*algorithms*

MDPI

*Article*

# An Evasion Attack against Stacked Capsule Autoencoder

**Jiazhu Dai * and Siwei Xiong**

School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China; qing8liang@shu.edu.cn

* Correspondence: daijz@shu.edu.cn

**Abstract:** Capsule networks are a type of neural network that use the spatial relationship between features to classify images. By capturing the poses and relative positions between features, this network is better able to recognize affine transformation and surpass traditional convolutional neural networks (CNNs) when handling translation, rotation, and scaling. The stacked capsule autoencoder (SCAE) is a state-of-the-art capsule network that encodes an image in capsules which each contain poses of features and their correlations. The encoded contents are then input into the downstream classifier to predict the image categories. Existing research has mainly focused on the security of capsule networks with dynamic routing or expectation maximization (EM) routing, while little attention has been given to the security and robustness of SCAEs. In this paper, we propose an evasion attack against SCAEs. After a perturbation is generated based on the output of the object capsules in the model, it is added to an image to reduce the contribution of the object capsules related to the original category of the image so that the perturbed image will be misclassified. We evaluate the attack using an image classification experiment on the Mixed National Institute of Standards and Technology Database (MNIST), Fashion-MNIST, and German Traffic Sign Recognition Benchmark (GTSRB) datasets, and the average attack success rate can reach 98.6%. The experimental results indicate that the attack can achieve high success rates and stealthiness. This finding confirms that the SCAE has a security vulnerability that allows for the generation of adversarial samples. Our work seeks to highlight the threat of this attack and focus attention on SCAE's security.

**Keywords:** machine learning; adversarial perturbation; evasion attack; stacked capsule autoencoder



**Citation:** Dai, J.; Xiong, S. An Evasion Attack against Stacked Capsule Autoencoder. *Algorithms* **2022**, *15*, 32. https://doi.org/10.3390/a15020032

Academic Editor: Xingjuan Cai

Received: 20 December 2021
Accepted: 17 January 2022
Published: 19 January 2022



## 1. Introduction

Image recognition is a popular research topic in machine learning, and convolutional neural networks (CNNs) are among the major methods addressing this task. CNNs abstract images into local features through operations, such as convolution and pooling, and the local features are then used as evidence for identification. CNNs are able to meet the requirements of common image recognition tasks, but they cannot handle images after affine transformation well, such as rotation and resizing.

A capsule network is a type of neural network that is designed to improve the performance of traditional CNNs when handling images after affine transformation. Based on abstraction, the capsule network further analyzes the spatial relationship between features to promote the reliability of the classification. The stacked capsule autoencoder (SCAE) is a state-of-the-art capsule network which has a different structure from the past capsule networks. First, the poses of features and the relationship between features are extracted from the image. Then, they are combined and encoded into capsules. Finally, the predicted result of the image is obtained by inputting the output of the capsules into the classifier. One of the highlights of the SCAE is the unsupervised classification on the capsule network, which uses bipartite graph matching [1] to find the permutation of cluster indices after finding multiple clusters using the k-means approach.

Recent studies have found that capsule networks also face security threats [2–5]. However, these studies only focus on capsule networks based on dynamic routing. To the

best of our knowledge, the potential security vulnerabilities of SCAE have not been studied. In this paper, we address this issue. The contributions of the paper can be summarized as follows:

1. We propose an evasion attack against the SCAE in which the attacker can compute the perturbation based on the output of the capsules in the model to generate adversarial samples that lead to misclassification of the SCAE;
2. The attack achieves a high attack success rate on various datasets, which confirms that the SCAE has a security vulnerability that allows for the generation of adversarial samples without changing the original structure of the image to fool the unsupervised classifier in the SCAE.

The remainder of this paper is arranged as follows: Section 2 introduces the related works. Section 3 describes the architecture of the SCAE and explains its operations. Section 4 presents our attack method and algorithm in detail. Section 5 describes the experiments and the results. Finally, Section 6 provides a summary and briefly presents direction for future work.

## 2. Related Works

### *2.1. Capsule Network*

Capsule networks have been developed through three main stages: the dynamic routing capsule network proposed by Sabour et al. [6] in 2017, the EM routing capsule network proposed by Hinton et al. [7] in 2018, and the stacked capsule autoencoder (SCAE) proposed by Kosiorek et al. [8] in 2019.

The SCAE, as a state-of-the-art capsule network, uses autoencoders instead of a routing algorithm and conducts both supervised and unsupervised classification at the same time. The whole model is composed of a part capsule autoencoder (PCAE) and an object capsule autoencoder (OCAE). After the image is input into the model, the parts of each object in the image are extracted by the PCAE and then combined into whole objects by the OCAE. The classifier makes predictions according to the presence of different parts and objects. Each capsule in the PCAE contains a six-dimensional pose vector, a one-dimensional presence probability, and an n-dimensional attribute vector. The OCAE uses a set transformer [9] to encode part capsules into object capsules. Regarding the classifiers, the SCAE uses linear classifiers for supervised classification and k-means classifiers for unsupervised classification.

The main contribution of the SCAE is that it provides a new learning method using the PCAE to segment images into multiple parts and the OCAE to make the parts into whole objects. This method considers the spatial relationship between features and the variety of representations of similar features; therefore, it is less vulnerable to random perturbation. The SCAE can achieve a high unsupervised classification accuracy of 98.7% on the MNIST dataset.

### *2.2. Poisoning Attacks and Evasion Attacks*

The major security threats in machine learning are adversarial attacks, which can be categorized into two types: poisoning attacks and evasion attacks.

Poisoning attacks, which aim at degrading the performance of the model or creating a backdoor in the model so as to control its behavior, occur during the training phase. The attacker adds elaborately constructed malicious samples to the training set, thus causing the trained model to output the attacker's expected results for specific samples or reducing the classification accuracy of the model at test time [10–20].

Evasion attacks, which aim at creating adversarial samples to deceive the model, occur during the testing phase. The attacker adds a carefully constructed perturbation to the clean sample to form a malicious sample. Its appearance is generally consistent with that of the clean sample, but the model will misclassify it or make a prediction specified by the attacker [21–33]. The adversarial attack proposed in this paper is an evasion attack.

According to the information held by the attacker, adversarial attacks can also be categorized into white-box attacks and black-box attacks. In white-box attacks, the attacker can obtain the detail of the model, including the structure and parameters of the model. While in black-box attacks, only the inputs and outputs are available for the attacker, and the model itself remains sealed.

### *2.3. Security Threats of the Capsule Network*

After the emergence of the capsule network, research on its security focused on the dynamic routing capsule network. Jaesik [2] provided a variety of successful methods of adversarial attacks on capsule networks. Michels et al. [3] proved that the ability of encapsulated networks to resist white-box attacks is not better than that of traditional CNNs. Marchisio et al. [4] designed a black-box attack algorithm against a capsule network and verified its effectiveness on the German Traffic Sign Recognition Benchmark (GTSRB) dataset. De Marco [5] proved that capsule networks with different scales are vulnerable to adversarial attacks to varying degrees.

The SCAE has a different structure than previous capsule networks, and its robustness also needs to be analyzed. To the best of our knowledge, few research reports have focused on the SCAE. In this paper, we study the vulnerability of the SCAE to adversarial attacks to help improve its robustness to such security threats.

## 3. Stacked Capsule Autoencoder

The complete structure of the SCAE is shown in Figure 1. The SCAE treats an image as a composition of multiple objects that each consist of multiple parts. These parts and objects will be analyzed by two main units of the model, that is, the part capsule autoencoder (PCAE) and the object capsule autoencoder (OCAE). First, after the image is input into the model, a CNN will be used by the PCAE to extract the poses, presences and features of parts that constitute the objects in the image, which are stored in part capsules. Each part capsule represents a part that may exist in the image. Next, the OCAE uses a set transformer [9] to conduct the autoencoding operation, which combines these scattered parts into complete objects, calculates the pose and presence of each object, and stores them in object capsules. Each object capsule represents an object that may exist in the image. Finally, the classifier will use the capsule output encoded by the OCAE to predict the label of the image.

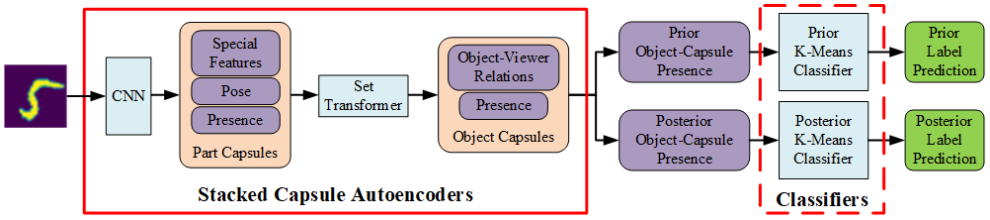


**Figure 1.** SCAE's architecture.

There are two forms of SCAE output: the prior object-capsule presence with dimension $[B, K]$ and the posterior object-capsule presence with dimension $[B, K, M]$, where $B$ is the batch size, $K$ is the number of object capsules, and $M$ is the number of part capsules. The output of the SCAE is the probability of existence of each object in the image, and its value ranges from 0 to 1. The classifier can select any one of the above outputs for k-means clustering and then use bipartite graph matching [1] to find the mapping permutation between clustering labels and ground truth labels. During the test stage, the output of the SCAE is input into the classifier to obtain the predicted label of the image. It should be noted that if the posterior object-capsule presence is used, dimension $M$ needs to be reduced and its values need to be summed to ensure shape consistency. Our attack in this paper targets the classification with the prior k-means classifier and posterior k-means classifier.

## 4. Proposed Evasion Attack

The proposed evasion attack is a white-box attack that consists of two steps. In the first step, we identify the object capsule subset $S$ that contributes most to the original label according to the output of the SCAE model $E$ on input image $x$, and obtain the target function $f(x+p)$ for computing the perturbation, which is described in detail in Section 4.1. In the second step, inspired by [25,26,33], we design the following three algorithms to compute the minimum perturbation $p$, which aim to decrease the sum of the output of the object capsules belonging to subset $S$ so that the perturbed images are misclassified:

- **Computing the Perturbation with Gradient Direction Update (GDU)**: in this algorithm, the adversarial sample is iteratively updated according to the direction of gradient of the target function $f(x+p)$;
- **Computing the Perturbation with Pixel Saliency to Capsules (PSC)**: in this algorithm, we iteratively choose and modify a pair of pixels which contributes most to the value of the target function $f(x+p)$ until the adversarial sample cannot be classified correctly;
- **Computing the Perturbation with Optimizer (OPT)**: in this algorithm, an optimizer is used to minimize the value of the target function $f(x+p)$ so as to generate the perturbation $p$ which causes misclassification.

The detail of the three algorithms is presented in Section 4.2.

### *4.1. Identifying the Object Capsule Subset*

For image $x$ and the SCAE model $E$, let $E(x)$ be the $K$-dimensional object-capsule presence that the SCAE outputs on $x$ and let $C(E(x))$ be the final classification result of $x$ on the classifier $C$. The goal of our attack is to find a perturbation $p$ with the minimum $\|p\|_2$ that makes the classifier $C$ misclassify $E(x+p)$. This problem is defined as follows:

$$\begin{array}{rl} \text{Minimize} & \|p\|_2 \\ s.t. & C(E(x+p)) \neq C(E(x)) \\ & x+p \in [0,1]^n \end{array} \tag{1}$$

Due to the sparsity loss used during training, different capsules are often associated with different classes. For an input image, only object capsules related to the label of the image will be activated and output with high presence, while the irrelevant object capsules remain inactive. This feature makes it possible to find the object capsule subset $S$ related to the image. To misclassify a perturbed image $x+p$, that is, $C(E(x+p)) \neq C(E(x))$, we can lower the output of the object capsule subset $S$ to reduce the contribution of this output to the original classification. In other words, we want the SCAE to mistakenly believe that "the objects that belong to the real category do not appear in the image", which is shown in Formula (2):

$$\text{Minimize} \quad f(x+p) = \sum_{i \in S} E(x+p)_i \tag{2}$$

For a well-trained SCAE model, the output object-capsule presence should be close to 1 or 0 (that is, the SCAE can accurately distinguish whether the objects represented by each capsule appear in the image). According to this feature, we compute the $K$-dimensional object-capsule presence $E(x)$ for $x$, calculate the average presence $\overline{E(x)} = \frac{1}{K}\sum_{i=1}^{K} E(x)_i$, and finally obtain the activated object capsule subset $S = \left\{ i \middle| E(x)_i > \overline{E(x)} \right\}$.

### *4.2. Computing the Perturbation*

Based on Formula (2), we design three algorithms to compute the minimum perturbation $p$ which aims to decrease the sum of the output of the object capsules belonging to subset $S$ and cause the perturbed image to be misclassified.

4.2.1. Computing the Perturbation with Gradient Direction Update

In this algorithm, we compute the sign of the gradient of Formula (2) to the image $x$, and obtain the direction to update the value of each pixel, which decreases $\sum_{i \in S} E(x)_i$. The adversarial sample is generated in this direction:

$$x_0^{adv} = x, \;\; x_{N+1}^{adv} = \mathrm{clip}_{0,1}\left(x_N^{adv} - \alpha \cdot \mathrm{sign}\left(\nabla_{x_N^{adv}} f\left(x_N^{adv}\right)\right)\right) \tag{3}$$

where $x_N^{adv}$ is the generated adversarial sample based on image $x$, $\alpha > 0$ is a hyperparameter that represents the step of change in each iteration, and $\nabla_x f(x)$ represents the gradient of $x$ to $f(x)$. The pixel values of the generated sample are clipped between 0 and 1. The whole procedure is shown in Algorithm 1:

**Algorithm 1:** Generating the perturbation with gradient direction update.

**Input:** Image $x$, SCAE model $E$, classifier $C$, hyperparameter $\alpha$, the number of iterations $n_{iter}$.
**Output:** Perturbation $p$.

Initialize $x^{adv} \leftarrow x$, $x_0^{adv} \leftarrow x$, $\mathcal{L}_p \leftarrow +\infty$.
$S \leftarrow \left\{ i \middle| E(x)_i > \frac{1}{K} \sum_{j=1}^{K} E(x)_j \right\}$
**for** $i$ in $n_{iter}$ **do**
    */* Compute next $x_i^{adv}$ */*
    $x_{i+1}^{adv} \leftarrow \mathrm{clip}_{0,1}\left(x_i^{adv} - \alpha \cdot \mathrm{sign}\left(\nabla_{x_i^{adv}} \sum_{j \in S} E\left(x_i^{adv}\right)_j\right)\right)$
    */* Judge whether the current result is the best one */*
    **if** $C\left(E\left(x_{i+1}^{adv}\right)\right) \neq C(E(x))$ **and** $\|x_{i+1}^{adv} - x\|_2 < \mathcal{L}_p$ **do**
        $x^{adv} \leftarrow x_{i+1}^{adv}$
        $\mathcal{L}_p \leftarrow \|x_{i+1}^{adv} - x\|_2$
    **end if**
**end for**
$p \leftarrow x^{adv} - x$
**return** $p$

4.2.2. Computing the Perturbation with Pixel Saliency to Capsules

In this algorithm, we iteratively choose a pair of pixels that contributes most to the value of Formula (2). The amount of contribution of a pixel pair $(u, v)$ is named pixel saliency, which is defined as follows:

$$\begin{aligned} \delta_1 &= \sum_{i=u,v} \left( \nabla_{x_i} \sum_{j \in S} E(x)_j \right) \\ \delta_2 &= \sum_{i=u,v} \left( \nabla_{x_i} \sum_{j \notin S} E(x)_j \right) \end{aligned} \tag{4}$$

where $\delta_1$ represents the influence on the sum of the output of the object capsule subset $S$ when the pixel pair $(u, v)$ changes, and $\delta_2$ represents the influence on the sum of the output of the object capsules not belonging to $S$. The pixels for modification are picked according to the following rule:

$$(u^*, v^*) = \underset{(u,v)}{\mathrm{argmax}}(-\delta_1 \times \delta_2 \times (\delta_1 > 0) \times (\delta_2 < 0)) \tag{5}$$

where $\delta_1 > 0$ means the sum of the output of the capsules in subset $S$ increases with the values of the pixels, and $\delta_2 < 0$ means the sum of the output of the other capsules decreases when the values of the pixels increase. Therefore, by reducing the value of the pixels, we can decrease the sum of the output of the capsules in subset $S$.

Before the iteration, a pixel search domain $\Gamma$ is initialized including all pixels whose values are above 0, which are available for modification. We use an iterative algorithm to compute the adversarial sample. In each iteration, we obtain the pair of pixels with the largest pixel saliency from $\Gamma$, and reduce the values of the two pixels by $\alpha$, which is

a positive hyperparameter that represents the step of change. During the iteration, the modification of $\Gamma$ follows this principle: if the value of a pixel reaches 0 after being chosen and decreased, then this pixel will be removed from $\Gamma$ since 0 is the minimum value of a pixel and further subtraction is meaningless. All pixels not in $\Gamma$ are excluded from the algorithm.

The algorithm halts as soon as the current perturbation causes misclassification to avoid affecting more pixels. On the other hand, the algorithm will also stop when $\Gamma$ is empty, which means that the attack fails. The whole procedure is shown in Algorithm 2:

**Algorithm 2:** Generating the perturbation with pixel saliency to capsules.

**Input**: Image $x$, SCAE model $E$, classifier $C$, hyperparameter $\alpha$, the number of iterations $n_{iter}$.
**Output**: Perturbation $p$.

*/* Initial feature search domain */*
Initialize $x^{adv} \leftarrow x$, $\Gamma \leftarrow \{i | x_i > 0\}$.
$S \leftarrow \left\{ i \middle| E(x)_i > \frac{1}{K} \sum_{j=1}^{K} E(x)_j \right\}$
**for** $i$ in $n_{iter}$ **do**
    **if** $C\left(E\left(x^{adv}\right)\right) \neq C(E(x))$ **or** $\Gamma$ is empty **do**
        **break**
    **end if**
    */* Choose the two pixels for modification */*
    $m \leftarrow 0$
    **for** each pair $(u, \mathrm{v})$ in $\Gamma$ **do**
        $\delta_1 = \sum_{i=u,v} \left( \nabla_{x_i^{adv}} \sum_{j \in S} E\left(x^{adv}\right)_j \right)$
        $\delta_2 = \sum_{i=u,v} \left( \nabla_{x_i^{adv}} \sum_{j \notin S} E\left(x^{adv}\right)_j \right)$
        **if** $-\delta_1 \times \delta_2 \times (\delta_1 > 0) \times (\delta_2 < 0) > m$ **do**
            $u^*, v^* \leftarrow u, v$
            $m \leftarrow -\delta_1 \times \delta_2$
        **end if**
    **end for**
    */* Decrease the values of the two pixels */*
    $x_{u^*}^{adv} \leftarrow \max\left(x_{u^*}^{adv} - \alpha, 0\right)$
    $x_{v^*}^{adv} \leftarrow \max\left(x_{v^*}^{adv} - \alpha, 0\right)$
    **if** $x_{u^*}^{adv} = 0$ **do**
        Remove $u^*$ from $\Gamma$.
    **end if**
    **if** $x_{v^*}^{adv} = 0$ **do**
        Remove $v^*$ from $\Gamma$.
    **end if**
**end for**
$p \leftarrow x^{adv} - x$
**return** $p$

#### 4.2.3. Computing the Perturbation with Optimizer

In this algorithm, an optimizer is used to minimize the value of Formula (2). To misclassify a perturbed image $x + p$, that is, $C(E(x + p)) \neq C(E(x))$, we can lower the output of the object capsule subset $S$ by decreasing the value of $f(x + p)$ to reduce their contribution to the original classification. Therefore, the optimization problem in Formula (1) can be transformed into the one in Formula (6):

$$\begin{array}{cc} \text{Minimize} & \|p\|_2 + \alpha \cdot f(x+p) \\ s.t. & x + p \in [0,1]^n \end{array} \tag{6}$$

where $\alpha > 0$ is a suitably chosen hyperparameter, which ensures that the two parts in Formula (6) can be optimized simultaneously. The amount of perturbation $\|p\|_2$ is also taken into consideration and minimized by the optimizer, so that the generated adversarial sample can obtain better stealthiness.

For the box constraints in Formula (6), namely, $x + p \in [0, 1]^n$, instead of optimizing over the variable $p$ and clipping the pixel values of $x + p$ directly, we introduce a new

variable $p'$ according to the "change of variables" method given by Carlini et al. [25] and optimize over it instead. The relationship between the two variables is as follows:

$$\begin{aligned} w &= \operatorname{arctanh}(2x-1) \\ p &= \tfrac{1}{2}(\tanh(w+p')+1) - x \end{aligned} \tag{7}$$

As $x \in [0,1]^n$, $w \in (-\infty,+\infty)^n$. During the experiments, we use $\operatorname{arctanh}((2x-1)*\epsilon)$ to avoid dividing by zero. For $p' \in (-\infty,+\infty)^n$, $\tanh(w+p') \in (-1,1)^n$, which leads to $x+p \in (0,1)^n$. The core idea of this method is that the original image $x$ is first transformed into the arctanh space and then mapped back into $[0,1]^n$. Therefore, the calculated image $x+p$ is always valid.

The full algorithm for computing perturbation consists of the inner iteration and the outer iteration.

In the inner iteration, we use an optimizer to iteratively conduct the process of Formula (6). The hyperparameter $\alpha$ is fixed from beginning to end during the computation. When an inner iteration is finished, we select the perturbation $p$ with the smallest $\|p\|_2$ from those satisfying $C(E(x+p)) \neq C(E(x))$ as the best inner result.

In the outer iteration, we initialize the optimizer, execute a complete inner iteration, and finally update the hyperparameter $\alpha$ to find the most suitable value of $\alpha$ so that the optimizer can obtain perturbation $p$ with a smaller $L_2$ norm. We perform multiple rounds of outer iterations, and finally select the perturbation $p$ with the smallest $\|p\|_2$ among all best inner results as the global optimal result.

The whole procedure is shown in Algorithm 3:

**Algorithm 3:** Generating the perturbation with optimizer.

**Input**: Image $x$, SCAE model $E$, classifier $C$, optimizer $opt$, hyperparameter $\alpha$, the number of outer iterations $n_{o\_iter}$, the number of inner iterations $n_{i\_iter}$.
**Output**: Perturbation $p$.

Initialize $p \leftarrow 0$, $\mathcal{L}_p \leftarrow +\infty$, $\alpha$.
$S \leftarrow \left\{ i \middle| E(x)_i > \frac{1}{K}\sum_{j=1}^{K} E(x)_j \right\}$
$w \leftarrow \operatorname{arctanh}(2x-1)$
**for** $i$ in $n_{o\_iter}$ **do**
    Init $opt$.
    $p'_0 \leftarrow \operatorname{rand}(0,1)$
    **for** $j$ in $n_{i\_iter}$ **do**
        $x_j^{adv} \leftarrow \frac{1}{2}\left(\tanh\left(w+p'_j\right)+1\right)$
        $\mathcal{L} \leftarrow \|x_j^{adv}-x\|_2 + \alpha \cdot \sum_{i \in S} E\left(x_j^{adv}\right)_i$
        */* The optimizer computes next $p'_j$ */*
        $p'_{j+1} \leftarrow opt\left(p'_j \middle| \mathcal{L}\right)$
        $x_{j+1}^{adv} \leftarrow \frac{1}{2}\left(\tanh\left(w+p'_{j+1}\right)+1\right)$
        */* Judge if the current result is the best one */*
        **if** $C\left(E\left(x_{j+1}^{adv}\right)\right) \neq C(E(x))$ **and** $\|x_{j+1}^{adv}-x\|_2 < \mathcal{L}_p$ **do**
            $\mathcal{L}_p \leftarrow \|x_{j+1}^{adv}-x\|_2$
            $p \leftarrow x_{j+1}^{adv}-x$
        **end if**
    **end for**
    Update $\alpha$.
**end for**
**return** $p$

The update algorithm of the hyperparameter $\alpha$ is as follows: we use a binary search to find the optimal value for $\alpha$. First, we specify the upper and lower bounds for $\alpha$ as $\alpha_{ub}$ and $\alpha_{lb}$ respectively, and assign $\alpha$ an initial value between them. Then, in the inner iterations, if the algorithm can obtain any perturbation $p$ that satisfies $C\left(E\left(x^{adv}\right)\right) \neq C(E(x))$, let $\alpha_{ub} \leftarrow \alpha$; otherwise, $\alpha_{lb} \leftarrow \alpha$. Finally, we take $\alpha \leftarrow (\alpha_{ub}+\alpha_{lb})/2$ as the new value for $\alpha$.

#### 4.2.4. Complexity of the Algorithms

The three proposed algorithms iteratively compute the perturbations. The complexity of the algorithms is shown in Table 1:

**Table 1.** The complexity of the three algorithms: GDU, PSC, and OPT.

| Algorithm | GDU | PSC | OPT |
|---|---|---|---|
| Num of iterations | $n_{iter}$ | Not greater than $n_{iter}$ | $n_{o\_iter} \times n_{i\_iter}$ |
| Gradient calculation per iteration | $\nabla_{x+p} \sum_{i \in S} E(x+p)_i$ | $\nabla_{x+p} \sum_{i \in S} E(x+p)_i + \nabla_{x+p} \sum_{i \notin S} E(x+p)_i$ | $\nabla_p \left( \sum_{i \in S} E(x+p)_i + \|p\|_2 \right)$ |
| Update of adversarial sample per iteration | Update along the sign of the gradient | Update along fixed direction | Update via the optimizer |
| Num of pixels modified per iteration | All pixels | 2 | All pixels |

The complexity of gradient calculation per iteration among the three algorithms has not much difference. OPT, which contains an optimizer and has a larger number of iterations, is more time-consuming than the other two algorithms. PSC is the fastest one among the three algorithms, as it modifies only two pixels at a time and halts as soon as the adversarial sample causes misclassification.

## 5. Experimental Evaluation

### *5.1. Experimental Setup*

The original paper of the SCAE [8] showed that the SCAE can achieve 98.7% accuracy on MNIST, while only achieve 55.33% and 25.01% accuracy on SVHN and CIFAR10, which is not high enough to meet the needs of our experiments. ImageNet, which is similar to CIFAR10, is unsuitable for our experiments as well. Based on the performance of the SCAE on various datasets, we finally choose three datasets for the experiments: MNIST, Fashion-MNIST, and German Traffic Sign Recognition Benchmark (GTSRB), in order to make our experimental results more convincing. We train three SCAE models [8,34] using the main parameter set as shown in Table 2:

**Table 2.** SCAEs' main parameters.

| Dataset | MNIST | Fashion-MNIST | GTSRB |
|---|---|---|---|
| Canvas size | 40 | 40 | 40 |
| Num of part capsules | 24 | 24 | 40 |
| Num of object capsules | 24 | 24 | 64 |
| Num of channels | 1 | 1 | 3 |
| Template size | 11 | 11 | 14 |
| Part capsule noise scale | 4.0 | 4.0 | 0.0 |
| Object capsule noise scale | 4.0 | 4.0 | 0.0 |
| Part CNN | $2 \times (128{:}2) - 2 \times (128{:}1)$ | $2 \times (128{:}2) - 2 \times (128{:}1)$ | $2 \times (128{:}1) - 2 \times (128{:}2)$ |
| Set transformer | $3 \times (1-16) - 256$ | $3 \times (1-16) - 256$ | $3 \times (2-64) - 256$ |

Most of the parameters are equivalent to those given in the supplemental document of the original paper of the SCAE [8]. For part CNN, $2 \times (128{:}2)$ means two convolutional layers with 128 channels and a stride of two. For the set transformer, $3 \times (1-16) - 256$ means three layers, one attention head, 16 hidden units and 256 output units. All three models share the same optimizer shown in Table 3:

**Table 3.** Settings of the optimizer to train the SCAEs.

| Optimizer Parameter | Value |
|---|---|
| Algorithm | RMSProp |
| Learning rate | $3 \times 10^{-5}$ |
| Momentum | 0.9 |
| $\epsilon$ | $1 \times 10^{-6}$ |
| Learning rate decay steps | 10,000 |
| Learning rate decay rate | 0.96 |
| Batch size | 100 |

According to [8], the value of $k$ of the k-means classifiers is set as 10, which equals the number of the categories.

In particular, when training the SCAE on GTSRB, the performance is poor when all 43 categories are used; thus, we choose only 10 categories to ensure that the classification accuracy meets the requirement of our experiments. Furthermore, we use min-max normalization (shown Formula (8)) to reduce the luminance difference between images. The chosen categories are shown in Figure 2.

$$x_i' = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{8}$$


**Figure 2.** The 10 chosen categories of the German traffic sign recognition benchmark dataset used to train the SCAE model.

The classification accuracy of all three models is given in Table 4:

**Table 4.** Accuracy of the SCAE models trained on three datasets (in %).

| Dataset | MNIST | Fashion-MNIST | GTSRB |
|---|---|---|---|
| Prior k-means classifier | 97.82 | 63.20 | 60.11 |
| Posterior k-means classifier | 97.62 | 63.79 | 56.14 |

### *5.2. Experimental Method*

We evaluate our attack on both the prior k-means classifier and the posterior k-means classifier. In each experiment, we choose 5000 samples that can be recognized correctly by the target classifier. The perturbed images are computed using the above three algorithms separately. Then, these images are input into the SCAE, and the encoded results of the SCAE are fed into the corresponding classifier to test the attack success rate (the ratio of the number of samples that are misclassified by the classifier to the total number of all test samples). We compare the three algorithms to find the best algorithm for our evasion attack. The source code of our attack is available at https://github.com/FrostbiteXSW/SCAE_Attack (accessed on 12 January 2022).

The imperceptibility of the perturbation is a challenge on images from the MNIST dataset and the Fashion-MNIST dataset since the background of those images is black (the pixel value is zero) and white noise on the black background can be easily noticed and detected. Therefore, we place mask $m$ on the perturbation to confine its scope to the objects and their vicinity rather than the background of the images to improve the stealthiness of the attack. The mask value $m_i$ at each pixel position $i$ is calculated using Formula (9):

$$m_i = \frac{1}{9}\left(x_i + \sum N_8(x_i)\right) \tag{9}$$

where $m_i$ is the average of the corresponding pixel $x_i$ and its eight neighbors $N_8$. For the perturbation value $p_i$ on each pixel $i$, let $p'_i \leftarrow p_i * m_i$, where $m_i$ is used as the weight of variable $p_i$. In the background area $p'_i = 0$, which is due to the fact that $m_i = 0$; therefore, the background area of the image is not affected by the perturbation.

In the following subsections, we will introduce the setups for each algorithm, and explain how the mask method is applied to them separately.

#### 5.2.1. Experimental Settings for GDU

We set the hyperparameter $\alpha$ to 0.05 and the number of iterations to 100. The mask $m_i$ is multiplied by $\alpha \cdot \text{sign}\left(\nabla_{x_N^{adv}} \sum_{i \in S} E\left(x_N^{adv}\right)_i\right)$ to limit the area affected by the perturbation.

#### 5.2.2. Experimental Settings for PSC

We set the hyperparameter $\alpha$ to 0.5 and the number of iterations to 200. The mask $m_i$ is multiplied by the saliency $\delta_1$ and $\delta_2$ to suppress the saliency of particular pixels thus limiting the area affected by the perturbation.

#### 5.2.3. Experimental Settings for OPT

We set the initial value of hyperparameter $\alpha$ and its upper and lower bounds to 100, $+\infty$ and 0, respectively. If the upper bound of $\alpha$ is $+\infty$ when updating it, then we simply let $\alpha \leftarrow \alpha \times 10$. The mask $m_i$ is multiplied directly by the generated perturbation. The numbers of inner and outer iterations are limited to 300 and 9, respectively, and the optimizer used to compute the perturbations is shown in Table 5:

**Table 5.** Settings of the optimizer used to attack the SCAEs.

| Optimizer Parameter | Value |
|---|---|
| Algorithm | Adam |
| Learning rate | 1.0 |
| $\beta_1$ | 0.9 |
| $\beta_2$ | 0.999 |
| $\epsilon$ | $1 \times 10^{-8}$ |

### *5.3. Results and Discussion*

We perform the experiment on each dataset and each classifier with the above three algorithms, and the results are shown in Tables 6–8 respectively. The attack success rate is the ratio of the number of samples that are misclassified by the classifier to the total number of all test samples. The average $L_2$ norm of $p$ represents the average amount of the perturbations, and a lower value corresponds to an attack with better stealthiness. The standard deviation of $p$ is used to estimate the stability of the attack. A higher standard deviation means that the performance of the attack fluctuates according to different samples.

**Table 6.** Experimental results for the MNIST dataset.

| Classifier | Algorithm for Computing Perturbations | Attack Success Rate | Average $L_2$ Norm of $p$ | Standard Deviation of $p$ |
|---|---|---|---|---|
| Prior k-means classifier | GDU | 0.9790 | 2.6292 | 1.0797 |
| | PSC | **1.0000** | 3.7964 | 1.5333 |
| | OPT | **1.0000** | **1.0839** | **0.6015** |
| Posterior k-means classifier | GDU | 0.8994 | 3.1017 | 1.3118 |
| | PSC | 0.9994 | 4.4982 | 1.6447 |
| | OPT | **1.0000** | **1.1838** | **0.6222** |

**Table 7.** Experimental results for the Fashion-MNIST dataset.

| Classifier | Algorithm for Computing Perturbations | Attack Success Rate | Average $L_2$ Norm of $p$ | Standard Deviation of $p$ |
|---|---|---|---|---|
| Prior k-means classifier | GDU | 0.9836 | 3.0482 | 1.5615 |
| | PSC | 0.9960 | 4.0945 | 1.8384 |
| | OPT | **1.0000** | **1.4160** | **0.8982** |
| Posterior k-means classifier | GDU | 0.9686 | 3.0768 | 1.6260 |
| | PSC | 0.9622 | 4.0196 | 1.9138 |
| | OPT | **1.0000** | **1.3598** | **0.9856** |

**Table 8.** Experimental results for the GTSRB dataset.

| Classifier | Algorithm for Computing Perturbations | Attack Success Rate | Average $L_2$ Norm of $p$ | Standard Deviation of $p$ |
|---|---|---|---|---|
| Prior k-means classifier | GDU | **1.0000** | 2.1903 | 0.8650 |
| | PSC | 0.9778 | 3.7358 | 1.8549 |
| | OPT | **1.0000** | **0.9633** | **0.3490** |
| Posterior k-means classifier | GDU | **1.0000** | 2.7019 | 1.2333 |
| | PSC | 0.9792 | 4.3079 | 2.4017 |
| | OPT | **1.0000** | **0.8301** | **0.5032** |

The above results show that our proposed evasion attack has high attack success rates with all three algorithms. When using GDU or PSC, the average $L_2$ norm of $p$ is larger than that obtained using OPT. Additionally, the perturbations generated by OPT have the smallest standard deviation of $p$ since GDU and PSC focus on misclassification rather than invisibility, and their performance is relatively worse. As OPT takes the amount of perturbation into consideration and is more time-consuming, perturbations generated by it have better stealthiness and stability.

We randomly select one sample from each dataset to perform the attack as examples. The visualization of the perturbed images is shown in Figure 3:

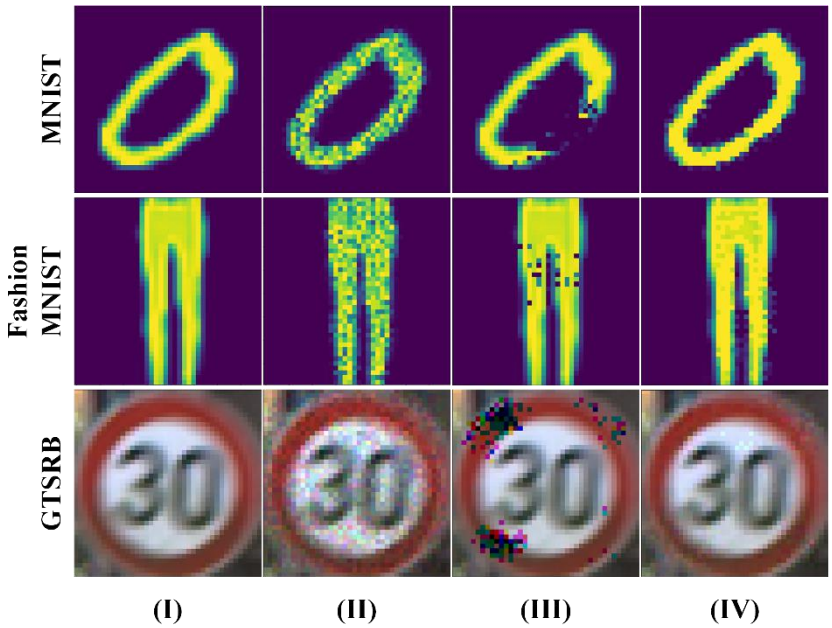


**Figure 3.** (**I**) Original images; (**II**) adversarial samples generated by GDU; (**III**) adversarial samples generated by PSC; and (**IV**) adversarial samples generated by OPT.

Figure 3 shows that for the MNIST and Fashion-MNIST samples, the perturbations are confined to the objects and their surrounding areas, meaning that the adversarial images

have a similar structure to the original images, which greatly improves the stealthiness of the attack. The experiments on the GTSRB dataset achieve satisfying results as well. When using OPT to generate perturbations based on our method, the perturbations are nearly invisible and the adversarial samples can hardly be distinguished from normal samples; thus, it performs the best among the three algorithms.

The results show that although the SCAE is a type of encoder that encodes the image structure, our attack method, which changes neither the image structure nor the original pose of the existing parts in the image, can still induce a change in the coding result of the SCAE and thus lead to misclassification.

There are still some limits of our attack method. The SCAE is designed to handle affine transformation. When the adversarial samples generated by our attack method are rotated or resized, their effectiveness will be affected, which means that our attack method is not capable enough to bypass such situation. However, we have planned to address this issue in the future.

## 6. Conclusions

In this paper, we propose an attack on the state-of-art structure of a capsule network (i.e., the stacked capsule autoencoder). After identifying the object capsule subset related to the original category of the image, an adversarial attack algorithm is used to iteratively compute the perturbation to reduce the presence output by these object capsules. A mask is used when generating the perturbation to improve stealthiness, and the perturbed image will be misclassified by the downstream k-means classifier. The experimental results show that the SCAE can be cheated with a high probability by our attack, and the generated adversarial samples are similar to the clean ones. Through our work, it is confirmed that the SCAE has a security vulnerability in which adversarial samples can be generated without changing the original structure of the image to fool the classifiers. Our work is presented to highlight the threat of this attack and focus attention on SCAE security. In our future work, we will study methods to defend against this attack and improve the robustness of the SCAE.

**Author Contributions:** Conceptualization, J.D. and S.X.; methodology, J.D. and S.X.; software, S.X.; validation, J.D. and S.X.; formal analysis, S.X.; investigation, S.X.; resources, J.D. and S.X.; data curation, S.X.; writing—original draft preparation, S.X.; writing—review and editing, J.D. and S.X.; visualization, S.X.; supervision, J.D.; project administration, J.D.; funding acquisition, J.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the State Scholarship Fund of the China Scholarship Council (Grant No. 201606895018).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** All data included in this study are openly available here: https://github.com/FrostbiteXSW/SCAE_Attack (accessed on 16 January 2022).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Kuhn, H.W. The hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97. [CrossRef]
2. Yoon, J. Adversarial Attack to Capsule Networks. Available online: https://github.com/jaesik817/adv_attack_capsnet (accessed on 11 March 2021).
3. Michels, F.; Uelwer, T.; Upschulte, E.; Harmeling, S. On the vulnerability of capsule networks to adversarial attacks. *arXiv* **2019**, arXiv:1906.03612.
4. Marchisio, A.; Nanfa, G.; Khalid, F.; Hanif, M.A.; Martina, M.; Shafique, M. CapsAttacks: Robust and imperceptible adversarial attacks on capsule networks. *arXiv* **2019**, arXiv:1901.09878.
5. De Marco, A. Capsule Networks Robustness against Adversarial Attacks and Affine Transformations. Ph.D. Thesis, Politecnico di Torino, Turin, Italy, 2020.

6. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic routing between capsules. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; Guyon, I., Luxburg, U., von Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R., Eds.; 2017; pp. 3856–3866. Available online: https://arxiv.org/abs/1710.09829 (accessed on 11 March 2021).
7. Hinton, G.E.; Sabour, S.; Frosst, N. Matrix capsules with EM routing. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018. Conference Track Proceedings.
8. Kosiorek, A.R.; Sabour, S.; Teh, Y.W.; Hinton, G.E. Stacked capsule autoencoders. In Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada, 8–14 December 2019; Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R., Eds.; 2019; pp. 15486–15496. Available online: https://arxiv.org/abs/1906.06818 (accessed on 11 March 2021).
9. Lee, J.; Lee, Y.; Kim, J.; Kosiorek, A.R.; Choi, S.; Teh, Y.W. Set Transformer: A Framework for Attention-Based Permutation-Invariant Neural Networks. In Proceedings of the Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, CA, USA, 9–15 June 2019; Chaudhuri, K., Salakhutdinov, R., Eds.; 2019; Volume 97, pp. 3744–3753.
10. Akhtar, N.; Mian, A.S. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access* **2018**, *6*, 14410–14430. [CrossRef]
11. Chen, X.; Liu, C.; Li, B.; Lu, K.; Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv* **2017**, arXiv:1712.05526.
12. Zhong, H.; Liao, C.; Squicciarini, A.C.; Zhu, S.; Miller, D.J. Backdoor embedding in convolutional neural network models via invisible perturbation. In Proceedings of the CODASPY '20: Tenth ACM Conference on Data and Application Security and Privacy, New Orleans, LA, USA, 16–18 March 2020; Roussev, V., Thuraisingham, B.M., Carminati, B., Kantarcioglu, M., Eds.; 2020; pp. 97–108.
13. Shafahi, A.; Huang, W.R.; Najibi, M.; Suciu, O.; Studer, C.; Dumitras, T.; Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In Proceedings of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, Montréal, QC, Canada, 3–8 December 2018; Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; 2018; pp. 6106–6116. Available online: https://arxiv.org/abs/1804.00792 (accessed on 11 March 2021).
14. Saha, A.; Subramanya, A.; Pirsiavash, H. Hidden trigger backdoor attacks. In Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, 7–12 February 2020; AAAI Press: Palo Alto, CA, USA, 2020; pp. 11957–11965.
15. Dai, J.; Chen, C.; Li, Y. A backdoor attack against LSTM-based text classification systems. *IEEE Access* **2019**, *7*, 138872–138878. [CrossRef]
16. Yao, Y.; Li, H.; Zheng, H.; Zhao, B.Y. Latent backdoor attacks on deep neural networks. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, 11–15 November 2019; Cavallaro, L., Kinder, J., Wang, X., Katz, J., Eds.; 2019; pp. 2041–2055. Available online: https://dl.acm.org/doi/abs/10.1145/3319535.3354209 (accessed on 11 March 2021).
17. Shen, J.; Zhu, X.; Ma, D. TensorClog: An imperceptible poisoning attack on deep neural network applications. *IEEE Access* **2019**, *7*, 41498–41506. [CrossRef]
18. Zhu, C.; Huang, W.R.; Li, H.; Taylor, G.; Studer, C.; Goldstein, T. Transferable clean-label poisoning attacks on deep neural nets. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, CA, USA, 9–15 June 2019; Chaudhuri, K., Salakhutdinov, R., Eds.; 2019; Volume 97, pp. 7614–7623. Available online: http://proceedings.mlr.press/v97/zhu19a.html (accessed on 11 March 2021).
19. Liu, Y.; Ma, S.; Aafer, Y.; Lee, W.-C.; Zhai, J.; Wang, W.; Zhang, X. Trojaning attack on neural networks. In Proceedings of the 25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, CA, USA, 18–21 February 2018; The Internet Society: Reston, VA, USA, 2018.
20. Kwon, H.; Yoon, H.; Park, K.-W. Selective poisoning attack on deep neural networks. *Symmetry* **2019**, *11*, 892. [CrossRef]
21. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.J.; Fergus, R. Intriguing properties of neural networks. In Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, 14–16 April 2014; Conference Track Proceedings. Bengio, Y., LeCun, Y., Eds.; 2014. Available online: https://arxiv.org/abs/1312.6199 (accessed on 11 March 2021).
22. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015; Conference Track Proceedings. Bengio, Y., LeCun, Y., Eds.; 2015. Available online: https://arxiv.org/abs/1412.6572 (accessed on 11 March 2021).
23. Moosavi-Dezfooli, S.-M.; Fawzi, A.; Frossard, P. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016; IEEE Computer Society: Washington, DC, USA, 2016; pp. 2574–2582.
24. Moosavi-Dezfooli, S.-M.; Fawzi, A.; Fawzi, O.; Frossard, P. Universal adversarial perturbations. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; IEEE Computer Society: Washington, DC, USA, 2017; pp. 86–94.

25. Carlini, N.; Wagner, D.A. Towards evaluating the robustness of neural networks. In Proceedings of the 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, 22–26 May 2017; IEEE Computer Society: Washington, DC, USA, 2017; pp. 39–57.
26. Kurakin, A.; Goodfellow, I.J.; Bengio, S. Adversarial Examples in the Physical World. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017. Workshop Track Proceedings.
27. Su, J.; Vargas, D.V.; Sakurai, K. One pixel attack for fooling deep neural networks. *IEEE Trans. Evol. Comput.* **2019**, *23*, 828–841. [CrossRef]
28. Sarkar, S.; Bansal, A.; Mahbub, U.; Chellappa, R. UPSET and ANGRI: Breaking high performance image classifiers. *arXiv* **2017**, arXiv:1707.01159.
29. Baluja, S.; Fischer, I. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv* **2017**, arXiv:1703.09387.
30. Cissé, M.; Adi, Y.; Neverova, N.; Keshet, J. Houdini: Fooling deep structured prediction models. *arXiv* **2017**, arXiv:1707.05373.
31. Din, S.U.; Akhtar, N.; Younis, S.; Shafait, F.; Mansoor, A.; Shafique, M. Steganographic universal adversarial perturbations. *Pattern Recognit. Lett.* **2020**, *135*, 146–152. [CrossRef]
32. Brendel, W.; Rauber, J.; Bethge, M. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018. Conference Track Proceedings.
33. Papernot, N.; McDaniel, P.D.; Jha, S.; Fredrikson, M.; Celik, Z.B.; Swami, A. The limitations of deep learning in adversarial settings. In Proceedings of the IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, 21–24 March 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 372–387.
34. Kosiorek, A. Stacked Capsule Autoencoders. Available online: https://github.com/akosiorek/stacked_capsule_autoencoders (accessed on 24 September 2020).