because the approach is able to detect weak image spots also in the target domain, but because by moving a higher number of pixels it is more probable to move sensitive pixels also in the target domain, thus fooling the associated model.

V. CONCLUSION

In this paper, we proposed the first comprehensive study of L_0 attacks in the infrared space. We proposed two novel approaches, one which needs the internal state of the model, and another one which does not. Regarding the first one, we used it to also create more robust models, by following an adversarial attack schema. The resulting model is more robust against both L_0 attacks and natural perturbations of the input images. In the end, we also studied if a successful attack in a domain can be transferred into another one without any further tuning.

As future work, we aim to understand better the correlation between adversarial training and the robustness against natural perturbations. We also want to expand the proposed attack, in order to create a more reliable approach which is capable of attacking the models using a lower number of iterations. In the end, we want to expand our research about multi-domain robustness also to other kinds of perturbation schemes, such as L_∞.

REFERENCES

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

[2] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1625–1634.

[3] S. Qiu, Q. Liu, S. Zhou, and C. Wu, "Review of artificial intelligence adversarial attack and defense technologies," *Appl. Sci.*, vol. 9, no. 5, p. 909, Mar. 2019.

[4] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang, "Recent advances in adversarial training for adversarial robustness," 2021, *arXiv:2102.01356*.

[5] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, and X. X. Zhu, "A survey of uncertainty in deep neural networks," 2021, *arXiv:2107.03342*.

[6] H. M. D. Kabir, A. Khosravi, M. A. Hosen, and S. Nahavandi, "Neural network-based uncertainty quantification: A survey of methodologies and applications," *IEEE Access*, vol. 6, pp. 36218–36234, 2018.

[7] M. Schmitt, L. H. Hughes, C. Qiu, and X. X. Zhu, "SEN12MS—A curated dataset of georeferenced multi-spectral Sentinel-1/2 imagery for deep learning and data fusion," 2019, *arXiv:1906.07789*.

[8] C. Yuan, Z. Liu, and Y. Zhang, "Fire detection using infrared images for UAV-based forest fire surveillance," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2017, pp. 567–572.

[9] N. Chebrolu, P. Lottes, A. Schaefer, W. Winterhalter, W. Burgard, and C. Stachniss, "Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields," *Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1045–1052, Sep. 2017.

[10] J. Pomponi, S. Scardapane, and A. Uncini, "Pixle: A fast and effective black-box attack based on rearranging pixels," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2022, pp. 1–7.

[11] N. Dalvi, P. Domingos, S. Sanghai, and D. Verma, "Adversarial classification," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 99–108.

[12] D. Lowd and C. Meek, "Adversarial learning," in *Proc. 11st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2005, pp. 641–647.

[13] D. Lowd and C. Meek, "Good word attacks on statistical spam filters," in *Proc. CEAS*, 2005, pp. 1–8.

[14] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," 2012, *arXiv:1206.6389*.

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

[16] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.

[17] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[18] C. Yang, A. Kortylewski, C. Xie, Y. Cao, and A. Yuille, "Patchattack: A black-box texture-based attack with reinforcement learning," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 681–698.

[19] M. Jere, L. Rossi, B. Hitaj, G. Ciocarlie, G. Boracchi, and F. Koushanfar, "Scratch that! An evolution-based adversarial attack against neural networks," 2019, *arXiv:1912.02316*.

[20] Z. Zhou, D. Tang, X. Wang, W. Han, X. Liu, and K. Zhang, "Invisible mask: Practical attacks on face recognition with infrared," 2018, *arXiv:1803.04683*.

[21] S. Bhattacharjee and S. Marcel, "What you can't see can help you—Extended-range imaging for 3D-mask presentation attack detection," in *Proc. Int. Conf. Biometrics Special Interest Group (BIOSIG)*, Sep. 2017, pp. 1–7.

[22] F. Spasiano, G. Gennaro, and S. Scardapane, "Evaluating adversarial attacks and defences in infrared deep learning monitoring systems," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2022, pp. 1–6.

[23] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "A survey on adversarial attacks and defences," *CAAI Trans. Intell. Technol.*, vol. 6, no. 1, pp. 25–45, Mar. 2021.

[24] T. Long, Q. Gao, L. Xu, and Z. Zhou, "A survey on adversarial attacks in computer vision: Taxonomy, visualization and future directions," *Comput. Secur.*, vol. 121, Oct. 2022, Art. no. 102847.

[25] H. Liang, E. He, Y. Zhao, Z. Jia, and H. Li, "Adversarial attack and defense: A survey," *Electronics*, vol. 11, no. 8, p. 1283, Apr. 2022.

[26] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," 2019, *arXiv:1903.12261*.

[27] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[31] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1492–1500.

[32] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11976–11986.

[33] M. Brown and S. Susstrunk, "Multi-spectral SIFT for scene category recognition," in *Proc. CVPR*, Jun. 2011, pp. 177–184.

[34] H. Kim, "Torchattacks: A PyTorch repository for adversarial attacks," 2020, *arXiv:2010.01950*.

[35] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *Proc. Int. Conf. Learn. Represent.*, 2019.

JARY POMPONI (Graduate Student Member, IEEE) received the master's degree in artificial intelligence and robotics from the Sapienza University of Rome, where he is currently pursuing the Ph.D. degree, with a focus on continual learning and Bayesian methods in neural networks. Previously, he worked at Babelscape Srl, Rome, a startup company, and at Sopra Steria, before his master's degree.

DANIELE DÁNTONI received the bachelor's degree in statistics and the master's degree in data science from the Sapienza University of Rome.

NICOLOSI ALESSANDRO received the B.Sc. degree in computer engineering and the M.Sc. degree in control engineering from the University of Rome "Tor Vergata," in 2007 and 2010, respectively. He has worked in the field of industrial research and development mainly focused on unmanned aerial vehicle technologies and machine learning. Since 2015, he has been working in the field of computer vision and deep learning applied to video analysis and image processing products. Currently, he is a Principal Investigator of AI research area with Leonardo Labs (IT).

SIMONE SCARDAPANE received the B.Sc. degree in computer engineering from Roma Tre University, in 2009, and the M.Sc. degree in artificial intelligence and robotics and the Ph.D. degree from the Sapienza University of Rome, in 2011 and 2016, respectively. He worked as a software/web developer, for one year. Currently, he is an Assistant Professor with the Sapienza University of Rome. His research interests include distributed machine learning and adaptive audio processing.

Open Access funding provided by 'Università degli Studi di Roma "La Sapienza"' within the CRUI CARE Agreement