SPATIALLY TRANSFORMED ADVERSARIAL EXAMPLES

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent studies show that widely used Deep neural networks (DNNs) are vulnerable to the carefully crafted adversarial examples. Many advanced algorithms have been proposed to generate adversarial examples by leveraging the $L_p$ distance for penalizing perturbations. Different defense methods have also been explored to defend against such adversarial attacks. While the effectiveness of $L_p$ distance as a metric of perceptual quality remains an active research area, in this paper we will instead focus on a different type of perturbation, namely spatial transformation, as opposed to manipulating the pixel values directly as in prior works. Perturbations generated through spatial transformation could result in large $L_p$ distance measures, but our extensive experiments show that such spatially transformed adversarial examples are more perceptually realistic and more difficult to defend against with existing defense systems. This potentially provides a new direction in adversarial example generation and the design of corresponding defenses. We visualize the spatial transformation based perturbation for different examples and show that our technique can produce realistic adversarial examples with smooth image deformation. Finally, we visualize the attention of deep networks with different types of adversarial examples to better understand how these examples are interpreted.

1 INTRODUCTION

Deep neural networks (DNNs) have demonstrated their outstanding performance in different domains, ranging from image processing (Krizhevsky et al., 2012; He et al., 2016), text analysis (Collobert & Weston, 2008) to speech recognition (Hinton et al., 2012). Though deep networks have exhibited high performance for these tasks, recently they have been shown to be particularly vulnerable to adversarial perturbations added to the input images (Szegedy et al., 2013; Goodfellow et al., 2015). These perturbed instances are called adversarial examples, which can lead to undesirable consequences in many practical applications based on DNNs. For example, adversarial examples can be used to subvert malware detection, fraud detection, or even potentially mislead autonomous navigation systems (Papernot et al., 2016b; Evtimov et al., 2017; Grosse et al., 2016) and therefore pose security risks when applied to security related applications. A comprehensive study about adversarial examples is required to motivate effective defenses. Different methods have been proposed to generate such adversarial examples. Fast gradient sign methods (FGSM) (Goodfellow et al., 2015) have been proposed to produce adversarial instances rapidly. Optimization based methods (Opt) have been applied to search for adversarial examples with smaller magnitude of perturbation (Carlini & Wagner, 2017a).

One important criterion for adversarial examples is that the perturbed images should “look like” the original instances. The traditional attack strategies adopt $L_2$ (or other $L_p$) norm distance as a perceptual similarity metric to evaluate the distortion (Gu & Rigazio, 2014). However, this is not an ideal metric (Johnson et al., 2016; Isola et al., 2017), as $L_2$ similarity is sensitive to lighting and viewpoint change of a pictured object. For instance, an image can be shifted by one pixel, which will lead to large $L_2$ distance, while the translated image can appear “the same” to human perception. Motivated by this example, in this paper we aim to look for other types of adversarial examples and propose to create more perceptually realistic examples by changing the positions of pixels instead of directly manipulating existing pixel values. This has been shown to better preserve the identity and structure of the benign image (Zhou et al., 2016b). Thus, the proposed spatially transformed adversarial example optimization method (stAdv) can keep adversarial examples less distinguishable from real instances (such examples can be found in Figure 3).
Various defense methods have also been proposed to defend against adversarial examples. Adversarial training based methods have so far achieved the most promising results (Goodfellow et al., 2015; Tramèr et al., 2017; Madry et al., 2017). They have demonstrated the robustness of improved deep networks under certain constraints. However, the spatially transformed adversarial examples are generated through a rather different principle, whereby what is being minimized is the local distortion rather than the $L_p$ pixel error between the adversarial and original instances. Thus, the previous adversarial training based defense method may appear less effective against this new attack given the fact that these examples generated by stAdv have never been seen before. This opens a new challenge about how to defend against such attacks, as well as other attacks that are not based on direct pixel value manipulation.

We provide visualization of the spatial deformation generated by stAdv; it is seen to be very smooth and virtually imperceptible to the human eye. In addition, to better understand the properties of deep neural networks on different adversarial examples, we provide visualization of the attention of the DNN given adversarial examples generated by different attack algorithms. We find that the spatial transformation based attack is more resilient across different defense models, including adversarially trained robust models.

Our contributions are summarized as follows:

- We propose to generate adversarial examples based on spatial transformation instead of direct manipulation of the pixel values, and we show realistic and effective adversarial examples on MNIST, CIFAR10, and ImageNet datasets.
- We provide the visualization of optimized transformation and show that such geometric changes are small and locally smooth, leading to high perceptual quality.
- We empirically show that, compared to other attacks, adversarial examples generated by stAdv are more difficult to detect with current defense methods.
- Finally, we also provide the visualization for the attention of deep networks on different adversarial examples and demonstrate that adversarial examples based on stAdv can more consistently mislead the adversarial trained robust deep networks compared to other existing attack methods.

2 RELATED WORK

Here we first briefly summarize the existing adversarial algorithms as well as the current defense methods. We then discuss the spatial transformation used in our adversarial attacks.

**Adversarial Examples** Given a benign sample $x$, an attack instance $x_{adv}$ is referred to as an adversarial example, if a small magnitude of perturbation $\epsilon$ is added to $x$ (i.e. $x_{adv} = x + \epsilon$) so that $x_{adv}$ is misclassified by the targeted classifier $g$. Based on the adversarial goal, attacks can be classified into two categories, targeted and untargeted attacks. In a targeted attack, the adversary’s objective is to modify an input $x$ such that the target model $g$ classifies the perturbed input $x_{adv}$ in a targeted class chosen, which differs from its ground truth. In an untargeted attack, the adversary’s objective is to cause the perturbed input $x_{adv}$ to be misclassified in any class other than its ground truth. Based on the adversarial capabilities, these attacks can be categorized as white-box and black-box attacks, where an adversary has full knowledge of the classifier and training data in the white-box setting (Szegedy et al., 2014; Goodfellow et al., 2015; Carlini & Wagner, 2017a; Moosavi-Dezfooli et al., 2015; Papernot et al., 2016b; Biggio et al., 2013; Kurakin et al., 2016); while having zero knowledge about them in the black-box setting (Papernot et al., 2016a; Liu et al., 2017; Moosavi-Dezfooli et al., 2016; Mopuri et al., 2017). In this work, we will focus on the white-box setting to explore what a powerful adversary can do based on the Kerckhoffs’s principle (Shannon, 1949) to better motivate defense methods.

**Spatial Transformation** In computer vision and graphics literature, two main aspects determine the appearance of a pictured object (Szeliski, 2010): (1) the lighting and material, which determine the brightness of a point as a function of illumination and object material properties, and (2) the geometry, which determines where the projection of a point will be located in the scene. Most previous adversarial attacks (Goodfellow et al., 2015) build on changing the lighting and material
aspect, while assuming the underlying geometry stays the same during the adversarial perturbation generation process.

Modeling geometric transformation with neural networks was first explored by “capsules,” computational units that locally transform their input for modeling 2D and 3D geometric changes (Hinton et al., 2011). Later, Jaderberg et al. (2015) demonstrated that similar computational units, named spatial transformers, can benefit various recognition tasks. Zhou et al. (2016a) adopted the spatial transformers for synthesizing novel views of the same object and has shown that a geometric method can produce more realistic results compared to pure pixel-based method. Inspired by these works, we also use the spatial transformers to deform the input images, but with the different goal of generating realistic adversarial examples.

### Defensive Methods
Following the emergence adversarial examples, various of defense methods have been studied, including adversarial training (Goodfellow et al., 2015), distillation (Papernot et al., 2016c), gradient masking (Gu & Rigazio, 2014) and feature squeezing (Xu et al., 2017). However, it has been shown that these defenses can either be evaded by Opt attacks or only provide marginal improvements (Carlini & Wagner, 2017b; He et al., 2017). Among these defenses, adversarial training has achieved the state-of-the-art performance. Goodfellow et al. (2015) proposed to use the fast gradient sign attack as an adversary to perform adversarial training, which is much faster, followed by ensemble adversarial training (Tramèr et al., 2017) and projected gradient descent (PGD) adversarial training (Madry et al., 2017). In this work, we explicitly analyze how effective the spatial transformation based adversarial examples are under these adversarial training based defense methods.

## 3 Generating Adversarial Examples

Here we first introduce several existing attack methods and then present our formulation for producing spatially transformed adversarial examples.

### 3.1 Problem Definition

Given a learned classifier \( g: \mathcal{X} \rightarrow \mathcal{Y} \) from the feature space \( \mathcal{X} \) to a set of classification outputs \( \mathcal{Y} \) (e.g., \( \mathcal{Y} = \{0, 1\} \) for binary classification), an adversary aims to generate adversarial example \( x_{\text{adv}} \) for an original instance \( x \in \mathcal{X} \) with its ground truth label \( y \in \mathcal{Y} \), so that the classifier predicts \( g(x_{\text{adv}}) \neq y \) (untargeted attack) or \( g(x_{\text{adv}}) = t \) (targeted attack) where \( t \) is the target class.

### 3.2 Background: Current Pixel-Value Based Attack Methods

There have been a number of methods for generating adversarial examples, all built on directly modifying the pixel values of the original image.

The fast gradient sign method (FGSM) (Goodfellow et al., 2015) uses a first-order approximation of the loss function in order to construct adversarial samples for the adversary’s target classifier \( g \). The algorithm achieves untargeted attack by performing a single gradient ascent step: \( x_{\text{adv}} = x + \epsilon \cdot \text{sign}(\nabla_x \ell_g(x, y)) \), where \( \ell_g(x, y) \) is the loss function (e.g. cross-entropy loss) used to train the original model \( g \), \( y \) is the ground truth label and the hyper-parameter \( \epsilon \) controls the magnitude of the perturbation. A targeted version of it can be done similarly.

Optimization based attack (Opt) produces an adversarial perturbation for a targeted attack based on certain constraints (Carlini & Wagner, 2017a; Liu et al., 2017) as formulated below:

\[
\min ||\delta||^2_p \quad \text{s.t.} \quad g(x + \delta) = t \quad \text{and} \quad x + \delta \in X,
\]

where the \( L_p \) norm penalty ensures that the added perturbation \( \epsilon \) is small. The same optimization procedure can achieve untargeted attacks with a modified constraint \( g(x + \delta) \neq y \).

### 3.3 Our Approach: Spatially Transformed Adversarial Examples

All the existing approaches directly modify pixel values, which may produce noticeable artifacts. Instead, we aim to smoothly change the geometry of the scene while keeping the original appearance,
producing more perceptually realistic adversarial examples. In this section, we introduce our geometric image formation model and then describe the objective for generating spatially transformed adversarial examples.

**Spatial transformation** We use $x_{adv}^{(i)}$ to denote the value of the $i$-th pixel and 2D coordinates $(u_{adv}^{(i)}, v_{adv}^{(i)})$ to denote its location in the adversarial image $x_{adv}$. We assume that $x_{adv}^{(i)}$ is transformed from the pixel $x^{(i)}$ from the original image. We use the per-pixel flow (displacement) field $f$ to synthesize the adversarial image $x_{adv}$ using pixels from the input $x$. For the $i$-th pixel within $x_{adv}$ at the pixel location $(u^{(i)}, v^{(i)})$, we optimize the amount of displacement in each image dimension, with the pair denoted by the flow vector $f_i := (\Delta u^{(i)}, \Delta v^{(i)})$. Note that the flow vector $f_i$ goes from a pixel $x_{adv}^{(i)}$ in the adversarial image to its corresponding pixel $x^{(i)}$ in the input image. Thus, the location of its corresponding pixel $x^{(i)}$ can be derived as $(u^{(i)}, v^{(i)}) = (u_{adv}^{(i)} + \Delta u^{(i)}, v_{adv}^{(i)} + \Delta v^{(i)})$. As the $(u^{(i)}, v^{(i)})$ can be fractional numbers and does not necessarily lie on the integer image grid, we use the differentiable bilinear interpolation [Jaderberg et al. 2015] to transform the input image with the flow field. We calculate $x_{adv}^{(i)}$ as:

$$x_{adv}^{(i)} = \sum_{q \in \mathcal{N}(u^{(i)}, v^{(i)})} x^{(q)} (1 - |u^{(i)} - u^{(q)})|) (1 - |v^{(i)} - v^{(q)})|), \quad (1)$$

where $\mathcal{N}(u^{(i)}, v^{(i)})$ are the indices of the 4-pixel neighbors at the location $(u^{(i)}, v^{(i)})$ (top-left, top-right, bottom-left, bottom-right). We can obtain the adversarial image $x_{adv}$ by calculating Equation 1 for every pixel $x_{adv}^{(i)}$. Note that $x_{adv}$ is differentiable with respect to the flow field $f$ [Jaderberg et al. 2015; Zhou et al. 2016b]. The estimated flow field essentially captures the amount of spatial transformation required to fool the classifier.

**Objective function** Most of the previous methods constrain the added perturbation to be small regarding a $L_p$ metric. Here instead of imposing the $L_p$ norm, we introduce a new regularization loss $L_{flow}$ on the local distortion $f$, producing higher perceptual quality for adversarial examples. Therefore, the goal of the attack is to generate adversarial examples which can mislead the classifier as well as minimizing the local distortion introduced by the flow field $f$.

Formally, given a benign instance $x$, we obtain the flow field $f$ by minimize the following objective:

$$f^* = \arg\min_f \mathcal{L}_{adv}(x, f) + \tau \mathcal{L}_{flow}(f), \quad (2)$$

where $\mathcal{L}_{adv}$ encourages the generated adversarial examples to be misclassified by the target classifier. $L_{flow}$ ensures that the spatial transformation distance is minimized to preserve high perceptual
quality and \( \tau \) balances the adversarial loss and the spatial transformation loss. The goal of \( L_{\text{adv}} \) is to guarantee the targeted attack \( g(x_{\text{adv}}) = t \) where \( t \) is the targeted class, different from the ground truth label \( y \). Recall that we transform the input image \( x \) to \( x_{\text{adv}} \) with the flow field \( f \) (Equation 1).

In practice, directly enforcing \( g(x_{\text{adv}}) = t \) during optimization is highly non-linear, we adopt the objective function suggested in Carlini & Wagner (2017c).

\[
L_{\text{adv}}(x, f) = \max_{i \neq t} \left( g(x_{\text{adv}}) - g(x_{\text{adv}})_i, 0 \right),
\]

where \( g(x) \) represents the logit output of model \( g \), and \( g(x)_i \) denotes the \( i \)th element of the logit vector.

To compute \( L_{\text{flow}} \), we calculate the sum of spatial movement distance for any two adjacent pixels. Given an arbitrary pixel \( p \) and its neighbors \( q \in \mathcal{N}(p) \), we enforce the locally smooth spatial transformation perturbation \( L_{\text{flow}} \) based on the total variation (Rudin et al., 1992):

\[
L_{\text{flow}}(f) = \sum_{p} \sum_{q \in \mathcal{N}(p)} \sqrt{||\Delta u^{(p)} - \Delta u^{(q)}||^2_2 + ||\Delta v^{(p)} - \Delta v^{(q)}||^2_2}.
\]

Intuitively, minimizing the spatial transformation can help ensure the high perceptual quality for stAdv, since adjacent pixels tend to move towards close direction and distance. We solve the above optimization with L-BFGS solver (Liu & Nocedal, 1989).

4 Experimental Results

In this section, we first show adversarial examples generated by the proposed spatial transformation method and analyze the properties of these examples from different perspectives. We then visualize the estimated flows for adversarial examples and show that with small and smooth transformation, the generated adversarial examples can already achieve a high attack success rate against deep networks. We also show that stAdv can preserve a high attack success rate against current defense methods, which motivates more sophisticated defense methods in the future. Finally, we analyze the attention regions of DNNs, to better understand the attack properties of stAdv.

Experiment Setup

We set \( \tau \) as 0.05 for all our experiments. We leverage L-BFGS (Liu & Nocedal, 1989) as our solver with backtracking linear search.

4.1 Adversarial Examples Based on Spatial Transformations

We show adversarial examples with high perceptual quality for both MNIST (LeCun & Cortes, 1998) and CIFAR-10 (Krizhevsky et al., 2014) datasets.

stAdv on MNIST

In our experiments, we leverage three target models whose network architectures are shown in Table A to generate adversarial examples in the white-box setting on the MNIST dataset. Models A, B and C are derived from Tramer et al. (2017), which represent different architectures. See Appendix A for more details. Table 1 presents the accuracy of pristine MNIST test data on each model as well as the attack success rate of adversarial examples generated by stAdv on these models. Figure 2 shows the adversarial examples against different models where the original instances appear in the diagonal. Each adversarial example achieves a targeted attack, with the target class shown on the top of the column. It is clear that the generated adversarial examples still appear to be in the same class as the original instance for humans. Another advantage for stAdv compared with traditional attacks is that examples based on stAdv seldom show noise pattern within the adversarial examples. Instead, stAdv smoothly deforms the digits and since such natural deformation also exists in the dataset digits, humans can barely notice such manipulation.

stAdv on CIFAR-10

For CIFAR-10, we use models ResNet-32\(^1\) and wide ResNet-34\(^2\) as the target classifiers (Zagoruyko & Komodakis, 2016; He et al., 2016; Madry et al., 2017). We show

\(^1\)https://github.com/tensorflow/models/blob/master/research/ResNet/ResNet_model.py
\(^2\)https://github.com/MadryLab/cifar10_challenge/blob/master/model.py
Table 1: Accuracy of pristine data (p) on different models, and attack success rate of adversarial examples generated by stAdv on MNIST dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (p)</td>
<td>98.58%</td>
<td>98.94%</td>
<td>99.11%</td>
</tr>
<tr>
<td>Attack Success Rate</td>
<td>99.95%</td>
<td>99.98%</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

Table 2: Performance of different models on pristine (p) and stAdv adversarial examples on CIFAR-10 dataset. The number in parentheses is the number of parameters.

<table>
<thead>
<tr>
<th>Model</th>
<th>ResNet32 (0.47M)</th>
<th>Wide ResNet34 (46.16M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (p)</td>
<td>93.16%</td>
<td>95.82%</td>
</tr>
<tr>
<td>Attack Success Rate</td>
<td>99.56%</td>
<td>98.84%</td>
</tr>
</tbody>
</table>

Comparison of different adversarial examples: In Figure 4, we show adversarial examples that are targeted attacked to the same class (“0” for MNIST and “airplane” for CIFAR-10), which is different from their ground truth. We compare adversarial examples generated from different methods and show that those based on stAdv clearly preserve higher perceptual quality compared with FGSM and Opt methods.

4.2 Visualizing Spatial Transformation

In order to better understand the spatial transformation applied to the original images, we visualize the optimized transformation flow for different datasets, respectively. Figure 5 visualizes a transformation on an MNIST instance, where the digit “0” is misclassified as “2.” We can see that the adjacent flows move in a similar direction in order to generate smooth figure. The flows are more focused on the edge of the digit and sometimes these flows move in different directions along the
edge, which implies that the object boundary plays an important role in our StAdv optimization. Figure 6 illustrates a similar visualization on CIFAR-10. It shows that the optimized flows often focus on the area of the main object, such as the airplane. We also observe that the magnitude of flows near the edge are usually larger, which similarly indicates the importance of edges for misleading the classifiers. This observation confirms the observation that when DNNs extract edge information in the earlier layers for visual recognition tasks (Viterbi, 1998). In addition, we visualize the similar flow for ImageNet (Krizhevsky et al., 2012) in Figure 7. The top-1 label of the original image in Figure 7(a) is “mountain bike”. Figure 7(b)-(d) show targeted adversarial examples generated by StAdv, which have target classes “goldfish,” “maltese dog,” and “tabby cat,” respectively, and which

![Flow visualization on MNIST. The digit “0” is misclassified as “2”.](image)

![Figure 4: Comparison for adversarial examples generated by FGSM, Opt and StAdv. (Left: MNIST, right: CIFAR-10) The target class for MNIST is “0” and “air plane” for cifar.](image)

![Figure 3: Adversarial examples generated by StAdv against different models on CIFAR-10. The ground truth images are shown in the diagonal while the adversarial examples on each column are classified into the same class as the ground truth image within that column.](image)
are predicted as such as the top-1 class. An interesting observation is that, although there are other objects within the image, nearly 90% of the spatial transformation flows tend to focus on the target object bike. Different target class correspond to different directions for these flows, which still fall into the similar area.

Figure 7: Flow visualization on ImageNet. (a): the original image, (b)-(c): images are misclassified into goldfish, dog and cat, respectively. Note that to display the flows more clearly, we fade out the color of the original image.

4.3 ATTACK EFFICIENCY UNDER DEFENSE METHODS

Here we generate adversarial examples in the white-box setting and test defense methods against these samples to evaluate the strength of these attacks under defense. To evaluate the efficiency of defenses, we directly apply different defense methods on these generated adversarial examples to make predictions. We mainly focus on the adversarial training defenses, since they have demonstrated the state-of-the-art performance. We apply three defense strategies in our evaluation: the FGSM adversarial training (Adv.) (Goodfellow et al., 2015), ensemble adversarial training (Ens.) (Tramèr et al., 2017), and projectile gradient descent (PGD) adversarial training (Madry et al., 2017) methods. For adversarial training purpose, we generate adversarial examples with 0.3 as the $L_\infty$ bound (Carlini & Wagner, 2017a). We test adversarial examples generated against model A, B, and C on MNIST as shown in Table 4, and similarly adversarial examples generated against ResNet32 and wide ResNet34 on CIFAR-10.

The results on MNIST and CIFAR-10 dataset are shown in Table 5. We observe that the three defense strategies can achieve high performance (less than 10% attack success rate) against FGSM and Opt attacks. Note that adversarial examples generated by FGSM and Opt apply 0.3 as the $L_\infty$ bound. For simplicity, we use confidence $\kappa = 0$ for both Opt and stAdv for a fair comparison. These defense methods only achieve low defense performance on stAdv, which improve the attack success rate to more than 30% among all defense strategies. These results indicate that new type of adversarial strategy, such as our spatial transformation-based attack, may open new directions for developing better defence systems. However, for stAdv, we cannot use $L_p$ norm to bound the distance as translating a image by one pixel may introduce large $L_p$ penalty. We instead constrain the spatial transformation flow and show that our adversarial examples have high perceptual quality in Figures 2, 3, and 4.
Table 3: Attack success rate of adversarial examples generated by stAdv against models A, B, and C under standard defenses on MNIST, and against ResNet and wide ResNet on CIFAR-10.

<table>
<thead>
<tr>
<th>Model</th>
<th>Def.</th>
<th>FGSM</th>
<th>Opt.</th>
<th>stAdv</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Adv.</td>
<td>4.3%</td>
<td>4.6%</td>
<td>32.62%</td>
</tr>
<tr>
<td></td>
<td>Ens.</td>
<td>1.6%</td>
<td>4.2%</td>
<td>48.07%</td>
</tr>
<tr>
<td></td>
<td>PGD</td>
<td>4.4%</td>
<td>2.96%</td>
<td>48.38%</td>
</tr>
<tr>
<td>B</td>
<td>Adv.</td>
<td>6.0%</td>
<td>4.5%</td>
<td>50.17%</td>
</tr>
<tr>
<td></td>
<td>Ens.</td>
<td>2.7%</td>
<td>3.18%</td>
<td>46.14%</td>
</tr>
<tr>
<td></td>
<td>PGD</td>
<td>9.0%</td>
<td>3.0%</td>
<td>49.82%</td>
</tr>
<tr>
<td>C</td>
<td>Adv.</td>
<td>3.22%</td>
<td>0.86%</td>
<td>30.44%</td>
</tr>
<tr>
<td></td>
<td>Ens.</td>
<td>1.45%</td>
<td>0.98%</td>
<td>28.82%</td>
</tr>
<tr>
<td></td>
<td>PGD</td>
<td>2.1%</td>
<td>0.98%</td>
<td>28.13%</td>
</tr>
</tbody>
</table>

4.4 Visualizing Attention of Networks on Adversarial Examples

In addition to the analyzing adversarial examples themselves, in this section, we further characterize these spatially transformed adversarial examples from the perspective of deep neural networks.

Here we apply Class Activation Mapping (CAM) (Zhou et al., 2016a), an implicit attention visualization technique for localizing the discriminative regions detected by a DNN. We use it to show the attention of the target ImageNet inception_v3 model (Szegedy et al., 2016) for both the original image and generated adversarial examples. Figure 8(a) shows an input bike image and Figure 8(b)–(d) show the targeted adversarial examples based on stAdv targeting three different classes (goldfish, dog, and cat). Figure 8(e) illustrates that the target model draws attention to the bicycle region. Interestingly, attention regions on examples generated by stAdv varies for different target classes as shown in Figure 8(f)–(h). Though humans can barely distinguish between the original image and the ones generated by stAdv, CAM map focus on completely different regions, implying that our attack is able to mislead the network’s attention.

In addition, we also compare and visualize the attention regions of both naturally trained and the adversarial trained inception_v3 model (Figure 9). The ground truth top-1 label is “cinema,” so the attention region for the original image (Figure 9(a)) includes both tower and building regions. However, when the adversarial examples are targeted attacked into the adversarial label “missile,” the attention region focuses on only the tower for all the attack algorithms as shown in Figure 9(b)–(d) with slight different attention region sizes. More interestingly, we also test these adversarial examples on the public adversarial trained robust inception_v3 model. The result shows in Figure 9(f)–(h). This time, the attention region is drawn to the building again for both FGSM and Opt methods, which is close to the attention region of the original image. The top-1 label for Figure 9(f) and (g) are again the ground truth “cinema”, which means they fail to attack the robust model. However, Figure 9(h) is still misclassified as “missile” under the robust model and the CAM visualization shows that the attention region still focuses on the tower. This example again implies that adversarial examples generated by stAdv are challenging to defend for the current “robust” ImageNet models.

5 Conclusions

Different from the previous works that generate adversarial examples by directly manipulating pixel values, in this work we propose a new type of perturbation based on spatial transformation, which aims to preserve high perceptual quality for adversarial examples. We have shown that adversarial examples generated by stAdv are much more difficult for humans to distinguish from original instances. We also analyze the attack success rate of these examples under existing defense methods and demonstrate they are harder to defend against, which opens new directions for developing more robust defense algorithms. Finally, we visualize the attention regions of DNNs on our adversarial examples to better understand this new attack.

3https://github.com/tensorflow/cleverhans/tree/master/examples/nips17_adversarial_competition/
Figure 8: CAM attention visualization for ImageNet inception_v3 model. (a) the original image and (b)-(d) are stAdv adversarial examples targeting different classes. Row 2 show the attention visualization for the corresponding images above.

Figure 9: CAM attention visualization for ImageNet inception_v3 model. Column 1 shows the CAM map corresponding to the original image. Column 2-4 show the adversarial examples generated by different methods. The visualization is drawn for Row 1: inception_v3 model; Row 2: (robust) adversarial trained inception_v3 model. (a) and (e)-(g) are labeled as the ground truth “cinema”, while (b)-(d) and (h) are labeled as the adversarial target “missile”.

REFERENCES


## A Model Architectures

Table 4: Architecture of models applied on MNIST

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv(64,5,5) + Relu</td>
<td>Conv(128,3,3) + Relu</td>
<td>Conv(128,3,3) + Relu</td>
<td></td>
</tr>
<tr>
<td>Conv(64,5,5) + Relu</td>
<td>Dropout(0.25)</td>
<td>Dropout(0.25)</td>
<td></td>
</tr>
<tr>
<td>Conv(128, 6, 6) + Relu</td>
<td>Conv(128, 5, 5) + Relu</td>
<td>Dropout(0.5)</td>
<td></td>
</tr>
<tr>
<td>Conv(128, 6, 6) + Relu</td>
<td>Dropout(0.5)</td>
<td>Dropout(0.5)</td>
<td></td>
</tr>
<tr>
<td>FC(128) + Relu</td>
<td>FC(128) + Relu</td>
<td>FC(128) + Relu</td>
<td></td>
</tr>
<tr>
<td>Dropout(0.5)</td>
<td>Dropout(0.5)</td>
<td>Dropout(0.5)</td>
<td></td>
</tr>
<tr>
<td>FC(10) + Softmax</td>
<td>FC(10) + Softmax</td>
<td>FC(10) + Softmax</td>
<td></td>
</tr>
<tr>
<td>FC(10) + Softmax</td>
<td>FC(10) + Softmax</td>
<td>FC(10) + Softmax</td>
<td></td>
</tr>
</tbody>
</table>