IMMUNIZING IMAGE CLASSIFIERS AGAINST LOCALIZED ADVERSARY ATTACKS

Henok Ghebrechristos and Gita Alaghband

Department of Computer Engineering, University of Colorado-Denver, Denver, Colorado

ABSTRACT

This paper addresses the vulnerability of deep learning models, particularly convolutional neural networks (CNN)s, to adversarial attacks and presents a proactive training technique designed to counter them. We introduce a novel volumization algorithm, which transforms 2D images into 3D volumetric representations. When combined with 3D convolution and deep curriculum learning optimization (CLO), it significantly improves the immunity of models against localized universal attacks by up to 40%. We evaluate our proposed approach using contemporary CNN architectures and the modified Canadian Institute for Advanced Research (CIFAR-10 and CIFAR-100) and ImageNet Large Scale Visual Recognition Challenge (ILSVRC12) datasets, showcasing accuracy improvements over previous techniques. The results indicate that the combination of the volumetric input and curriculum learning holds significant promise for mitigating adversarial attacks without necessitating adversary training.

KEYWORDS

Convolutional Neural Network, Adversary Attack, Deep Learning, Volumization, Adversary Défense, Curriculum Learning

1. INTRODUCTION

The security of any machine learning model is assessed in terms of the goals and capabilities associated with adversary attacks. Algorithmically crafted perturbations, even if minuscule, can be exploited as directives to manipulate classification outcomes [1]. Attacks can be classified as black-box or white-box [2] depending on the attacker's access to and knowledge of the model's information, which includes its architecture, parameters, training data, weights, and more. In a white-box attack, the attacker has complete access to the network's information, while a black-box attack is characterized by the absence of knowledge regarding the model's internal configuration. Occasionally, a gray-box attack can be generated by employing a generative model, enabling the creation of adversarial examples without access to the victim model. Localized adversarial attacks [3] exploit spatial invariance of CNN-based image classifiers by introducing minimal perturbations to deceive the model into producing incorrect classifications. These attacks are usually constrained to a small contiguous portion of the image and are image-agnostic (or universal) gray-box attacks.

In this paper, we introduce a new training methodology (Figure 1) designed to fortify CNNs against localized attacks. Our primary approach incorporates deep curriculum optimization [4] and a volumization algorithm. We employ an information-theoretic representation of an image along with optimization procedure that merges batch-based curriculum learning (CL), patch

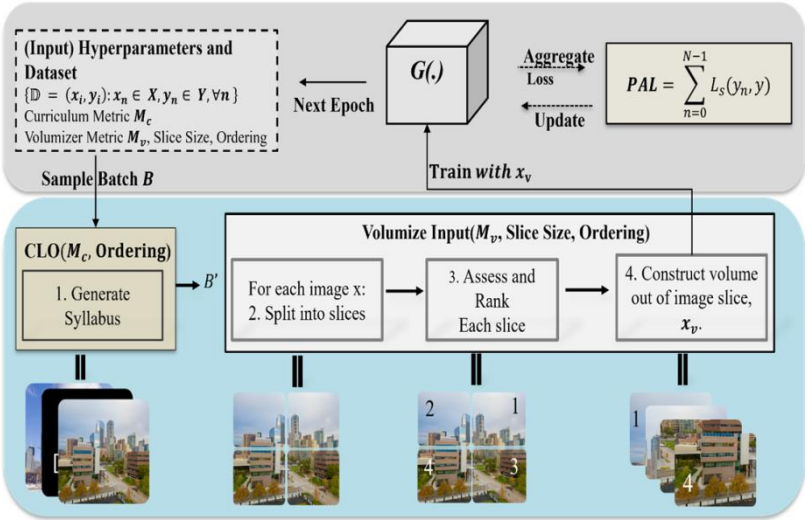


Figure 1. Overview of the proposed training process. The Curriculum Learning Optimization (CLO) component is used to generate a syllabus (input path) for the batch. It then volumizes each image, followed by feature extraction using 3D CNNs. The PAL loss function is applied to optimize parameters by calculating slice-wise errors.

aggregate loss (PAL) function, and 3D convolution to train and proactively defend against effective localized attacks; one-pixel [5] and adversary patch attacks (APA) [6].

1.1. Background on Adversary Attack

At its core, the purpose of adversary attack is to sabotage the generalization capability of a model by countering its learning objective. Given a CNN classifier $f(x;\ \theta)$, fully trained on a dataset D, its purpose is to map a source image x to a set of probabilities $f(x)$. An adversarial attack seeks to perturb this source image, producing an altered image x' such that the difference between x and x' is minimal to human perception. However, the classifier f, when processing x', produces an incorrect output that significantly deviates from the true label. This is achieved by exploiting the high-dimensional decision boundaries of the model, forcing it to misclassify x' while maintaining a semblance of the original image structure in x.

1.1.1. Attack Objective

Adversarial image attacks involve adding a perturbation $r \in \mathbb{R}^m$ to x, causing the maximized class probabilities to differ between the original and perturbed images. i.e., $argmax_i(f_i(x + r) \neq argmax_i(f_i(x)$. These types of attacks can be categorized as *targeted* or *untargeted*.

In a targeted attack, the adversarial image $x' = x + r$ is generated to induce the classifier to assign x' to a specific target class $c_t \in Y$, where $c_t \neq argmax_i(f_i(x))$. The perturbation r is selected such that $argmax_i(f_i\ (x')) = c_t$. Conversely, in an untargeted attack, the adversarial image $x' = x + r$ is crafted to cause the classifier to assign x' to any incorrect class without a particular target. In this case, the perturbation r is chosen to satisfy $argmax_i(f_i(x')) \neq argmax_i(f_i(x))$ without imposing additional constraints on the target class. *Our research focus is untargeted attacks.*

1.1.2. Défense Objective

The defense objective is to train a model that is robust to adversarial image attacks without sacrificing the accuracy of the classifier on the original dataset. Formally, the objective is to find g that minimizes the following loss:

$$\min_{g} \frac{1}{|D|} \sum_{(x,y)\in D} \max_{r \in R} L(g(x+r), y)\, 1$$

where R is the set of possible adversary perturbations added to a local region of the input, and L is a loss function used to train the model g. The objective is to minimize the maximum loss over all possible adversarial examples $x' = x + r$ generated by any allowable perturbation in R. R is constrained to be a set of *localized attacks*. Localized attacks are characterized by the property that the L2 norm of the perturbation vector r, denoted by $||r||$, is much smaller than the L2 norm of the original input image x, denoted by ||x||. Specifically, this condition can be expressed as ||r|| << ||x||. These attacks modify only a small subset of pixels that are confined to a localized region of the image.

1.1.3. Localized Universal Attacks Against Image Classifiers

Localized universal attacks are a subset of adversarial attacks that specifically target image classifiers. They exploit spatial invariances of CNNs to introduce perturbations that lead to misclassifications. These perturbations are usually confined to small, contiguous portions of the input image and can cause the model to produce incorrect output classifications, even when the introduced changes are almost imperceptible to the human eye. Two predominant types of such attacks are the N-Pixel Attack and the Adversary Patch Attack.

1.1.3.1. N-Pixel Attack

Szegedy et al. introduced adversarial attacks through minor perturbations of pixels [7] to induce CNN image misclassification. These perturbations, often undetectable to the human eye (see **Error! Reference source not found.**above), expose the inherent vulnerabilities in the robustness of Convolutional Neural Networks (CNNs).

Diving deeper into this, the N-Pixel Attack, which can be viewed as an extension or generalization of the ideas presented by Szegedy et al., specifically perturbs 'N' distinct pixels in an image to induce misclassification. The challenge and intrigue of this method arise from its seemingly benign nature; altering a minimal number of pixels in a high-resolution image intuitively appears harmless. Yet, such alterations can drastically alter CNN's prediction, underscoring the intricate and potentially fragile decision boundaries upon which these networks operate.

While the attack has profound implications for the integrity and reliability of image classifiers, it also catalyzes a renewed interest in understanding the foundational workings of CNNs. This understanding is crucial, especially in applications where trust in model predictions is paramount, such as in medical imaging or autonomous vehicles.

The N-Pixel Attack serves as a pivotal reminder that even state-of-the-art models, trained on extensive datasets and exhibiting high accuracy, can be vulnerable to carefully constructed, minimal adversarial perturbations. As researchers and practitioners continue to deploy CNNs in

varied applications, it is imperative to develop strategies that not only enhance performance but also fortify against adversarial threats.

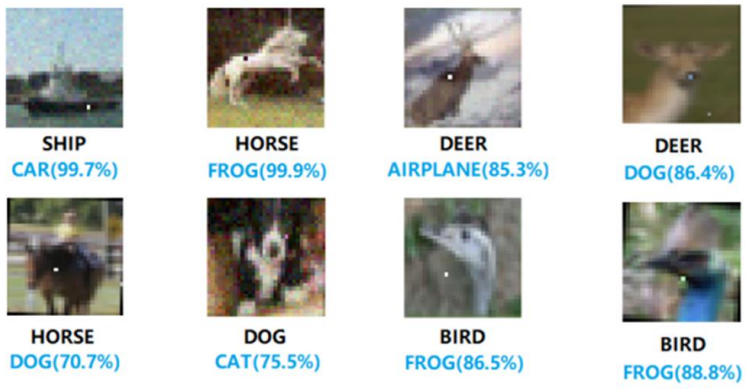


Figure 2. One Pixel Attack.

1.1.3.2. Adversary Patch Attack

Introduced by Brown et al., the Adversary Patch Attack unveils a unique and potent vulnerability in deep neural network (DNN) based classifiers [6]. Unlike other adversarial attacks that perturb an image globally, this approach is localized and focuses on modifying a confined region of the image with an adversarial patch (**Error! Reference source not found.**), which can be recognized as a seemingly harmless object or pattern added to the image. Remarkably, this addition can dramatically alter the classifier's output, demonstrating a classifier's inability to discern genuine content from deceptive information.

Central to the findings of Brown et al. was the realization that these adversarial patches were resistant to various changes, especially affine transformations such as translation, scaling, and rotation. Their methodology optimized the patches such that they were robust to these transformations. This means that the relative position, size, or orientation of the adversarial patch doesn't need to be precise for it to deceive the classifier effectively. This robustness elevates the potential real-world implications of this attack as the adversarial patch remains effective under different viewing conditions.

The adversarial patch, crafted using a white-box approach, can be applied in a "universal" manner. This universality signifies that a single patch can be effective across different images and is not tied to a specific target image. The conspicuous nature of these patches (often visually distinct) contrasts with the often-imperceptible alterations in traditional adversarial attacks, making it an intriguing anomaly in adversarial research.

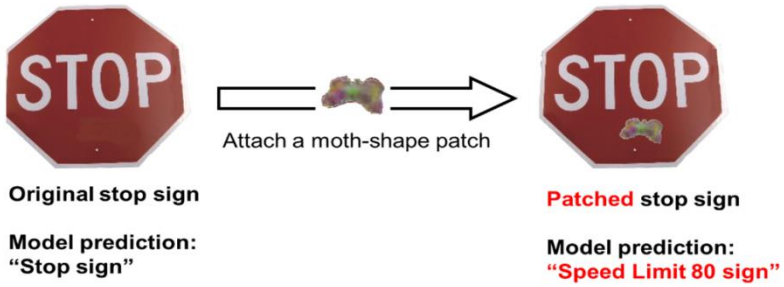


Figure 3. Adversary Patch Attack (APA).

Furthermore, Brown et al.'s findings underscore the importance of understanding not just the global, but also the localized processing dynamics of CNNs. The attack challenges the prevalent notion of CNNs' spatial hierarchies, wherein larger spatial structures (like objects) are assumed to have a dominant influence over classification compared to smaller structures or patterns. The Adversary Patch Attack highlights that this might not always be the case, as a localized, conspicuous pattern can effectively override the neural network's perception of larger structures.
In real-world scenarios, this type of attack could be employed in deceptive practices, such as placing adversarial stickers or objects in strategic locations to deceive AI systems in surveillance, autonomous driving, or even augmented reality applications. As such, the research by Brown et al. underscores the importance of defensive mechanisms that consider both global and local image features.

Both Brown et al., and later Gittings et al. [8] backpropagate through the target model to generate ‘stickers’ that can be placed anywhere within the image to create a successful attack. This optimization process can take several minutes for one single patch. Karmon et al. showed in LaVAN that the patches can be much smaller if robustness to affine transformation is not required [3] but require pixel-perfect positioning of the patch which is impractical for real APAs.

1.1.3.3. Local Distortion Attack

Local Distortion Attacks[9] focus on altering small, strategically chosen areas of an input image to deceive a classifier. Unlike adversarial attacks that introduce perturbations across the entire image, these methods pinpoint areas that disproportionately influence the classifier's decision. Identifying these critical regions often involves saliency maps and gradient-based analysis[9]. Once the key regions are determined, the attacker leverages optimization techniques to introduce subtle distortions, designed to mislead the classifier. These distortions include pixel intensity changes, targeted geometric transformations (like localized rotation or scaling), or the injection of meticulously crafted noise patterns. The distortion process frequently uses an iterative approach; the attacker modifies the image and observes the impact on the classifier's output, refining the distortions until misclassification occurs.

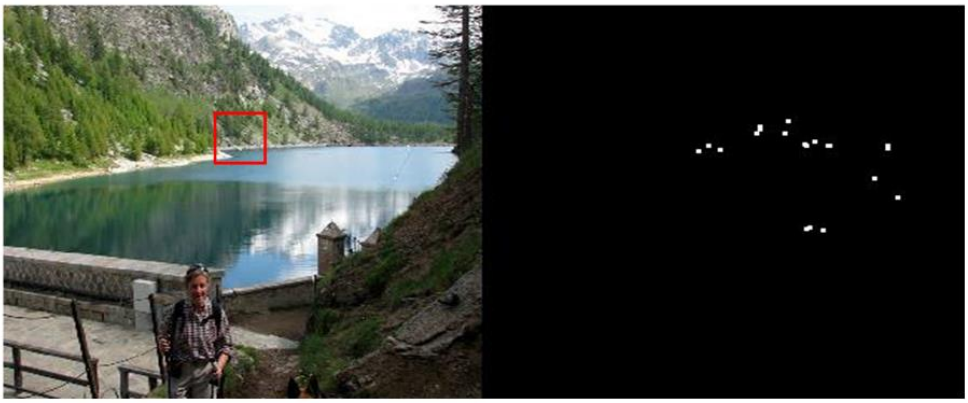


Figure 4.GreedyFool: Local Distortion Attack.

Once these influential regions are identified, the attacker employs optimization techniques to introduce subtle distortions designed to mislead the classifier (Figure X). These local distortions can manifest as pixel intensity changes, targeted geometric transformations (e.g., localized rotation, scaling, or shearing), or carefully crafted noise patterns. The optimization process is frequently iterative, with the attacker applying distortions, evaluating the impact on the classifier's output, and refining the manipulations until misclassification is achieved.

A fundamental goal of local distortion attacks is to induce misclassification while minimizing the overall perceptibility of the introduced alterations. This introduces a trade-off between attack efficacy and the visibility of the changes. Attackers may be constrained by specific limits on the allowable types of distortions or the total number of pixels they can manipulate. In certain scenarios, the attacker can further refine the attack by exploiting knowledge of the target model's architecture or its underlying training data.

Local distortion attacks pose unique challenges compared to other adversarial attack types. The localized nature of the perturbations necessitates a nuanced understanding of the model's internal decision-making processes. Unlike adversarial patches, they are often designed to be subtle and blend with the original image content. This focus on minimizing human detectability raises concerns about the potential for these attacks to bypass security measures or mislead human inspection in real-world scenarios.

1.1.3.4. TnT Attack: Universal Naturalistic Adversarial Patches Against Deep Neural Network Systems

The TnT Attack (Transformers 'n' Trojans) presents a sophisticated adversarial patch methodology designed to disrupt DNN-based image classifiers [10]. Unlike traditional adversarial patches, TnT Attack patches are crafted to mimic natural textures and patterns, making them less conspicuous. The attack leverages a Transformer architecture, often a Vision Transformer (ViT), and a separate external image dataset to generate its patches. The Transformer's attention mechanisms play a crucial role in identifying patterns and textures that have a potent adversarial effect when selectively introduced into images. Furthermore, the TnT Attack emphasizes universality, meaning a single patch can be placed in various locations across a wide range of images and still cause the target classifier to misclassify. This flexibility increases the potential threat in real-world scenarios with less constrained patch placement.

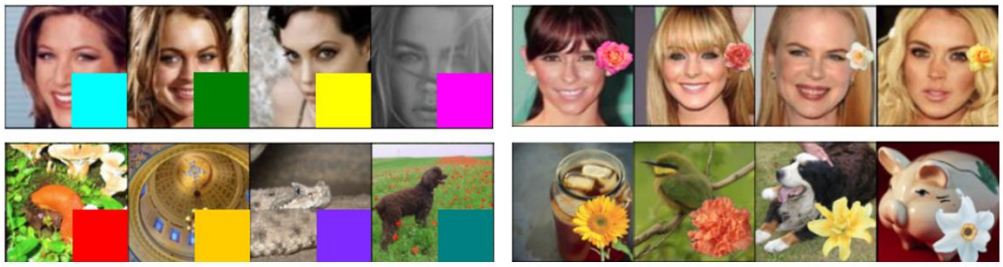


Figure 5.Random color and flower patches generated by TnT attack and applied to PubFig dataset.

At the heart of the TnT Attack lies a Transformer architecture, Vision Transformer (ViT). Transformers have become prevalent in image processing tasks due to their ability to model long-range dependencies. The Transformer is fed images from a separate external dataset, unrelated to the dataset used to train the target classifier. Critically, the Transformer's attention mechanisms analyze these external images to identify patterns and textures that have a potent adversarial effect when selectively introduced into images the classifier was designed to process.

The TnT Attack further emphasizes the concept of universality. Unlike adversarial attacks tailored to a specific target image, the goal is to generate a single patch that can be placed in various locations across a wide range of images and still cause the target classifier to misclassify. This lack of specificity increases the potential threat in real-world scenarios where an attacker might have limited control over the exact context and placement of the adversarial patch.

2. RELATED WORK

Goodfellow et al. proposed a method to enhance the robustness of neural networks against such attacks. This approach, known as adversarial training [1], involves using adversarial examples as inputs during training to teach the network how to accurately classify them, thus improving its ability to defend against attacks. Adversary training methods require access to the target model to backpropagate gradients to update pixels, inducing high frequency noise that is fragile to resampling. In 2016 Papernot et al. proposed a defensive mechanism called defensive distillation [11], in which a smaller neural network learns and predicts the class probabilities generated by the original neural network's output. Despite growing number of defense approaches, several attacks remain effective – particularly attacks generated using generative architectures [12], [13].

2.1. N-Pixel Attack Defences

Brown et al. demonstrated that adversarial patches could be used to fool classifiers; they restricted the perturbation to a small region of the image and explicitly optimized for robustness to affine transformations [6]. Su et al. in 2017, which generates fooling images (adversarial examples) by perturbing only one pixel or few pixels, has proven to be difficult to defend [5]**.** To date, the most successful defense against this attack is a method presented by Chen et al. [14]. The authors propose Patch Selection Denoiser (PSD) that removes few of the potentially attacked pixels in the whole image. At the cost of image degradation, the authors achieve a successful defense rate of 98.6% against one-pixel attacks. Similarly,

Liu et al. [15] proposed a three-step image reconstruction algorithm to remove attacked pixels. The authors report protection rate (defense success rate) of up to 92% under for N-pixel attack for N chosen from the range (1, 15). Shah et al. [16] proposed the usage of an Adversarial Detection Network (ANNet) for detection of N-pixel attacks where N is 1, 3 or 5 and report up to 97.7 adversarial detection accuracy on MNIST Fashion dataset. Husnoo and Anwar [17] proposed an image recovery algorithm based on Accelerated Proximal Gradient (APG) [18] approach to detect and recovered the attacked pixels.

2.2. Adversary Patch Defences

Defenses for patch attacks are typically viewed as a detection problem [19], [20]. Once the patch's location is detected, the suspected region would be either masked or in-painted to mitigate the adversarial influence on the image. Hayes [21] first proposed DW (Digital Watermarking), a defense against adversarial patches for nonblind and blind image inpainting, inspired by the procedure of digital watermarking removal. A saliency map of the image was constructed to help remove small holes and mask the adversarial image, blocking adversarial perturbations. This was an empirical defense with no guarantee against adaptive adversaries.

Naseer et al. [22] proposed LGS (Local Gradient Smoothing) to suppress highly activated and perturbed regions in the image without affecting salient objects. Specifically, the irregular gradients were regularized in the image before being passed to a deep neural network (DNN) model for inference. LGS could achieve robustness with a minimal drop in clean accuracy because it was based on local region processing in contrast to the global processing on the whole image as done by its counterparts.

Chou et al. [23] proposed SentiNet for localized universal attacks to use the particular behavior of adversarial misclassification to detect an attack, which was the first architecture that did not require prior knowledge of trained models or adversarial patches. Salient regions were used to

help observe the model's behavior. SentiNet was demonstrated to be empirically robust and effective even in real-world scenarios. However, it evaluated adversarial regions by subtracting the suspicious region, which might at times cause false adversarial region proposals. Moreover, the suspicious adversarial region was placed at random locations in the preserving image, which possibly occluded the main objects in the scene resulting in incorrect predictions.

Chen et al. [24] proposed Jujutsu to detect and mitigate robust and universal adversarial patch attacks by leveraging the attacks' localized nature via image inpainting. A modified saliency map [25] was used to detect the presence of highly active perturbed regions, which helped to place suspicious extracted regions in the least salient regions of the preserved image and avoid occlusion with main objects in the image. Jujutsu showed a better performance than other empirical defenses in terms of both robust accuracy and low false-positive rate (FPR), across datasets, patches of various shapes, and attacks that targeted different classes.

2.3. TnT Attack Defences

Developing robust defences against TnT attacks is a critical area of active research. Current defence strategies can be broadly categorized into several key approaches, texture analysis, and anomalous pattern detection.

Texture Analysis focuses on detecting subtle textural anomalies introduced by TnT patches. Even when designed to mimic natural patterns, adversarial textures may exhibit subtle statistical deviations or atypical characteristics compared to textures found in genuine images[2]. Researchers employ a variety of techniques including statistical measures, specialized filters, and image processing methods sensitive to textural irregularities to expose these anomalies. However, a potential limitation lies in highly sophisticated TnT attack generation, where adversaries could craft patches exceptionally difficult to distinguish based on texture alone. Carefully calibrated thresholds are also important in texture-based defences to avoid excessive false positives on legitimate images, which often possess diverse and sometimes unusual natural textures.

Anomalous Pattern Detection detects anomalous patterns to circumnavigates these attaches. Since TnT patches originate from unrelated image datasets, they introduce the risk of incorporating visual patterns that are contextually out of place within the images they are applied to [26]. Defence strategies capitalize on this by developing algorithms designed to detect these contextually unusual or anomalous patterns. A key challenge in this approach is defining what constitutes "normal" patterns for a given image class, as real-world images can exhibit significant visual diversity. An adaptive adversary could also potentially modify TnT attacks to introduce patterns that are less jarringly out of context, decreasing the effectiveness of such defences.

2.4. Distortion Attack Defences

Defending against local distortion attacks is inherently challenging due to the subtle and highly targeted nature of the introduced perturbations. Strategies generally focus on detecting anomalous regions within the image and subsequently mitigating their impact on the classifier's decision.

One defence approach relies on gradient-based detection, exemplified by the LGS method [22]. It operates under the assumption that local distortions introduce irregular gradient patterns that are distinct from the natural variations found in benign images. By identifying and suppressing these irregular patterns within localized regions, LGS can potentially neutralize the adversarial effect. Another detection strategy leverages saliency maps to pinpoint areas of the image that exhibit unusual activation patterns or deviate from the expected distribution of saliency in clean images [27], [28]. These flagged regions might indicate the presence of a local distortion. Defence

methods like SentiNet [23] take a behavioural analysis approach, monitoring the classifier's actions when presented with potentially manipulated images. Pronounced changes in predictions, particularly when the modified regions are not semantically significant, can be a powerful signal of the presence of a local distortion attack.

Once a potentially distorted region is identified, mitigation techniques are employed. Methods like Jujutsu [26] use inpainting techniques to replace the suspicious region with image content that seamlessly blends with the surrounding areas. This aims to neutralize the distortion while maintaining the overall coherence and natural appearance of the image. In some cases, a simpler approach of masking out the suspect region may be sufficient to prevent the classifier from processing the manipulated content. However, the effectiveness of masking is contingent on the distorted area not occluding critical components of the image for classification.

Many defenses are designed with specific assumptions about how distortions manifest. An adaptive attacker might modify their techniques to circumvent existing detection mechanisms. Furthermore, defenses must strike a delicate balance between detecting subtle distortions and maintaining accuracy on clean, unmodified images. Overzealous detection strategies can lead to false positives where benign image regions are incorrectly flagged as adversarial. Evaluating the robustness of local distortion attack defenses across varying distortion types, target models, and datasets is an essential component of the defense development process. Instead of relying on patch or pixel detection and removal techniques, our method uses deep curriculum learning optimization (Deep-CLO) [29] and 3D convolutional neural networks [30] to proactively defend against these attacks. An overview of our proposed training methodology is shown in Figure 1.

3. METHOD

Our aim is to develop a defended classifier, $\dot{G}$, that inherently defends against localized attacks without relying on adversarial training or prior knowledge of the attacks. To realize this objective, we propose a *proactive defense approach* characterized by following key steps:

- *Volumization:* For each image in the batch, we convert it to a 3D volume to capture the spatial information of the input.
- *3D Convolution:* We modify the contemporary CNN model architectures to do 3D convolution, which enables the model to extract features from the 3D input volumes. Details of the modifications are below.
- *Deep curriculum learning optimization*[4]*:* For each batch taken from the training dataset D, we generate a syllabus that determines the input order of the samples in the batch.

The combination of 3D convolution, Deep-CLO, and the volumization algorithm empowers $\dot{G}$ to maintain high performance on both clean and adversarial inputs. These techniques ensure that the model remains resilient to perturbations and that it maintains accurate classification and verification of both the original images x and adversarial images $x' = x + r$. Refer to analysis section for detailed justification.

When used as a preprocessing step, the algorithm allows $\dot{G}$ to extract spatial relationships from the volumized data, resulting in enhanced robustness against localized attacks. The employment of Deep Curriculum Optimization as the training procedure further bolsters the $\dot{G}$'s resilience to adversarial attacks. The resulting classifier not only defends against the targeted attacks but also retains high performance on non-adversarial images, achieving classification accuracy comparable to state-of-the-art models. Consequently, our approach demonstrates a unique

balance between robustness and accuracy without relying on adversarial training, making it a significant contribution to the field of image classification under adversarial conditions.

3.1. Volumization Algorithm

The volumization algorithm, is a pixel-preserving and reversible transformation operator denoted as a function:

$$V : (x, H, W, C) \rightarrow (x_i', p_h, p_w, C, Z). \qquad 2$$

Given an input image $x \in D$ of shape (H, W, C), the algorithm uses a configurable hyperparameter, patch size $S(p_h, p_w)$ - where p_h and p_w are the height and width of each slice - to split x into Z non-overlapping slices $x_i'$of size P satisfying the pixelconservation condition:

$$x = \bigcup_{i=0}^{Z-1} x_i' \qquad 3$$

This states that the original image x is equivalent to the union of all its extracted slices (see the image at the bottomof Figure 1). Here, ∪ denotes the union operation. The algorithm extracts a list of slices and proceeds to rank each slice according to a prespecified metric m – a configurable hyperparameter. The chosen metric can be of type distance or standalone (Table 1). If m is a distance-based metric, a reference slice P_{ref} is selected, which can be user-definedor automatically determined by choosing the most salient slice. On the other hand, m is considered standalone if it measures some characteristics of a given slice. All slices are then ordered based on their individual metric scores or their distances from the reference slice. The ordering ord is user define configurable hyperparameter that can be either descending or ascending. Finally, the ranked slices are stacked along the depth axis to create a 3D volume $x_v = V(x)$ ofshape (p_h, p_w, C, N), where Z (depth of the volume) isthe total number of slices. Refer to Ghebrechristos et. al. [4] for detailon the ranking and ordering process, which is identical for both the volumizer and CL when generating a syllabus for a batch.

By breaking the image into smaller patches, we localize the region of analysis, making the training process more sensitive to adversarial attacks that affect only a small portion of the image. This localization often aligns with the attack region of localized adversarial attacks, enhancing the models' ability to detect and respond to them.

3.2. Model Architecture

Given a conventional CNN classifier architecture f designed to learn from 2D images, we perform the following modifications to construct Ġ - a 3D counterpart f.

3.2.1. Input Layer

f's input layer, denoted as I_{2D}, is designed to accept a 2D input data x with dimensions $H \times W \times$ C such that $I_{2D}: x \in \mathbb{R}^{H \times W \times C}$. In order to extract features from the volumized images, the input layer of model Ġ is adjusted to be 3-dimensional, I_{3D}, where the input to this layer x_v is a 4D tensor with dimensions $H' \times W' \times$ C × Z, such that $I_{3D}: x_v \in \mathbb{R}^{H' \times W' \times C \times Z}$. In shorthand notation, this reversible transformation can be represented as:

$$f(I_{2D}: x \in \mathbb{R}^{H \times W \times C}) \leftrightarrow \dot{G}(I_{3D}: x \in \mathbb{R}^{H' \times W' \times C \times Z}), \qquad 4$$

where H' and W' are prespecified width and height of the individual patches within the volume and Z signifies the total number of patches. This enables the classifier to learn features from the volumized data x_v.

3.2.2. Convolution Layer

For CNN, convolution represents the interaction between an input (image or feature map) and a kernel (filter). The kernel is a small matrix that slides over the input data, performing an element-wise multiplication and summing the results to generate a new feature map. In a 2D convolution, the input data and the kernel are both two-dimensional.

$$(K * x)(i,j) = \sum_m \sum_n K(m,n)\, x(i-m, j-n)\, 5$$

Here, K represents the 2D kernel, x represents the 2D input and (i,j) are the coordinates in the output feature map. The summation is performed over all spatial dimensions (m,n) of the 2D kernel.

For a given classifier, 3D counterpart of the above operation is:

$$(K_{3D} * x_v)(i,j,k) = \sum_m \sum_n \sum_p K_{3D}(m,n,p) x_v(i-m, j-n, k-p)\, 6$$

where K_{3D}represents the 3D kernel, x_v is the 3D input data and (i,j,k) are the coordinates in the output feature map. The summation is performed over all spatial dimensions (m,n,p) of the 3D kernel. This modification enables the classifier to learn features from the volumized data by processing spatial information across height, width, and depth dimensions simultaneously.

Note that the optimal kernel size depends on the size of the individual slices within the volume and the desired level of spatial information capture. For example, if f consists of 1×1, 3×3, and 5×5 2D convolution layers, we adjust these layers to be $1 \times 1 \times Z$, $3 \times 3 \times Z$, and $5 \times 5 \times Z$ 3D convolution layers, respectively. To ensure compatibility, we enforce the constraint that the kernel size is much smaller than the size of the individual slices and that the kernel operates on each slice in the volume. That is, $H' \ll H$, and $W' \ll W$. The stride and padding values are also adjusted accordingly.

3.2.3. Pooling Layer

All pooling layers of f are modified to handle the 3D volume representation of the input data. In f, the 2D pooling layers denoted as P_{2D}:

$$P_{2D}: \mathbb{R}^{H_{in} \times W_{in} \times C_{in}} \rightarrow \mathbb{R}^{H_{out} \times W_{out} \times C_{out}} \quad 7$$

Where H_{in}, W_{in} and C_{in} represent the height, width, and number of channels of the input feature maps, while H_{out}, W_{out} and C_{out} denote the height, width, and number of channels of the output feature maps, respectively.

To effectively process volumized inputs, we replace the 2D pooling layers with 3D counterparts:

$$P_{3D}: \mathbb{R}^{H_{in} \times W_{in} \times C_{in}} \rightarrow \mathbb{R}^{p_{h_{out}} \times p_{w_{out}} \times C_{out} \times N_{out}} \quad 8$$

where $p_{h_{in}}, p_{w_{in}}, C_{in}$ *and* N_{in} represent the height, width, number of channels, and number of patches of the input volume, while $p_{h_{out}}, p_{w_{out}}, C_{out}$ *and* N_{out} denote the height, width, number of channels, and number of patches of the output 3D volume, respectively.

3.2.4. Normalization and Activation Layers

These layers usually play an essential role in maintaining a stable and efficient training process and introducing non-linearity to the model. For normalization layers, we transition from 2D normalization methods of f, such as Batch Normalization (BN) and Instance Normalization (IN), to their 3D counterparts in g. Given 3D input tensor, $x_v \in \mathbb{R}^{\mathbf{H}\times\mathbf{W}\times\mathbf{C}\times\mathbf{N}}$, the 3D normalization layer computes the mean μ and standard deviation σ across the specified dimensions (usually height, width, and depth) and normalizes x_v as follows:

$$x_{v-normalized} = \frac{x_v - \mu}{\sigma}, \qquad 9$$

where μ and σ are broadcasted to match the dimensions of x_v.

For activation layers, the transition from 2D to 3D input data is more straightforward. Common activation functions, such as $ReLU$ and $sigmoid$, can be directly applied to the 3D input data, with minor tweaking, as these functions perform element-wise operations on the input tensor. The output tensor $y_v \in \mathbb{R}^{\boldsymbol{H}\times\boldsymbol{W}\times\boldsymbol{C}\times\boldsymbol{N}}$, by applying the activation function Af elementwise to the input tensor x_v:

$$y_v = Af(x_v[i,j,c,n]), \forall\, i \in [0,H), j \in [0,W), c \in [0,C), n \in [0,N). \quad 10$$

By ensuring that normalization and activation layers are compatible with the 3D input data, we maintain the stability and efficiency of the training process, while enabling the model to effectively learn non-linear features from the volumized input data.

3.2.5. Fully Connected and Output Layers

To perform patch-wise error calculation and enhance model's robustness, we modify $f's$ fully connected layer function *F:* $\mathbb{R}^{(M,N)} \rightarrow \mathbb{R}^{(L,Z)}$ where M represents the number of input features, L denotes the number of output features, and Z is the total number of patches. The fully connected layer function F' now maps each patch's input features to its respective output features, allowing for patch-wise error calculations during backpropagation.

The output layer function $O: \mathbb{R}^{(L,N)} \rightarrow \mathbb{R}^{(k,N)}$, where k is the number of classes. To ensure compatibility with the 3D input data, the output of the preceding layers must be reshaped or flattened before connecting to the fully connected layers. This modification allows g to map each patch's output features to its respective class probabilities, further enabling *patch-wise error* calculations during the training.

3.2.5.1. Patch Aggregate Loss (PAL) Function

PAL is designed to enable backpropagation on individual patches. During training, the loss for each patch is calculated separately. The patch-wise losses are then aggregated to obtain the overall loss for the image. Given the modified output *O:* $\mathbb{R}^{(L,N)} \rightarrow \mathbb{R}^{(k,N)}$, we define PAL as follows:

Patch-wise Error Calculation: Computes the loss for each patch separately using a suitable loss function $L_p: \mathbb{R}^{(N',k)} \rightarrow \mathbb{R}^{(N')}$. For a given patch $n \in \{0, 1, \ldots, Z-1\}$, the patch-wise loss is calculated as $L_p(y_n, y)$, where y_n represents the predicted class probabilities for patch n, and y denotes the true class labels of the original input.

Loss Aggregation: Aggregate the patch-wise losses to obtain the overall loss for the image. This function, termed Patch Aggregate Loss (PAL) function, computes the overall loss of an image by summing up the individual patch-wise losses:

$$PAL = \sum_{n=0}^{N-1} L_p(y_n, y) \qquad 11$$

Using the sum of slice-wise losses directly emphasizesthe importance of minimizing the error for each individualslice, driving the model to learn more robust features from each slice. This increased emphasis on localized featuresresults in a more robust model that is better equipped tocounteract attacks.

3.3. Training Methodology

We incorporate deep curriculum learning optimization (CLO) as described in Ghebrechristos et. al. [4] at a batch level to enhance the training process. Given a batch $B \subseteq D$, we define a syllabus S as a function $S: B \rightarrow B'$, where B' is a reordered version of the original batch B. S describes an input order of the samples in B' such that thelearning process progresses from simpler to more complex samples as quantified by a concrete metric m taken from Table 1.

Table 1. List of measures used in this study. Given samples $x, x_1, x_2 \in B$ where b_x is normalized histogram of pixel intensities and i is an index of a pixel value in the image's vector. σ is standard deviation and μ is mean or average pixel intensities.

Metric	Implementation	Category
Entropy	$H(x) = \sum_{i \in \chi, x \in D} b_x(i) \log \frac{N}{b_x(i)}$	standalone
Joint Entropy (JE)	$JE(x_1, x_2) = \sum_i b_x(i) \log b_x(i)$	distance
Mutual Information (MI or I)	$MI(x_1, x_2) = H(x_1) + H(x_2) - JE(x_1, x_2)$	distance
KL-Divergence (KL)	$D_{kL}(x_1 \mid\mid x_2) = \sum_i x_{1_i} \log \frac{x_{1_i}}{x_{2_i}}$	distance
Structural Similarity index (SSIM)	$SSIM(x_1, x_2) = \frac{(2\mu_{x_1}\mu_{x_2} + C_1)(2\sigma_{x_1x_2} + C_2)}{(\mu_{x_1}^2 + \mu_{x_2}^2 + C_1)(\sigma_{x_1}^2 + \sigma_{x_2}^2 + C_2)}$	distance
Max Norm (MN)	$x_\infty = \max(x_1, x_2)$	distance
Peak signal to noise ratio (PSNR)	$PSNR = 20 \log_{10}(\frac{MAX}{\sqrt{MSE}})$	distance
Mewan Squared Error	$MSE(x_1, x_2) = \frac{1}{N^2} \sum_i^N \sum_j^N \left(x_{1_{ij}} - x_{2_{ij}}\right)^2$	na

Ordering of samples for the batch is done in the sameway the volumizer algorithm orders slices to create a volume. Given $B = \{x_1, x_2, \ldots, x_n\}$, a batch of n samples (or a set of patches belonging to x), let $SM(x_i)$ be the standalone metric value of xi and $DM(x_i, P_{ref})$ be the distance metric value of the sample or slice x_i with respect to a reference image (or patch) P_{ref}, respectively. We define order relations $R_{SM} \subseteq B$ and $R_{DM} \subseteq B$, such that:

$$(x_i, x_j) = \begin{cases} R_{SM} \; if \; SM(x_i) \leq SM(x_j) \\ R_{DM} if \; DM(x_i, P_{ref}) \leq DM(x_j, P_{ref}) \end{cases} \quad 12$$

Thus, the syllabus (or volumizer) algorithm transforms B (set of slices) into an ordered one B':

$$S_{SM(B)} = \{x'_1, \ldots., x'_n, where\ (x'_i, x'_j) \in R_{SM}\} \quad 13$$

$$S_{DM(B)} = \{x'_1, \ldots., x'_n, where\ (x'_i, x'_j) \in R_{DM}\} \quad 14$$

The learning process progresses from simpler to more complex samples based on a specific metric, which enhances model performance and speeds up convergence. This method also strengthens models against localized attacks by ordering patches based on their features. Adversarial perturbations in a single patch have less impact on the model's image understanding due to this arrangement.

4. EXPERIMENTS & RESULTS

Our approach is evaluated on EfficientNet-B0, InceptionV3, ResNet50, and VGG19 architectures modified for 3D input compatibility. These modifications result in a *significant but* tolerable increase in the number of parameters: approximately 20M for VGG, 49M for ResNet, 44M for Inception, and 10M for EfficientNet. The models, implemented via open-source TensorFlow [31] library, are tested on CIFAR10, CIFAR100, and ILSVRC12 datasets under different attack settings. CIFAR10 facilitates comprehensive study, while CIFAR100 and ILSVRC12 test the approach's generalizability.

We measure the *classification accuracy* – of the models on both clean images (acc_{clean}) and adversarial images acc_{attack}. We calculate *robustness score* **(**δ**)** as the difference between model's classification accuracy on clean images and its classification accuracy on adversarial images, $\delta = (acc_{clean} - acc_{attack})$. A smaller δ demonstrates greater proactive robustness against adversarial attacks. We also measure *defense success rate* β – the percentage of successfully defended adversarial attacks. A higher defense success rate indicates a better proactive defense against adversarial attacks.

We contrast performance with a baseline defense approach using similar datasets. We used adversary stickers (**Error! Reference source not found.**) synthesized by A-ADS method of Brown et al. [6] and flower patches extend the adversary benchmark library, FoolBox's [32] to implement N-pixel attacks.

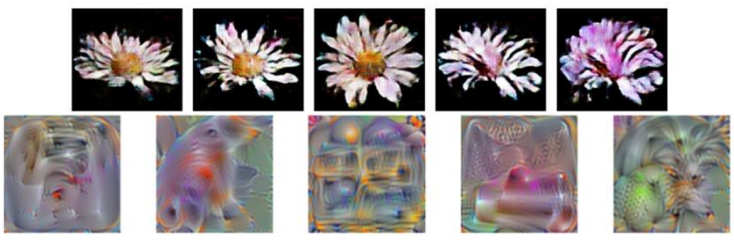


Figure 6 Patch Attack Stickers (bottom) and TnT Flower Patches (top) Used in this Study.
Table 2. Number of Parameters for different models; original (f) and with modifications (f_{3D}).

Model	Original (f)	Modified (f_{3D} or G)
VGG16	143,357,544	183,021,512
ResNet50	25,636,712	74,203,245
InceptionV3	23,851,784	68,104,050
EfficientNetB0	5,330,564	15,900,000

4.1. Défense Success Rate

Error! Reference source not found.7 shows the defense success rate of model trained with our method surpasses 80% after 300 epochs, indicating effective defense against 1-pixel attacks at each validation run. After 50 epochs, the classifier rapidly learns to resist the attacked pixel, increasing $\dot{G}'s$ success rates while those of f stagnate below 72%. This confirms that the approach delivers models that match the undefended model's performance on clean datasets while resisting localized attacks.

4.2. Défense Effectiveness

We evaluate the performance of our defense in reducing the effectiveness of N-Pixel and patch attacks. We use 1, 2, up to 16-pixel coverage for N-pixel. We use adversarial patches – Toaster, School-Bus, Lipstick and Pineapple - synthesized by attack methods A-ADS, covering up to 25% of the entire image. Our approach is compared with existing defense strategies in terms of clean accuracy and defense success rate β. We mount such attacks against our defense (I, KL, H, MN, and PSNR syllabi), and an undefended model as a control. The patch size(p_h, p_w) of the volumization algorithm for all syllabi is set to 16×16 pixels. We take reported results of all baseline defenses for comparison.

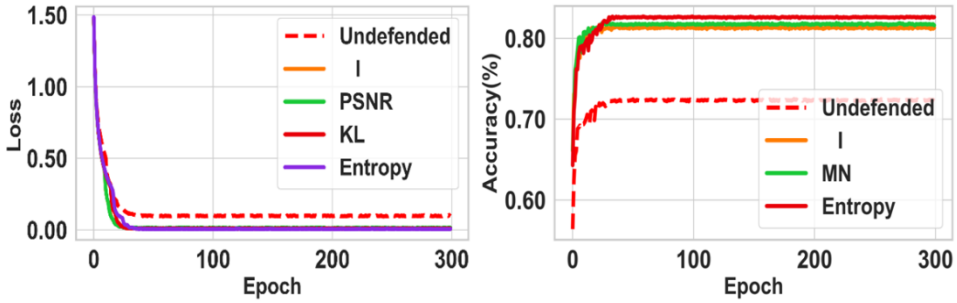


Figure 7. CIFAR10 Training losses and Success Rates of Defense on EfficientNet. (Left) Shows the Losses of Defended Model G Using Different Syllabus Configurations and Undefended Model F. (Right) Shows the Training Success Rates of Both Models Under 1-Pixel Attack.

Table 3 presents a comparison of generalization performance of EfficientNet on CIFAR10, CIFAR100 and ImageNet datasets, with and without our defense mechanisms, for both clean and attacked test sets. Though the undefended model exhibits good performance clean data, its performance significantly deteriorates under adversarial attacks. In contrast, models defended using our approach show resilient performance under adversarial scenarios, with minimal trade-offs in clean data accuracy.

Notably, the model defended using measure MI outperforms other models under N-Pixel attack across all datasets. For instance, the model defended using mutual information (I) achieves attack accuracies of 91.3 (N-Pixel) and 61.6 (APA) on CIFAR10, remarkably higher than the undefended model's 43.2 and 44.3, respectively. Similarly, the KL-defended model yields considerably better attack accuracy on CIFAR100 (73 and 43.2 for N-Pixel and APA respectively) compared to the undefended version (32.5 and 22.1).

Error! Reference source not found.7 illustrate the overall robustness (δ) of EfficientNet and Inception against N-pixel and patch attacks, respectively. The plots highlight the dependence of model robustness on attack size for both defended and undefended models, with the undefended model being 40% less accurate at worst. Our defense is effective for both architectures at all attack magnitudes. However, like the undefended model, the performance of our method degrades as the size of the attack increases, indicating a shared vulnerability to larger-scale attacks.

Table 3. Generalization accuracy of EfficientNet on CIFAR10, CIFAR100, and ILSVRC12 datasets with and without our defence mechanisms. The performance is compared under three scenarios: Clean Test Sets, Test Sets Under One-Pixel Attack (N-Pixel Where N=1),Test Sets Under Patch Attack with a Toaster Sticker (APA).

Defence	Acc_{clean}			$Acc_{attack}(N-pixel)$			$Acc_{attack}(APA)$		
	CIFAR10	**CIFAR100**	**ISLVRC12**	**CIFAR10**	**CIFAR100**	**ISLVRC12**	**CIFAR10**	**CIFAR100**	**ISLVRC12**
Entropy(H)	94	90.5	76.8	**85**	**74**	**56.8**	**79**	**55.2**	**58.2**
MI	96.3	**98**	**79**	**91.3**	**70**	**69**	**61.6**	**53**	**61.2**
KL	93	93.2	75	**63**	**73**	**62.1**	**65**	**43.2**	**36.8**
PSNR	89	90.3	76	**83.6**	**51**	**56**	**52**	**53**	**48.3**
Norm(MN)	92	86	75.4	**74**	**51**	**49.5**	42	**38**	**32**
Undefended	**99**	96.8	78.3	43.2	32.5	12.4	44.3	22.1	10.8

As presented in Tables 3 & 4, our proposed defense demonstrates a significant performance against N-Pixel attack compared to the undefended models. The undefended models exhibit sharp decline when under attack (Acc_{attack}column). Our proposed method (I) not only achieves high Acc_{clean} of 96.3% for EfficientNet and 99% for VGG and Inception but also shows a minor degradation when under attack; by 5%, and 0.4% for Efficient and VGG respectively.

Table 4. Generalization accuracy of EfficientNet on CIFAR10, CIFAR100, and ILSVRC12 datasets with and without our defence mechanisms. The performance is compared under three scenarios: Clean Test Sets, Test Sets Under TnT Flower Attack and Distortion Attack.

Defence	Acc_{clean}			$Acc_{attack}(TnT)$			$Acc_{attack}(Distortion)$		
	CIFAR10	**CIFAR100**	**ISLVRC12**	**CIFAR10**	**CIFAR100**	**ISLVRC12**	**CIFAR10**	**CIFAR100**	**ISLVRC12**
Entropy(H)	94	90.5	76.8						
MI	96.3	**98**	**79**						
KL	93	93.2	75						
PSNR	89	90.3	76						
Norm(MN)	92	86	75.4						
Undefended	**99**	96.8	78.3						

Comparing mutual information (MI) with the PSD method, our approach has a slightly lower $\boldsymbol{Acc_{clean}}$for VGG, with a difference of 0.53%, but delivers a better $\boldsymbol{Acc_{attack}}$ for the same model, with an improvement of 1.2%. When comparing I to Liu et al.'s method, our methoddemonstrates a substantial improvement in $\boldsymbol{Acc_{clean}}$ for VGG, with a difference of 9.2%, and a higher $\boldsymbol{Acc_{attack}}$ as well, with an improvement of 4.6%. PSD and Liu et al. do not provide results for EfficientNet.

Table 5 presents defense success rates (β) for various defense methods and ours against APA on three datasets. For CIFAR10, our methods achieved 95.6% and 96.12% success rates, while Jujutsu and LGS obtained only 86.5% and 93.2%, respectively. Similarly, for CIFAR100, our methods reached success rates of 95.43% and 94.3%, outperforming Jujutsu's 55.7% and LGS's 73.7%. In the ImageNet dataset, ours (MI) achieved the highest defense success rate of 89.1%, while PSNR obtained 83.2%, both surpassing Vax-a-Net's 86.8% and DW's 65.2% and66.2% for VGG and Inception models, respectively. Not all methods have reported results for every dataset, limiting a comprehensive comparison of their effectiveness.

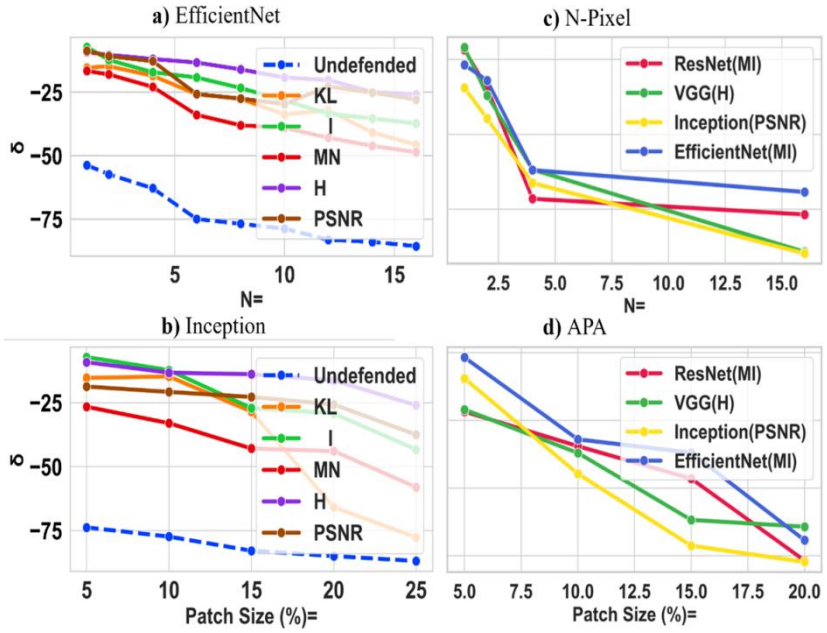


Figure8. (a) EfficientNet Robustness as a Function of N - Number of Pixels Attacked, (b) Inception Robustness Against APA As a Function of Patch Size, (c) Defense Success Rate of Various Models as a Function of N-Pixel Attack Magnitude, (d) Defense Success Rate B of Various Models as a Function of APA Attack Magnitude.

Table 5. Accuracy of models over the set of test images without attacks $\mathbf{Acc_{clean}}$ and with attack $\mathbf{Acc_{attack}}$, reported for all CIFAR10 classes. Reported as Top-1 accuracy of 1 pixel attack for the undefended model, the model defended by our method (I) and other comparable approaches. All images in the dataset are attached for this report.

Defence	$\boldsymbol{Acc_{clean}}$		$\boldsymbol{Acc_{attack}}$	
	EfficientNet	**VGG**	**EfficientNet**	**VGG**
Undefended	98	98.9	44.3	35.9
I/Ours	96.3	99	91.3	98.6
PSD	-	99.53	-	97.8

Liu et al	-	89.8	-	91

Table 6. Defense Success Rate (B) of Various Defense Methods Against APA Covering At Least 5% of the Image. Adversary Patches; Toaster, Lipstick, Pineapple, And School-Bus Were Used.

Defence	**Défense Success Rate(β)**		
	CIFAR10	**CIFAR100**	**ILSVRC12**
H/Ours (EfficientNet)	91.3	80.5	-
MI/Ours (ResNet)	95.6	**95.43**	**89.1**
PSNR/Ours (Inc)	**96.12**	**94.3**	83.2
Jujutsu (ResNet)	86.5	55.7	-
LGS	93.2	73.7	-
V-a-N(VGG)	-	91.6	86.8
DW(VGG)	-	-	65.2
DW(Inc)	-	-	66.2
ECViT-B	47.39	-	41.7

Table 7. Defense Success Rate (B) of Various Defense Methods Against TnT Covering at Least 5% of the Image.

Defence	**Défense Success Rate(β)**		
	CIFAR100	**CIFAR100**	**ILSVRC12**
H/Ours (EfficientNet)	91.3	80.5	-
MI/Ours (ResNet)	95.6	**95.43**	**89.1**
PSNR/Ours (Inc)	**96.12**	**94.3**	83.2
Texture-based			
Anomaly detection			

Table 8. Defense Success Rate (B) of Various Defense Methods Against Distortion Attacks.

Defence	**Défense Success Rate(β)**		
	CIFAR100	**CIFAR100**	**ILSVRC12**
H/Ours (EfficientNet)	91.3	80.5	-
MI/Ours (ResNet)	95.6	**95.43**	**89.1**
PSNR/Ours (Inc)	**96.12**	**94.3**	83.2
Gradient-based			
Saliency-based			

4.3. Attack Size Impact on Model Performance

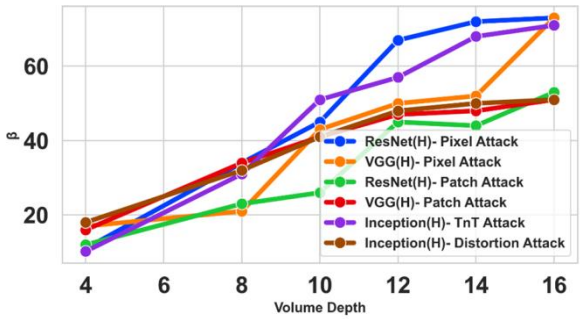


Figure 9. Impact of Patch size(Volume Depth) on Defense Success Rate (B) Against Pixel Attack on CIFAR10 Dataset. the Training Samples Are of Shape (32, 32, 3). We Generated Volumes Staring with a Patch of Size 16 by 16 With Depth 4, All the Way to a Patch Size of 4 by 4 With Depth 16.

Figure 8shows the defense success rate (β) for all four models under N-pixel and APA attacks, respectively, using a depth of 16 for the volumizer algorithm. We notice defense success decreases when attacked pixels surpass the patch size of the volume. This is due to the volumization algorithm's design, which focuses on small attacks and becomes less effective when perturbations exceed the patch size or span multiple slices. This limitation is more prominent if the attack covers a large image portion, potentially obscuring important object details.

4.4. Class Generalization

Figure 9depicts VGG generalization performance on CIFAR10 test data. The undefended model exhibits an AUC of 0.5, while the defended model achieves an AUC of 0.69 under

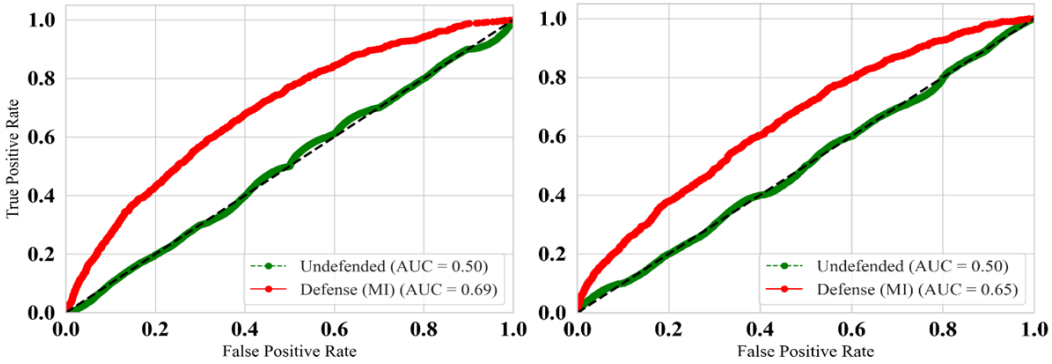


Figure9. (Left) ROC of VGG Model Under 1-Pixel Attack. (Right) ROC of VGG Model Under APA of Size 8 by 8 Pixels. Class Generalization Performance Comparison Between Defended and Undefended VGG on CIFAR10 Test Set. the Plots are Micro Average ROC Curve Across the 10 Classes.

1-pixel attack while the same model achieves AUC of 0.65 under APA. This indicates that the defended model shows improved performance in terms of class generalization compared to the undefended model. These plots suggest that the defended model has better discriminative power and can effectively distinguish between different classes in the CIFAR10 dataset when under 1 pixel and APA attacks. This improvement in AUC demonstrates the effectiveness of our proposed approach in enhancing the model's class generalization capabilities when under localized universal attacks.

4.5. Ablation Study

We conduct an ablation study to assess the impact of volume depth and curriculum learning in our defense methodology.

4.5.1. Impact of CL on Défense Success Rate

For this experiment, we train our models with and without the curriculum learning phase and compare the results with the fully trained models. The results are presented in Table 9**.**

Table9**.** Impact of Curriculum Learning and Volumizer: $\mathbf{Acc_{clean}}$And $\mathbf{Acc_{attack}}$For Models Trained Without Curriculum Learning (I-Vol), Models Trained with Curriculum Learning but Without Volumizer (I-CL), And Fully Trained Models (I).

Method	Acc_{clean}			Acc_{attack}		
	EffNet	**VGG**	**Inception**	**EffNet**	**VGG**	**Inception**
I-Vol	93.7	97.2	97.4	90.6	95.7	95.2
I-CL	97.9	99.4	98.6	56.8	38.1	12.3
I	96.3	99	99	91.3	98.6	96.12

The first scenario we tested was our method without CL but with the volumizer (I-Vol). The performance under this configuration was reasonably good, with the Acc_{clean}being 93.7%, 97.2%, and 97.4% for EfficientNet , VGG, and Inception, respectively. However, the $Acc_{attack}$was markedly lower, specifically, it was 81.6% for EfficientNet , 89.7% for VGG, and 85.2% for Inception. Compared to the full method (I), the $Acc_{attack}$was lower by 9.7%, 8.9%, and 10.9%, respectively. This indicates the effectiveness of Curriculum Learning in improving the model's robustness against adversarial attacks. Second, we studied the effect of Curriculum Learning without the volumizer (I-CL). This configuration achieved even higher Acc_{clean}scores, specifically 97.9%, 99.4%, and 98.6% for EfficientNet , VGG, and Inception, respectively. However, the Acc_{attack}suffered significantly without the volumizer. For EfficientNet , VGG, and Inception, the Acc_{attack}were 56.8%, 38.1%, and 12.3%, respectively, revealing drops of 34.5%, 60.5%, and 83.8% compared to the full I syllabus. This demonstrates the vital role the volumizer plays in enhancing the model's resilience to adversarial attacks.

Lastly, our fully implemented method (I), incorporating both Curriculum Learning and the volumizer, consistently outperformed the other configurations in terms of Acc_{attack}, achieving 91.3%, 98.6%, and 96.12% for EfficientNet , VGG, and Inception, respectively. These figures indicate the combined effect of both components in improving the model's resilience to adversarial attacks.

4.6. Timing Information

Inference and training time comparisons between our method and undefended models are presented in Table 9. An inference overhead for a model protected with our method is noticeable compared to the undefended models– around 6 milliseconds on average across all three models. This increased latency is primarily due to the modifications made to the model architecture to accommodate our defense strategy. Additionally, our defense incurs a significant overhead during training. Depending on the size of the dataset, the additional training time can span from hours to days. However, this process only needs to be run once, as does preprocessing dataset a priori. All inference runs used an NVIDIA RTX A4000, while training was conducted on a node equipped with four RTX A100 GPUs. Despite the increased computational demands, the benefits of enhanced security provided by our defense method offer a worthwhile trade-off.

Table10. Inference Time (ms) For VGG, Resnet, And Inception Trained on ImageNet and the Same Model with Our Defense Based on I Syllabus And a 16 Depth Volumized Inputs.

Method	VGG	ResNet	Inception
Undefended	0.12	0.14	0.09
I/Ours	0.23	0.18	0.16

5. CONCLUSION

We introduced a proactive defence approach against localized adversarial attacks, which preserves model performance on clean data. Our method combines a volumization algorithm that converts 2D images into 3D volumetric representations while maintaining spatial relationships, increasing resilience to perturbations. Additionally, we employ a deep curriculum learning optimization strategy, ordering training samples by complexity, enabling progressive learning from simple to complex samples. By incorporating these techniques into popular CNN architectures, we demonstrated the effectiveness of our method against N-pixel and patch attacks. Experimental results indicated improved robustness without sacrificing performance on cleandata, confirming our approach's ability to enhance image classification model resilience against localized adversarial attacks.

REFERENCES

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," *ArXiv14126572 Cs Stat*, Dec. 2014, Accessed: Nov. 10, 2018. [Online]. Available: http://arxiv.org/abs/1412.6572

[2] N. Akhtar, A. Mian, N. Kardan, and M. Shah, "Advances in adversarial attacks and defenses in computer vision: A survey." arXiv, Sep. 02, 2021. doi: 10.48550/arXiv.2108.00401.

[3] D. Karmon, D. Zoran, and Y. Goldberg, "LaVAN: Localized and Visible Adversarial Noise." arXiv, Mar. 01, 2018. doi: 10.48550/arXiv.1801.02608.

[4] H. Ghebrechristos and G. Alaghband, "Deep curriculum learning optimization," *SN Comput. Sci.*, vol. 1, no. 5, p. 245, Jul. 2020, doi: 10.1007/s42979-020-00251-7.

[5] J. Su, D. V. Vargas, and S. Kouichi, "One pixel attack for fooling deep neural networks," Oct. 2017, Accessed: Nov. 12, 2018. [Online]. Available: https://arxiv.org/abs/1710.08864

[6] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial Patch," *ArXiv171209665 Cs*, May 2018, Accessed: Mar. 12, 2022. [Online]. Available: http://arxiv.org/abs/1712.09665

[7] C. Szegedy *et al.*, "Intriguing properties of neural networks," *ArXiv13126199 Cs*, Dec. 2013, Accessed: Oct. 26, 2018. [Online]. Available: http://arxiv.org/abs/1312.6199

[8] T. Gittings, S. Schneider, and J. Collomosse, "Robust Synthesis of Adversarial Visual Examples Using a Deep Image Prior." arXiv, Jul. 03, 2019. doi: 10.48550/arXiv.1907.01996.

[9] X. Dong *et al.*, "GreedyFool: Distortion-Aware Sparse Adversarial Attack," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2020, pp. 11226–11236. Accessed: Feb. 25, 2024. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/8169e05e2a0debcb15458f2cc1eff0ea-Abstract.html

[10] B. G. Doan, M. Xue, S. Ma, E. Abbasnejad, and D. C. Ranasinghe, "TnT Attacks! Universal Naturalistic Adversarial Patches Against Deep Neural Network Systems," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 3816–3830, Jan. 2022, doi: 10.1109/TIFS.2022.3198857.

[11] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks." arXiv, Mar. 14, 2016. doi: 10.48550/arXiv.1511.04508.

[12] S. Baluja and I. Fischer, "Learning to Attack: Adversarial Transformation Networks," *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, Art. no. 1, Apr. 2018, doi: 10.1609/aaai.v32i1.11672.

[13] T. Bai *et al.*, "AI-GAN: Attack-Inspired Generation of Adversarial Examples." arXiv, Jan. 12, 2021. doi: 10.48550/arXiv.2002.02196.

[14] D. Chen, R. Xu, and B. Han, "Patch Selection Denoiser: An Effective Approach Defending Against One-Pixel Attacks," in *Neural Information Processing*, T. Gedeon, K. W. Wong, and M. Lee, Eds., in Communications in Computer and Information Science. Cham: Springer International Publishing, 2019, pp. 286–296. doi: 10.1007/978-3-030-36802-9_31.

[15] Z.-Y. Liu, P. S. Wang, S.-C. Hsiao, and R. Tso, "Defense against N-pixel Attacks based on Image Reconstruction," in *Proceedings of the 8th International Workshop on Security in Blockchain and Cloud Computing*, in SBC '20. New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 3–7. doi: 10.1145/3384942.3406867.

[16] S. A. A. Shah, M. Bougre, N. Akhtar, M. Bennamoun, and L. Zhang, "Efficient Detection of Pixel-Level Adversarial Attacks," in *2020 IEEE International Conference on Image Processing (ICIP)*, Oct. 2020, pp. 718–722. doi: 10.1109/ICIP40778.2020.9191084.

[17] M. A. Husnoo and A. Anwar, "Do not get fooled: Defense against the one-pixel attack to protect IoT-enabled Deep Learning systems," *Ad Hoc Netw.*, vol. 122, p. 102627, Nov. 2021, doi: 10.1016/j.adhoc.2021.102627.

[18] H. Li and Z. Lin, "Accelerated Proximal Gradient Methods for Nonconvex Programming," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2015. Accessed: Aug. 01, 2022. [Online]. Available: https://papers.nips.cc/paper/2015/hash/f7664060cc52bc6f3d620bcedc94a4b6-Abstract.html

[19] A. Levine and S. Feizi, "(De)Randomized Smoothing for Certifiable Defense against Patch Attacks." arXiv, Jan. 08, 2021. doi: 10.48550/arXiv.2002.10733.

[20] J. H. Metzen and M. Yatsura, "Efficient Certified Defenses Against Patch Attacks on Image Classifiers." arXiv, Feb. 08, 2021. doi: 10.48550/arXiv.2102.04154.

[21] J. Hayes, "On Visible Adversarial Perturbations & Digital Watermarking," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Salt Lake City, UT, USA: IEEE, Jun. 2018, pp. 1678–16787. doi: 10.1109/CVPRW.2018.00210.

[22] M. Naseer, S. Khan, and F. Porikli, "Local Gradients Smoothing: Defense Against Localized Adversarial Attacks," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2019, pp. 1300–1307. doi: 10.1109/WACV.2019.00143.

[23] E. Chou, F. Tramèr, and G. Pellegrino, "SentiNet: Detecting Localized Universal Attacks Against Deep Learning Systems." arXiv, May 09, 2020. doi: 10.48550/arXiv.1812.00292.

[24] Z. Chen, P. Dash, and K. Pattabiraman, "Turning Your Strength against You: Detecting and Mitigating Robust and Universal Adversarial Patch Attacks." arXiv, Jun. 23, 2022. Accessed: Aug. 02, 2022. [Online]. Available: http://arxiv.org/abs/2108.05075

[25] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "SmoothGrad: removing noise by adding noise." arXiv, Jun. 12, 2017. doi: 10.48550/arXiv.1706.03825.

[26] Z. Chen, P. Dash, and K. Pattabiraman, "Jujutsu: A Two-stage Defense against Adversarial Patch Attacks on Deep Neural Networks," in *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, in ASIA CCS '23. New York, NY, USA: Association for Computing Machinery, Jul. 2023, pp. 689–703. doi: 10.1145/3579856.3582816.

[27] T. Combey, A. Loison, M. Faucher, and H. Hajri, "Probabilistic Jacobian-based Saliency Maps Attacks." arXiv, Dec. 10, 2020. doi: 10.48550/arXiv.2007.06032.

[28] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," *ArXiv13126034 Cs*, Dec. 2013, Accessed: Apr. 02, 2018. [Online]. Available: http://arxiv.org/abs/1312.6034

[29] H. Ghebrechristos, "Deep-CLO: Enhancing Feature Extraction from 2D and 3D Topological Data," 2020.

[30] G. Ras, L. Ambrogioni, P. Haselager, M. A. J. van Gerven, and U. Güçlü, "Explainable 3D Convolutional Neural Networks by Learning Temporal Transformations." arXiv, Jun. 29, 2020. doi: 10.48550/arXiv.2006.15983.

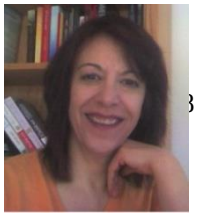
[31] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," *ArXiv160508695 Cs*, May 2016, Accessed: Jun. 23, 2018. [Online]. Available: http://arxiv.org/abs/1605.08695

[32] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A Python toolbox to benchmark the robustness of machine learning models." arXiv, Mar. 20, 2018. doi: 10.48550/arXiv.1707.04131.

AUTHORS

Henok E. Ghebrechristos is a graduate studentspecializing in deep learning and computer vision. He received his B.S. degree in Physics from the University of Colorado, Boulder in 2010. His research interest includes gaining insight from deep learning black box by controlling convolution operations and feature maps generation.

Gita Alaghband is professor and Chair ofComputer Science and Engineering. Her research interests in parallel processing and distributed systems include application



programs and algorithm design, computer architectures, operating systems, performance evaluation, and simulation.