Reproduction of Spatially Transformed Adversarial Examples

A Blog Post from Group 68

Authors: Emiel den Haan, Bryan He, Annabel Simons & Mike Wu

Table of Contents

1. Introduction

1.1. The Need to Reproduce Papers

According to Hutson [1], there is an artificial intelligence (AI) reproducibility crisis. However, Raff [2] states that the reproduction rates of papers in this field have not changed over the past decades. Raff [2] successfully replicated 162 out of 255 papers attempted in the field of AI/machine learning (ML)/deep learning (DL). Still, this means that almost 40% of their included papers were not reproducible. Reproduction of scientific papers is essential for several reasons:

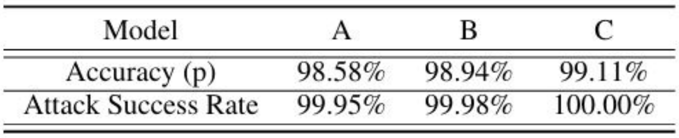
- Verification of Results: Reproducing the results of a scientific paper helps verify the findings of that paper. It ensures that the results were not due to chance, errors, or biases.
- Scientific Progress: Reproducibility is an important part of the scientific process. It helps to build a body of knowledge that is reliable. Each reproduction of a paper adds to the evidence base and helps science move forward.
- Understanding the Methodology: Reproducing a paper can help to gain a more in-depth understanding of the methods used. It can help identify any potential weaknesses or areas for improvement in the original paper.

This blog post will, therefore, contribute by independently reproducing the deep learning paper: ‘Spatially Transformed Adversarial Examples’ (stAdv) from Xiao et al. [3]. With an independent reproduction, we mean that we reproduced the paper from scratch, including the code. Figure 1 shows the results from Xiao et al. [3], which are the primary targets for our reproduction. We will mainly try to verify their first claim: "we show realistic and **effective** adversarial examples on MNIST" [3]. The word "effective" is important because it indicates that their results show that it works well quantitatively; see Figure 1, which we wanted to test.

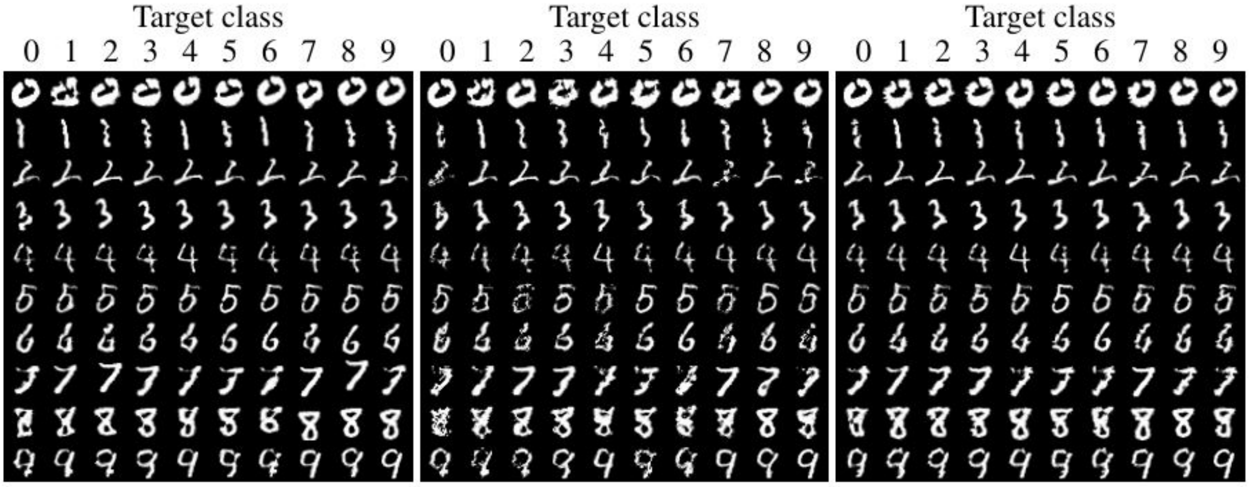
1.2. Adversarial Examples

The paper from Xiao et al. [3] presents a new way of generating adversarial examples. With the growing interest in deep neural networks (DNNs) because of how well they perform in ML tasks processing images or analyzing text, for example, [3, 4], Xiao et al. [3] state in their introduction that these DNNs "are particularly vulnerable to adversarial perturbations added to the input images," which are the adversarial examples. These examples are therefore called attacks, and defences for these attacks are to use "adversarial training based methods" [3]. The new variant presented by Xiao et al. [3] is based on spatially transforming the adversarial examples, such as distorting the image or rotating it slightly. This is new because, before their paper, the pixel values were usually changed. They claim it is "realistic and effective", which other methods lack according to them [3]. Here, "effective" refers to the quantified attack success rates on three different CNN classification models, all of which are above 99%, see row Figure 1. Additionally, they state that defence mechanisms are less robust against these spatially transformed adversarial examples, and it is harder to detect their attack by humans [3]. Moreover, "realistic" refers to how their human perceptibility study suggests that humans have trouble telling the adversarial examples apart from the unperturbed versions.

In 2020, Xu et al. [4] published a review in which they looked at several adversarial attack methods and their defences. The paper from Xiao et al. [3] was also included in their analysis. In Xu et al. [4] review, the spatially transformed adversarial examples are labelled as so-called "white-box attacks". The attacker has the model and the victim samples in contrast to "black-box attacks" or "grey-box attacks", which are settings where the models are (partially) not available for the attacker [4]. In the rest of this paragraph, we will focus on the "white-box attacks". Firstly, they describe several "traditional adversarial attacks" [4]. As stated earlier, these directly change the values of pixels in the victim image, which often results in adversarial examples which humans can easily detect. One of these attacks is the fast gradient sign method (FGSM) from Goodfellow et al. [5]. FGSM can be used for a targeted attack and is relatively fast compared to other attacks. FGSM "searches the point which has the minimum loss value to label t", where t is the target. By doing this, it finds the area with the biggest influence on predicting the target [4, 5]. Xu et al. [4] mention the attack from Xiao et al. [3] as one of the last 'white-box attacks". They also mention the advantage that it is not easy for humans to detect the attack [4].



Model	A	B	C
Accuracy (p)	98.58%	98.94%	99.11%
Attack Success Rate	99.95%	99.98%	100.00%



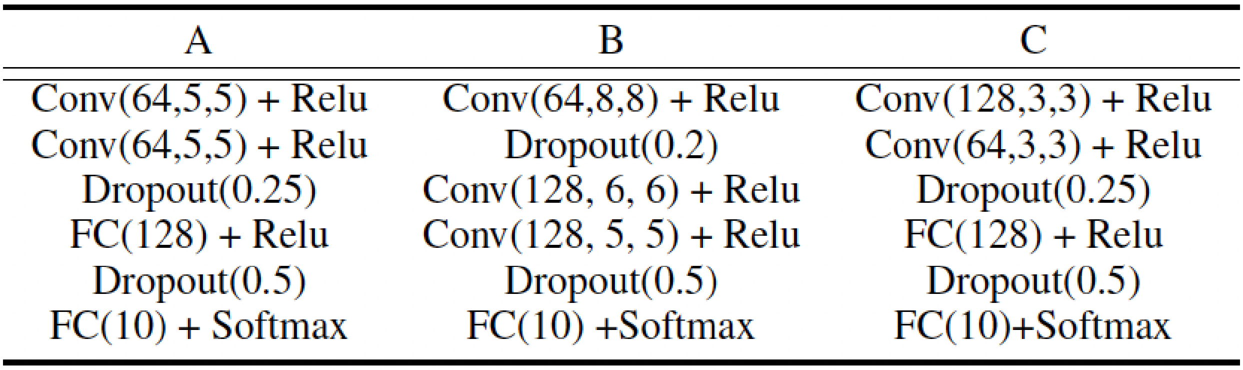
(a) Model A (b) Model B (c) Model C

Figure 1: Reproducibility goals (Figure & Table adapted from Xiao et al. [3]).

2. Implementation

2.1. Implementation of Models A, B & C

In order to reproduce the results from Xiao et al. [3], as seen in Figure 1, we had to reproduce the convolutional neural networks (CNN) A, B and C. Xiao et al. [3] described in their paper how the architecture of the three models was implemented (see Figure 2), and mention the use of the well-known MNIST dataset in their paper. All three CNNs were implemented using PyTorch version 2.2.0. In the original paper by Xiao et al. [3], the specific loss function, optimizer and learning rate used are not mentioned. The amount of epochs used for training the models was also not mentioned. Therefore, we chose to use the SGD optimizer with a learning rate of 0.01. Additionally, we used a learning rate decay; the learning rate was decreased to 0.001 after 5 epochs. In total, the models were trained for 10 epochs. Luckily, the absence of such information did not hinder our reproduction of the three models, as the accuracy of our models is similar to the accuracy reported in Table 1 of [3] (see the Results section for more information).



A	B	C
Conv(64,5,5) + Relu	Conv(64,8,8) + Relu	Conv(128,3,3) + Relu
Conv(64,5,5) + Relu	Dropout(0.2)	Conv(64,3,3) + Relu
Dropout(0.25)	Conv(128, 6, 6) + Relu	Dropout(0.25)
FC(128) + Relu	Conv(128, 5, 5) + Relu	FC(128) + Relu
Dropout(0.5)	Dropout(0.5)	Dropout(0.5)
FC(10) + Softmax	FC(10) +Softmax	FC(10)+Softmax

Figure 2: The architecture of models A, B & C as seen in Xiao et al. [3] (see Appendix A of their paper).

2.2. Implementation of the Adversarial Examples

To create a spatially transformed adversarial example, one needs an (unperturbed) image and a flow field, see Figure 3 for an example. The flow field has the same height and width as the image and assigns a flow vector $f_i = (\Delta u, \Delta v)$ to each pixel in the image. The i-th pixel in the adversarial image xadv has coordinates denoted by $(u_{adv}^{(i)}, v_{adv}^{(i)})$ and its pixel value denoted by $x_{adv}^{(i)}$. The value of $x_{adv}^{(i)} = x^{(q)}$, where $x^{(q)}$ is the pixel value of the q-th pixel of the (unperturbed) original image with the coordinates $(u_{adv}^{(i)} + \Delta u, v_{adv}^{(i)} + \Delta v)$. Because the flow vector f_i can contain fractions, bilinear interpolation (see Equation (1) below) is used to calculate the pixel value by interpolating the 4 neighbours of the pixel q.

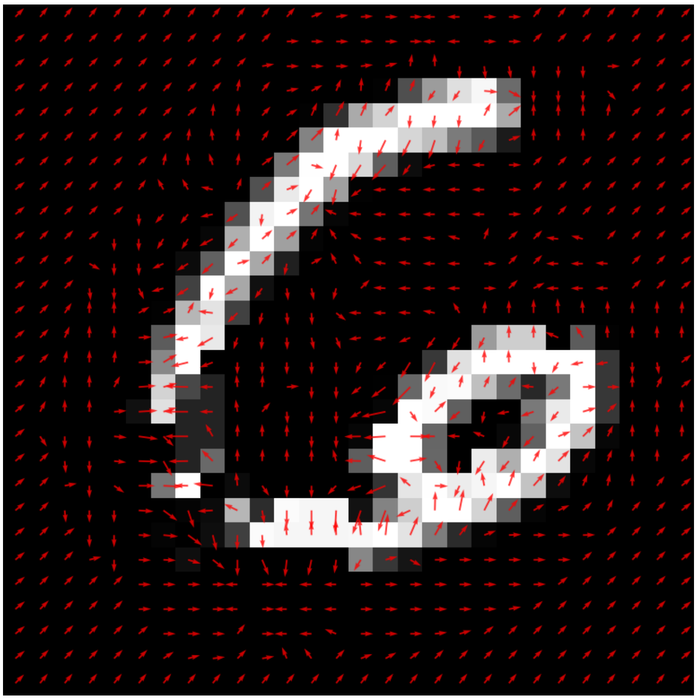


Figure 3: Red arrows are the flows for each pixel and the gray scale image is the (unperturbed) image.

For creating the adversarial examples the description and formulas from the original paper were used. Equation 1 is the formula for bilinear interpolation; equation 2 is the loss function optimized by the optimizer in order to find the optimized flowfield f^* which is utilized to obtain the optimal adversarial version of the original image; equations 3 and 4 are the constituents of equation 2 which the optimizer aims to minimize in order to fool the classification models A, B and C. Equations 1, 2, 3 and 4 were implemented from scratch as methods, see our implementation here (https://github.com/bryanheee/DL_stAdv).

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

$$f^* = \underset{f}{\operatorname{argmin}}\ \mathcal{L}_{\text{adv}}(x, f) + \tau \mathcal{L}_{\text{flow}}(f) \tag{2}$$

$$\mathcal{L}_{\text{adv}}(x, f) = \max\left(\left(\max_{i \neq t} g(x_{\text{adv}})_i\right) - g(x_{\text{adv}})_t, \kappa\right) \tag{3}$$

$$\mathcal{L}_{\text{flow}}(f) = \sum_{p}^{\text{all pixels}} \sum_{q \in \mathcal{N}(p)} \sqrt{\|\Delta u(p) - \Delta u(q)\|_2^2 + \|\Delta v(p) - \Delta v(q)\|_2^2} \tag{4}$$

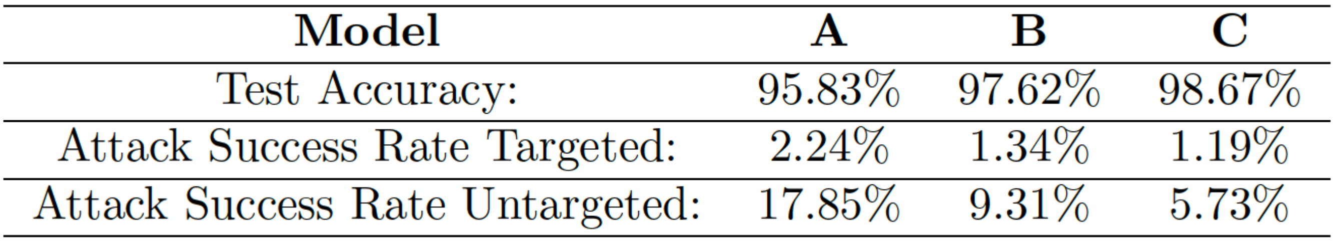
The implementation was done using the previously mentioned PyTorch version. The implementation details were not always explained in great depth by the authors of the paper. Therefore, we made some choices ourselves which may differ from their implementation. These choices are described in this paragraph. The initialization of the flowfield that needs to be optimized, and the hyperparameters used in the LBFGS algorithm were not scrupulously specified by Xiao et al. [3]. Therefore, we tried out different values for the hyperparameters, which are the number of steps, number of iterations and learning rate for the optimizer; the exact values used in our reproduction can be found in our Jupyter Notebook here (https://github.com/bryanheee/DL_stAdv). In addition, several different initial flowfields were used to determine if this affected the final result: zeroes, random and Gaussian with zero mean and 0.5 standard deviation.

Xiao et al. [3] did not mention their method for creating the test set; only the method for creating the adversarial examples is mentioned. An exhaustive test set would consist of 90,000 adversarial examples as each image in the test set consisting of 10,000 images would have 9 targets. To create the test set for determining the adversarial attack success rate, a random target number, which is different from the ground truth, was selected for each sample due to constraints in computation, ultimately resulting in a test set consisting of 10,000 adversarial samples. This design choice might be one of the reasons why our attack success rate differs from the rates reported by Xiao et al. [3]. To determine the success rate, the number of successful miss-classifications was used.

3. Results

The training of the A, B and C models on the MNIST dataset yielded similar results to those mentioned in the paper of Xiao et al. [3] and can be seen in Table 1. All the models have a test The results for the adversarial examples differ quite a bit from the paper by Xiao et al. [3]. As shown in Table 1, our results for the attack success rate for model A are much lower targeted or untargeted compared to the results from Xiao et al. [3] visible in Figure 1. The difference in percentages might be a result of the information missing in the paper of Xiao et al. [3], which we had to fill in ourselves. Our 17,85% untargeted attack success rate for model A is somewhat similar to another reproduction on GitHub (https://github.com/as791/stAdv-PyTorch/blob/main/StAdv_attack.ipynb), which was able to get an attack success rate of 38.0%. The reproduction on GitHub (https://github.com/as791/stAdv-PyTorch/blob/main/StAdv_attack.ipynb) and our reproduction are both not near the 99% of [3]. Model B and C are even lower than model A for us. We assume this is the case, because we used model A while experimenting with different flow field initializations. A more detailed analysis of why the attack success rate was not reproducible for us can be found in the Conclusion.

Next to the reproduction of the table and the percentages, we also reproduced the visualization from Figure 1. Our reproduction is shown in Figure 4, where a visualization for models A, B, and C is shown. The original images are on the diagonal, and in the same row are adversarial examples for this image. These adversarial examples are targeted to what is stated above the column. This is exactly how Xiao et al. [3] visualized it in Figure 1. Additionally, we added colour (yellow/red) to the visualization to show which images were wrongly, and thus successfully misclassified, by the model. By adding this colour, it is clearly visible that, for most images, the attack was unsuccessful. These images are still classified to their ground truth.



Model	**A**	**B**	**C**
Test Accuracy:	95.83%	97.62%	98.67%
Attack Success Rate Targeted:	2.24%	1.34%	1.19%
Attack Success Rate Untargeted:	17.85%	9.31%	5.73%

Table 1: Our results for Models A, B & C.

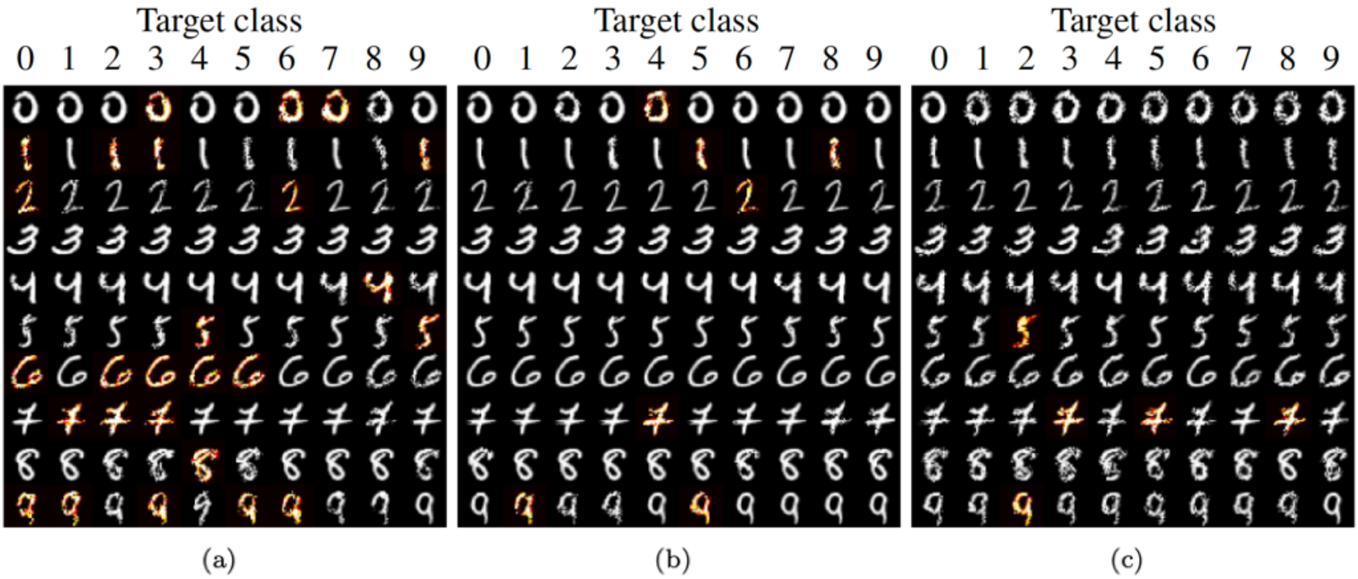


Figure 4: Result for (a) Model A, (b) Model B, (c) Model C. Our reproduction of Figure 2 from [3]. The diagonal shows the original images, and the images in the same row are adversarial examples targeted to what is on top of the column. The yellow/red tinted images are successfully misclassified.

4. Conclusion

Xiao et al. [3] presented in their paper from 2018 a new variant of making adversarial examples. Instead of changing the values of specific pixels, they transformed or distorted the images a little to achieve an effective adversarial attack. They called their method and their paper: ‘Spatially Transformed Adversarial Examples’ [3]. Their results are quite impressive, with an attack success rate of around 99%. While reproducing the paper, we noticed early on that the paper is fairly vague about how they exactly implemented their adversarial examples and tested these. Due to this vagueness, we had to make some implementation choices, which, in the end, potentially led to different numbers than the original paper. Therefore, due to the absence of some information in the paper, we conclude that the paper was unreproducible for us. In the rest of this section, we describe what made it unreproducible for us.

First of all, Xiao et al. [3] did not mention how they built their test set to compute their attack success rate, as seen in Figure 1. We chose to have 10,000 samples from the MNIST set and randomly selected a target that differed from the ground truth. The decision to have 10,000 samples was made due to the available resources. If these were not limited, we would have had 90,000 samples (the test set of MNIST), and we would have computed all adversarial versions. These choices may have led to our lower attack success rate compared to the paper [3].

Secondly, Xiao et al. [3] did not specify a few other things. They did not mention how they initialized the flow fields. Additionally, they did not mention, for LBFGS, what the most influential parameters (e.g. num steps) were initialized with. Lastly, in Figure 1 from Xiao et al. [3], they only have one attack success rate, while we have two in Table 1. From their paper, it is unclear if they reported only one version: targeted, untargeted or both.

To conclude, the main takeaway from our reproduction is that it is relatively complex to reproduce a paper that is quite vague about their implementation and their evaluation. Raff [2] also mentioned the importance of “parameters specified” in papers for their reproducibility. We were not able to have the same results as described in the paper from Xiao et al. [3], and the paper was, therefore, not reproducible for us.

Work Distribution

Emiel den Haan: Code Implementation, Debugging, Report Writing.
Bryan He: Code Implementation, Debugging, Report Writing.
Annabel Simons: Code implementation, Report Writing, HackMD Layout, Poster Making.
Mike Wu: Code Implementation, Debugging, Visualisation.

References

[1] M. Hutson, “Artificial intelligence faces reproducibility crisis,” *Science*, vol. 359, no. 6377, pp. 725–726, Feb. 2018, doi: 10.1126/science.359.6377.725 (https://doi.org/10.1126/science.359.6377.725).

[2] E. Raff, “A step toward quantifying independently reproducible machine learning research,” *CoRR*, vol. 32, pp. 5485–5495, Jan. 2019, arXiv: 1909.06674 (https://arxiv.org/abs/1909.06674).

[3] C. Xiao, J. Zhu, B. Li, W. He, M. Liu, and D. Song, “Spatially transformed adversarial examples,” *CoRR*, Jan. 2018, arXiv: 1801.02612 (http://arxiv.org/abs/1801.02612).

[4] H. Xu, Y. Ma, HC. Liu, D. Deb, H. Liu, JL. Tang, and A. K. Jain. Adversarial attacks and defences in images, graphs and text: A review. *International journal of automation and computing*, vol. 17, pp. 151–178, Apr. 2020, doi: 10.1007/s11633-019-1211-x (https://doi.org/10.1007/s11633-019-1211-x).

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *CoRR*, Dec. 2014, arXiv: 1412.6572 (https://arxiv.org/abs/1412.6572).