THE COMP6248 REPRODUCIBLITY CHALLENGE

Robert Henery
University of Southampton
rmh1g17@soton.ac.uk

Dominik Vranay
University of Southampton
dv1n17@soton.ac.uk

ABSTRACT

In this report, we assess the reproducability of "Nesterov Accelerated Gradient and Scale Invariance for adversarial Attacks" (Lin et al. (2020)). The paper proposes two new algorithms to perform adversarial attacks on deep learning models, with the aim of misclassifying the image without changing the human perception of the image. We detail our re-implementation of their proposed attack algorithms, and the testing methodology used to compare our results with those presented in Lin et al. (2020). Our results show that, despite the attack models and testing methodology being easy to replicate, our results were far worse than those achieved in the original paper. We found that the published findings were still valid however, as they provide better transferability over existing methods.

1 ORIGINAL PAPER APPROACH

The paper we chose to replicate is called "Nesterov Accelerated Gradient and Scale Invariance for Adversarial Attacks" (Lin et al. (2020)), and is is based on adversarial attacks on deep learning models. Adversarial attacks work by "applying human-imperceptible perturbations on the benign input" (Lin et al. (2020)) with the aim of misclassifying the image, in this case by applying noise to the input. The paper we attempt to replicate proposes two new attack algorithms: Nesterov Iterative Fast Gradient Sign Method (NI-FGSM) and Scale-Invariant attack Method (SIM). These proposed algorithms aim to improve the black box performance of adversarial attack algorithms. NI-FGSM improves on the existing MI-FGSM algorithm by using previous accumulated gradients to effectively look ahead. SIM uses scaled images during the creation of adversarial images, which will help with transferability.

2 IMPLEMENTATION

2.1 DATA SET

The dataset used in the publication is the popular ImageNet dataset (Deng et al. (2009)). The ImageNet dataset is a dataset of around 14 million images covering around 100,000 labels, taken from the WordNet hierarchy (Fellbaum (1998)) . We replicated the paper by sampling 1000 images from the ImageNet validation set to train the attack algorithms.

2.2 ALGORITHMS

We implemented all the code in Pytorch using the Pytorch Automatic Differentiation (PAD). The pseudo code from the paper was used for the implementation of the attack algorithms, and some features we ported from the Tensorflow code provided by the authors of Lin et al. (2020), as they were not clearly described in the paper, e.g. transformations used in Diverse Input Method (DIM).

To make the code more readable, each attack algorithm is implemented in a different function before the implementation it is needed. It takes batched inputs and labels, and a pretrained model, and returns the adversarial examples and the number of relabelled images in that batch. The attack success rate is calculated by firstly, using the model to predict the labels on the original data and then predict the labels on the adversarial data. We then count the number of changed labels, and divide the number by the total number of images in the data set to get the final performance.

The adversarial images are generated using PAD and the algorithms. The images are copied into a tensor which requires grad and this grad is calculated using the cross-entropy loss on the predictions at each step. Afterwards, the differences from the original images are clamped into the range of $[-\epsilon, \epsilon]$. This is done for T iterations.

2.3 MODELS

The proposed adversarial attack algorithms were tested against seven different deep learning models, four he four regularly trained models: Inception-v3, Inception-v4, Inception-Resnet-v2 and Resnet-101, and three adversarially trained models: include two variations of the Inception-v3 model and a variation on the Inception Resnet-v2 model, intended to improve the performance of these models against adversarial attacks.
We used pretrained implementations of these models, as we did not have the time nor resources to develop and train these our-self. We used Pytorch and TorchVision for the implementation of these models, with the Inception-v3 and Resnet-101 models having implementations directly available within the these modules. The Inception-v4 and Inception-Resnet-v2 models were sourced from a public GitHub repository of pretrained deep learning models [1]. In the paper we were replicating, the approach for implementing the adversarially trained models was not made clear, and we were only able to find one existing implementation of an adversarially trained Inception-v3 model, from a Python module called Timm.

3 APPROACH

For each attack algorithm, we trained an instance of the algorithm against each of the deep learning models, testing against all models each time. The test against the model on which the attack algorithm is trained on is defined as a white box attack, and denoted with a "*" in the results table X. The other attacks are defined as black box attacks, as the attack algorithm has no knowledge of the model before or during the attack. Note that some results (such as figure 4 in section 4) feature attacks trained over multiple models. The algorithms had to be slightly modified to accommodate learning on multiple models, hard coding them to use the 4 training models together.

The performance of the attack algorithms is measured by the Attack Success Rate, which is defined as the percentage of times the adversarial attack results in the predicted label being changed from the predicted label of the image before the attack.

The hyper-parameters used were the same as in the original paper: $\epsilon = 16$, $T = 10$, $\alpha = \epsilon/T$, $\mu = 1.0$, $p = 0.5$, $kernel = 7x7$ and $m = 5$

4 RESULTS AND ANALYSIS

Tables 1, 2, and 3 show the results of testing our approach, as defined in section 3. These show the Attack success rates (%) of adversarial attacks against five models, when trained against a single model. These tables are our replication of tables 1(a), 1(b) and 1(c) in Lin et al. (2020) respectively.

From looking at the results in Table 1, Table 2 and Table 3, when compared against those presented in the paper we are replicating, we can see that our replication of these attack algorithms performs much worse than those presented in the publication. The white box attacks (indicated by a "*") show high attack success rates, similar to those presented in the publication, however black box attack success rates are very low in comparison.

Table 4 shows the Attack success rates (%) of adversarial attacks against five models, when trained against multiple models. As can be seen in Table 4 the original results are again much better than ours. This could have been caused by the difference in calculating the gradients on multiple models as they are not specifying how it is done. We used the sum of all losses and calculated the gradient from that. Another cause could be that the algorithms are not correctly implemented. The only value which is close to the original results is the Res-101 as that has a difference of max 12%.

[1]https://github.com/Cadene/pretrained-models.pytorch/tree/master/pretrainedmodels/models

Table 1: A reproduction of table 1(a) from Lin et al. (2020)

Model	Attack	Inc-v3	Inc-v4	Inc-Res-v2	Res-101	Inc-v3-adv
Inc-v3	SI-NI-TIM (Theirs)	100.0*	77.2	75.8	66.5	51.8
	SI-NI-TIM (Ours)	97.4*	14.1	12.3	11.2	12.3
Inc-v4	SI-NI-TIM (Theirs)	83.5	100.0*	76.6	68.9	57.8
	SI-NI-TIM (Ours)	20.9	93.0*	15.8	15.2	10.9
Inc-Res-v2	SI-NI-TIM (Theirs)	86.4	83.2	99.5*	77.2	66.1
	SI-NI-TIM (Ours)	27.2	22.8	81.9*	17.3	12.5
Res-101	SI-NI-TIM (Theirs)	78.3	74.1	73.0	99.8*	58.9
	SI-NI-TIM (Ours)	17.1	12.6	10.2	98.6*	10.2

Table 2: A reproduction of table 1(b) from Lin et al. (2020)

Model	Attack	Inc-v3	Inc-v4	Inc-Res-v2	Res-101	Inc-v3-adv
Inc-v3	SI-NI-DIM (Theirs)	99.6*	84.7	81.7	75.4	36.9
	SI-NI-DIM (Ours)	97.0*	13.6	10.8	11.3	10.6
Inc-v4	SI-NI-DIM (Theirs)	89.7	99.3*	84.5	78.5	47.6
	SI-NI-DIM (Ours)	18.7	93.4*	12.0	14.3	8.2
Inc-Res-v2	SI-NI-DIM (Theirs)	89.7	86.4	99.1*	81.2	55.0
	SI-NI-DIM (Ours)	19.5	17.6	82.1*	14.4	9.6
Res-101	SI-NI-DIM (Theirs)	88.7	84.2	84.4	99.3*	52.4
	SI-NI-DIM (Ours)	11.9	9.7	7.1	99.0*	7.3

Table 3: A reproduction of table 1(c) from Lin et al. (2020)

Model	Attack	Inc-v3	Inc-v4	Inc-Res-v2	Res-101	Inc-v3-adv
Inc-v3	SI-NI-TI-DIM (Theirs)	99.6*	85.5	80.9	75.7	61.5
	SI-NI-TI-DIM (Ours)	95.0*	11.5	9.5	8.6	11.6
Inc-v4	SI-NI-TI-DIM (Theirs)	88.1	99.3*	83.7	77.0	65.0
	SI-NI-TI-DIM (Ours)	18.7	89.6*	13.2	12.7	9.2
Inc-Res-v2	SI-NI-TI-DIM (Theirs)	89.6	87.0	99.1*	83.9	74.0
	SI-NI-TI-DIM (Ours)	22.3	19.3	77.3*	12.8	11.6
Res-101	SI-NI-TI-DIM (Theirs)	86.4	82.6	84.6	99.0*	72.6
	SI-NI-TI-DIM (Ours)	13.5	10.0	8.3	96.9*	9.0

Table 4: A reproduction of table 2 from Lin et al. (2020)

Attack	Inc-v3*	Inc-v4*	Inc-Res-v2*	Res-101*	Inc-v3-adv
SI-NI-TIM (Theirs)	100.0	100.0	100.0	100.0	93.2
SI-NI-TIM (Ours)	84.5	77.4	61.1	92.4	19.3
SI-NI-DIM (Theirs)	100.0	100.0	100.0	99.9	88.2
SI-NI-DIM (Ours)	79.8	73.1	56.5	94.6	14.3
SI-NI-TI-TIM (Theirs)	99.9	99.9	99.9	99.9	96
SI-NI-TI-TIM (Ours)	79.3	68.6	52.2	87.9	17.8

Figure 1 (a) looks similar, but still, our implementation does not perform as well as theirs. For figures 1 (b) and 1 (c), there are no similarities at all. The results are completely different, and the tendencies of the plots are different as well. Theirs have a rising tendency, but ours look like they are random. This can be caused by using $\alpha = \epsilon/T$ as they wrote it in the pseudo code instead of 1.6, which was defined in their hyper-parameters section.

From looking at the results in Table 5, we can see that the performance of our implementation is much lower than what was published. The white box attack shows again a high attack success rates, very similar to that presented in the publication.

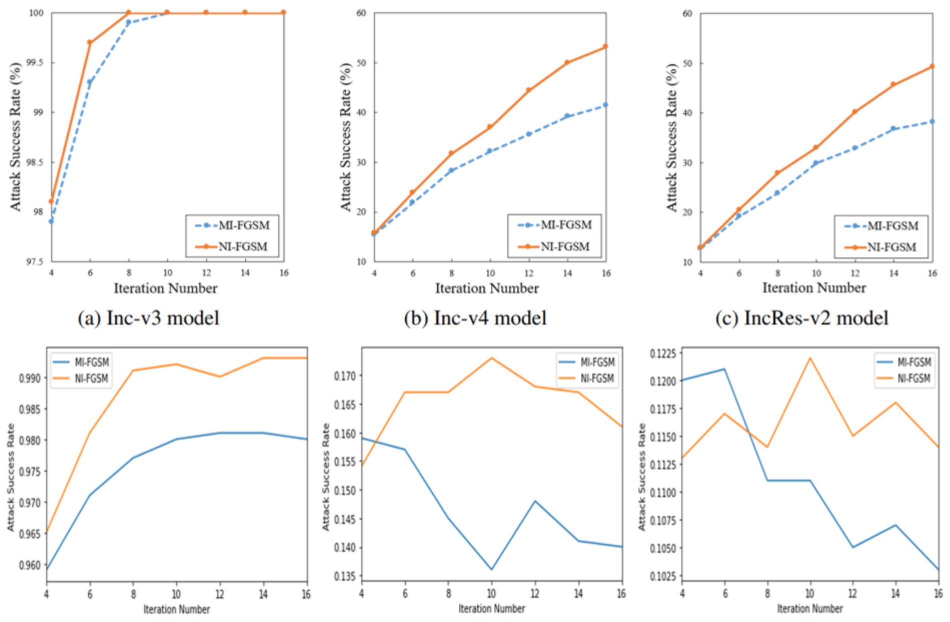

Figure 1: Differences between Figure 2 of Lin et al. (2020) and our recreation

Table 5: A reproduction of table 4 from Lin et al. (2020)

Attack	Inc-v3*	Inc-v4	Inc-Res-v2	Res-101	Inc-v3-adv
NI-FGSM (Theirs)	100.0	52.6	51.4	41.4	12.9
NI-FGSM (Ours)	99.2	17.3	12.2	12.6	7.9
SI-NI-FGSM (Theirs)	100.0	76.0	73.3	67.6	31.6
SI-NI-FGSM (Ours)	98.5	17.5	14.2	13.6	11.6

5 Conclusion

In general, the algorithms can be easily re-implemented, but the results are not as good as the ones published. There could be several causes for these results, but one probable is that the published paper was using the models which they trained themselves and had worse results than the pretrained networks we used. However the findings were still valid, as the algorithm they proposed has better transferabillity than the existing methods. In terms of resources, all results could be done using Google Colab and the longest algorithm was the SI-NI-TI-DIM, which took on average 1.5 hours for the multi-model setup. The total computation time was 12 hours and was done in parallel by 2 people(6 hours each) on separate Colab runtimes.

References

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E. Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks, 2020.