Deep Learning on Computational Accelerators - 236781

Optimization of Patch Adversarial Attacks on a SOTA Visual Odometry Model with Projected Nesterov Momentum Iterative Fast Gradient Sign Method

Name	ID	Email
Yotam Granov	211870241	yotam.g@campus.technion.ac.il

Faculty of Computer Science
Technion - Israel Institute of Technology
Spring 2022

Contents

Abstract

In this work, we explore optimization methods for producing patch adversarial attacks on a state of the art visual odometry model, TartanVO. The goal of the work is to generate a patch perturbation to be applied to synthetic images which the VO model will use to try to identify the location of the subject along the trajectory, with the best patch causing the greatest error in the VO model's predictions. The patches will be produced using a technique known as Projected Nesterov Momentum Iterated Fast Gradient Sign Method (PNMI-FGSM), and can be shown to be more quickly generated than with another popular method called Projected Gradient Descent (PGD), with roughly the same levels of loss incurred in the VO model.

1 Introduction

Back in 2014, Szegedy et. al. [6] presented an intriguing property of deep neural networks. They showed that one can cause a neural network to misclassify an input image by applying a certain hardly perceptible perturbation, which is obtained by maximizing the network's prediction error. A year later, Goodfellow et. al. [3] presented an effective new method for producing these adversarial examples, which they called the Fast Gradient Sign Method (FGSM). That same year, Bubeck proposed [1] the Projected Gradient Descent (PGD) attack scheme for producing adversarial attacks on deep neural nets. The majority of patch adversarial attack generation schemes studied and proposed since then have been optimized variations of these two methods, each with their own benefits and drawbacks.

FGSM-Based Attacks. The FGSM scheme involves finding an adversarial example x^* by maximizing a loss function $J(x^*, y)$, where J is often the cross-entropy loss. FGSM generates adversarial examples to meet the L_∞ norm bound $||x^* - x||_\infty \leq \epsilon$ as:

$$x^* = x + \epsilon \cdot \text{sign}(\nabla_x J(x, y))$$

where $\nabla_x J(x, y)$ is the gradient of the loss function with respect to x. Two years after FGSM was introduced, Kurakin et. al. [4] proposed an iterative modification to the scheme which iteratively applies the fast gradient multiple times with a small step size α, as follows:

$$x_0^* = x,\ x_{t+1}^* = x_t^* + \alpha \cdot \text{sign}(\nabla_x J(x, y))$$

After each iteration, x_t^* is clipped into the ϵ vicinity of x in order to satisfy the L_∞ bound. This method was shown to be stronger than the vanilla FGSM scheme, but at the cost of worse transferability. In order to improve transferability, Dong et. al. [2]

successfully integrated momentum into iterative FGSM, allowing the gradient descent process to escape local minima while still converging to a solution in a stable manner. Introducing a new momentum parameter μ, their Momentum Iterative FGSM (MI-FGSM) algorithm involves accumulating a velocity vector in the gradient direction of the loss function across iterations, which can be seen in the following step:

$$g_{t+1} = \mu g_t + \frac{\nabla_x J(x_t^*, y)}{||\nabla_x J(x_t^*, y)||_1}$$

The perturbation is then updated according to the following rule:

$$x_0^* = x,\ x_{t+1}^* = x_t^* + \alpha \cdot \text{sign}(g_{t+1})$$

This method allows for the memorization of previous gradients, which helps the gradient descent process to barrel through narrow valleys, small humps and poor local minima or maxima. Xu [7] further improved on this method by applying the Nesterov momentum method to MI-FGSM, which makes a correction during the gradient update to avoid moving too fast. This method was shown to improve the success rate of adversarial attacks, while producing smaller perturbation values of the adversarial examples. In this method, a new variable $\theta = x_t^* + \mu g_t$ is introduced, and the gradient is calculated using this paramter as the perturbation:

$$g_{t+1} = \mu g_t + \frac{\nabla_x J(\theta, y)}{||\nabla_x J(\theta, y)||_1}$$

The perturbation is then updated according to a new rule:

$$x_0^* = x,\ x_{t+1}^* = x_t^* + \alpha \cdot \text{sign}((1-\mu) g_t + \mu g_{t+1})$$

This algorithm, called Nesterov Momentum Iterative Fast Gradient Sign Method (NMI-FGSM), was shown to be more successful at attacking SOTA image classification neural nets than its predecessors (specifically vanilla FGSM and MI-FGSM). Our work in this report will aim to modify the NMI-FGSM further in order to introduce new improvements.

PGD-Based Attacks. Here will will briefly discuss adversarial attacks reliant on the Projected Gradient Descent (PGD) method, such as that used by Nemcovsky et. al. [5]. In the PGD method, a perturbation P is randomly initialized from a uniform distribution between 0 and 1. The gradient is then calculated according to the following rule:

$$g = \nabla_P J(VO(A(x, P)), y)$$

and the perturbation is updated according to the following rule:

$$P = P + \alpha \cdot \text{sign}(g)$$

Finally, the new perturbation is clipped back into the domain $[0, 1]$ in order to ensure that is will indeed produce a valid RGB image. This clipping is (partially) what separates PGD from FGSM, and we will implement a similar clipping scheme in our own algorithm.

2 Methods

This body of work is heavily based on the work of Nemcovsky et. al. [5], who explained the same patch adversarial perturbation, visual odometry attack, and task criterion that we use here in their own paper.

Recalling their work, our goal will be to find a patch adversarial attack P_a such that it maximizes the loss criterion l in the following manner:

$$P_a = \text{argmax}_{P \in \mathcal{I}}(l(VO(A(\{I_t\}_{t=0}^{L}, P)), \{\delta_t^{t+1}\}_{t=0}^{L-1})$$

where $A(\{I_t\}_{t=0}^{L}, P)$ is the perturbed image, $VO : (\mathcal{I} \times \mathcal{I}) \rightarrow (\mathcal{R}^3 \times so(3))$ is the monocular visual odometry model, $\{\delta_t^{t+1}\}_{t=0}^{L-1}$ are the ground truth motions, and $\mathcal{I} = [0,1]^{3 \times w \times h}$ is the normalized RGB image space for some width w and height h. As in Nemcovsky's paper, we choose the target criterion used for adversarial attacks to be the root mean square (RMS) deviation in the 3D physical translation between the accumulated trajectory motion (estimated by the VO model) and the ground truth trajectory.

Attack optimizer. The difference between our work and Nemcovsky's [5] begins with the attack optimizer for the adversarial patches. Whereas they chose to optimize the adversarial patch P using a PGD adversarial attack with l_∞ norm limitation, our work will explore the application of a new PNMI-FGSM adversarial attack scheme for patch optimization. This scheme is can be found in the Appendix section, under Algorithm 1 (non-universal case). In our implementation of the PNMI-FGSM, we uniquely choose to randomly initialize the patch P, rather than initialize it to be the albedo image (as was done by Dong et. al. [2]). We saw that this almost always produced better patches in the end, and allowed us to avoid undesirable local minima. We implement the θ variable used for the Nesterov method (in NMI-FGSM), but unlike Xu [7] we clip its value back to the region $[0,1]$, in order to ensure that it remains a valid RGB image. We update the gradient with the same rule as NMI-FGSM, and update the perturbation using the same rule as MI-FGSM. We then clip P back to the region $[0,1]$ (such as is done in PGD) in order to again ensure that we obtain a valid RGB image.

Optimization scheme. In order to improve generalization, we propose a universal optimization scheme (Algorithm 2), however due to time constraints and personal issues the author of this work did not have time to implement it successfully. Given the 50 trajectories in the data set, we would propose to split the total dataset into training and testing sets using an 80-20 train-test split. We would then use a k-fold cross validation scheme on the training data, and splitting it so that there is a 60-20-20 train-validation-test split in the data. Such splitting and cross validation schemes are common practice in the field of machine learning, and would likely serve us well in our mission to optimize the generalization of our adversarial attacks.

Optimization and evaluation criteria. For both the optimization and evaluation of our attacks, we considered only one criteria - the root mean squared (RMS) error l_{RMS} over the partial trajectories with the same origin as the full trajectory [5]:

$$\begin{aligned} l_{RMS}(VO(A(\{I_t\}_{t=0}^{L}, P)), \{\delta_t^{t+1}\}_{t=0}^{L-1}) &= \sum_{i=1}^{L} l_{VO}(VO(A(\{I_t\}_{t=0}^{i}, P)), \{\delta_t^{t+1}\}_{t=0}^{i-1}) \\ &= \sum_{i=1}^{L} ||q(\prod_{t=0}^{i-1} VO(I_t^P, I_{t+1}^P)) - q(\prod_{t=0}^{i-1} \delta_t^{t+1})||_2 \end{aligned}$$

We experimented with various loss criteria (still using only RMS), and our results will be explained later in the Experiments section.

3 Implementation and Experiments

In this section we will describe the experimental settings used for comparing the effectiveness of our method and various baselines. Over all experiments, we conducted training and evaluation over the same trajectories (which is not beneficial for generalization, but again there were personal constraints at hand - we proposed a method for improved generalization earlier in the report), and we focused mainly on various aspects of our PNMI-FGSM model and its performance against the PGD model.

(1) *Hyperparameter tuning experiment.* In this experiment, we conducted a hyperparameter search in order to determine which values for the learning rate α and momentum parameter θ would produce optimal results. We created a grid $\{\alpha \in [0.0001, 0.001, 0.01, 0.1], \mu \in [0.0001, 0.001, 0.01, 0.1]\}$, and allowed our PNMI-FGSM model to train with all 16 possible combinations of (α, μ) over 20 iterations. We obtained that $(\alpha, \mu) = (0.001, 0.001)$ was by far the most successful combination. All of the other combinations either progressed way too slowly or got stuck early on in poor local minima, while the $(\alpha, \mu) = (0.001, 0.001)$ configuration was able to continue improving the perturbations, even after 200 iterations (this was tested in a separate experiment).

(2) *Loss criterion experiment.* In this experiment, we compared the effects of using different loss criteria for optimization and evaluation. For an untargeted attack, the loss criterion l_{VO} was defined as follows:

$$l_{VO} = a \cdot l_{RMS,translation} + b \cdot l_{RMS,rotation} + c \cdot l_{RMS,flow}$$

We limit the values of a, b, c to be either 0 or 1 for the sake of ease for experimentation. We tested the following four cases over 10 iterations of training an PNMI-FGSM attack with $\alpha = 0.001$ and $\mu = 0.001$:

$$(a, b, c) \in \{(1, 0, 0), (1, 1, 0), (1, 0, 1), (1, 1, 1)\}$$

(3) *PGD vs. PNMI-FGSM experiment.* In this experiment, we directly compared the PGD attack from Nemcovsky et. al. [5] to our PNMI-FGSM method, over 200 iterations. The PGD attack used $\alpha = 0.05$, while the PNMI-FGSM attack used $\alpha = 0.001$ and $\mu = 0.001$ (which were found to be optimal according to our previous experiment).

4 Results and Discussion

In this final section, we will discuss the results obtained during our experiments, as well as our conclusions for the study as a whole and ideas for future improvements.

(1) We obtained interesting results during the hyperparameter tuning experiment of the PNMI-FGSM attacks. We noticed that for attacks where $\alpha > 0.001$ and $\mu > 0.001$, the gradient descent process almost always resulted in poor local minima that the algorithm could not overcome. This result is caused by the clipping steps in our method, which force the perturbation to maintain values in $[0, 1]$. When the step size α and momentum parameter μ are too large (imagine $\delta x \sim O(0.1)$), the perturbation's values very quickly exceed those bounds and are clipped to either 0 or 1 (depending on which side of the bounds they sit). Unfortunately, the values then get stuck close to either 0 or 1, and this produces perturbations with many regions of black spots and white spots, which can be seen for example in the following perturbation:

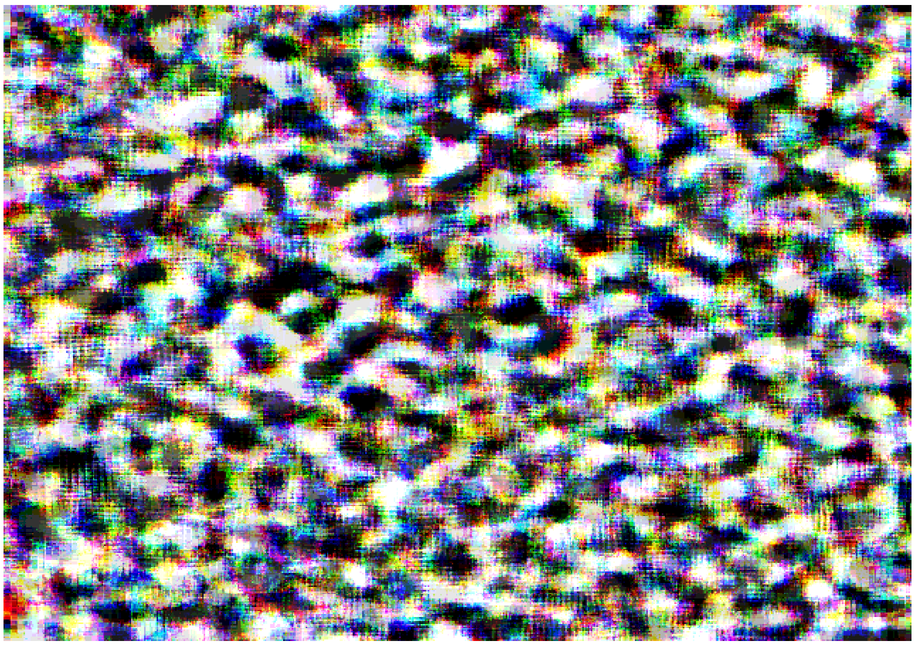

Figure 1: PNMI-FGSM perturbation, after 20 iterations with $(\alpha, \mu) = (0.1, 0.1)$

Perturbations with numerous black and white spots tend to remain in poor local minima and can't be further trained successfully, regardless of how many iterations are conducted. This result helped us to identify that small (α, μ) values should be more successful in traversing local minima and exploring more of the valid configuration space. We were cautious to avoid hyperparameter values that are too small (< 0.001), though, since they cause the training process to proceed too slowly.

(2) In the loss criterion experiment, we obtained the following results:

Translation	Rotation	Flow	Best Loss after 10 Iterations
RMS	None	None	153.286
RMS	*RMS*	*None*	*155.200*
RMS	None	RMS	153.990
RMS	RMS	RMS	153.888

Figure 2: Best loss after 10 iterations for various loss criteria for the optimization and evaluation of the PNMI-FGSM perturbation with $(\alpha, \mu) = (0.1, 0.1)$

We see that the best loss is obtained when the translation and rotation RMS losses are taken into account, without influence from the flow RMS loss. Thus, the most successful loss criterion we obtained is:

$$l_{VO} = l_{RMS,translation} + l_{RMS,rotation}$$

It seems quite reasonable that a loss criterion that takes into account more than just translation would be more effective for training adversarial attacks, and indeed we see that this assumption is valid.

(3) One of the first results that we obtained was that the PNMI-FGSM was able to significantly decrease training time, with each iteration requiring around 90 seconds on average for training and evaluation, whereas PGD required around 120 seconds on average. As predicted, the PNMI-FGSM algorithm does indeed guarantee quicker training, which is a very desirable outcome.

In our direct comparison of the PGD and PNMI-FGSM methods, we obtained the following results:

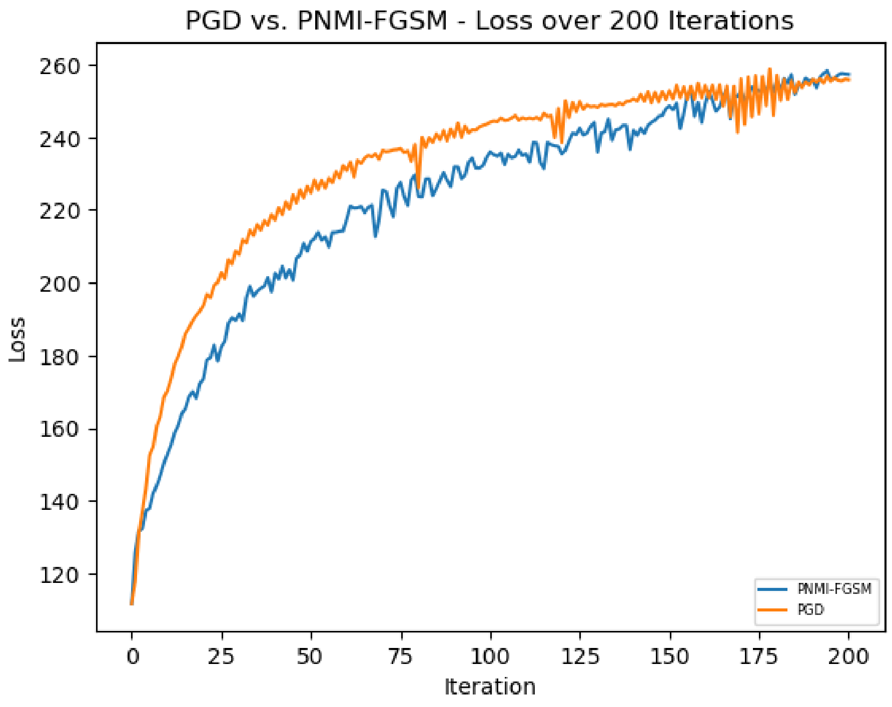

Figure 3: Loss caused by PGD vs. PNMI-FGSM attacks, after 200 iterations

We see that after 200 iterations, the PNMI-FGSM attacks (with $(\alpha, \mu) = (0.001, 0.001)$) incur greater loss than the PGD attacks (with $\alpha = 0.05$). While the loss values themselves are quite close, we note that the PNMI-FGSM training was significantly faster, requiring only 5 hours of training. The PGD training, on the other hand, required 6.67 hours of training for the same number of iterations. We consider this result to be a great success, indicating that it's possible to improve both the success level of the adversarial attacks as well as the training time required to produce them. In the following image, we can see the best perturbation produced by the PNMI-FGSM:

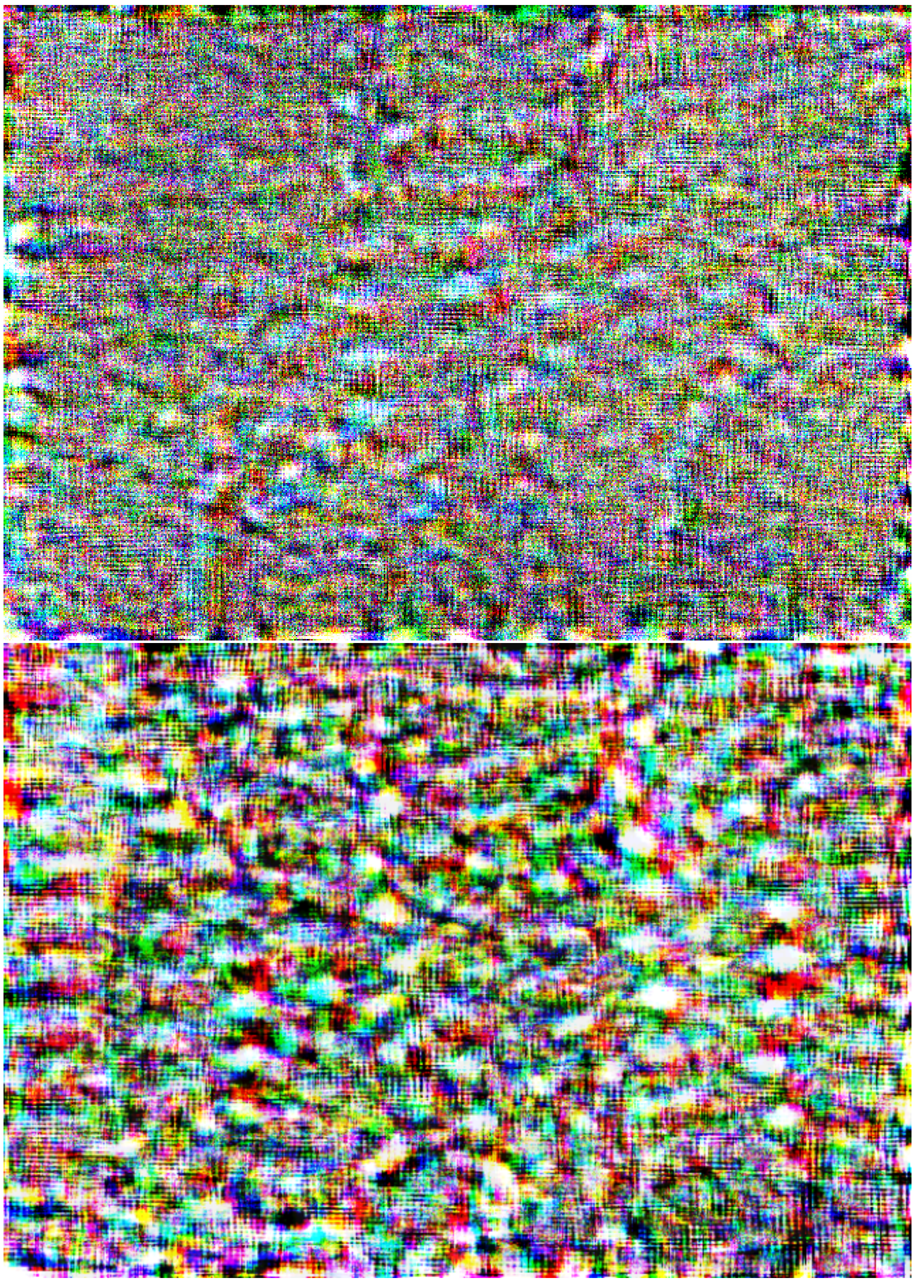

Figure 4: Top: Best PNMI-FGSM perturbation, after 200 iterations with $(\alpha, \mu) = (0.001, 0.001)$*. Bottom: Best PGD perturbation, after 200 iterations with* $\alpha = 0.05$*.*

The following table shows a comparison of the performance between the two models for this experiment:

Model	**Best Loss**	**Iterations**	**Training Time**
PNMI-FGSM	*258.370*	200	~5 hours
PGD	255.850	200	~6.67 hours

Figure 5: Loss caused by PGD vs. PNMI-FGSM attacks, after 200 iterations

Concluding remarks. The author would like to take this chance to thank the course staff for a great semester, and for a very fun and intriguing final project. He would like to note that he invested a lot of time into this assignment but his progress and results were greatly affected by personal circumstances, his partner dropping the assignment very close to the deadline, and a rigorous summer working schedule that severely impaired his ability to focus on this project. There is plenty of room for improvement on the work presented here, and we will offer some suggestions for future work that would improve the research at hand.

Attack optimizer. The PNMI-FGSM algorithm proved effective at reducing training time, but there is much more work that could be done to investigate it and improve it. In the case of hyperparameter tuning, a more robust grid search as well as a random search might turn up more effective hyperparameter values. Additionally, implementing some sort of scheduling approach for varying the hyperparameters might prove fruitful for exploring the configuration space even deeper and surpassing more weak local minima. It's also possible that SOTA optimizers such as Adam could be implemented, to further enhance the gradient descent process. It would also serve us well to directly compare more methods next time, such as directly comparing PNMI-FGSM to MI-FGSM, APGD, etc.

Optimization scheme. Our optimization scheme was admittedly lacking here, but we already provided a suggestion earlier in this report for how to go about implementing such a scheme for this work. This would surely improve the generalization of our attacks, which is the goal of the work at the end of the day.

Optimization and evaluation criteria. It would serve us well to develop more advanced experiments for comparing the effectiveness of various loss criteria. In the future, we propose to explore the mean partial RMS (MPRMS) loss as well as the RMS that was solely used here. We also kept the optimization and evaluation criteria the same in this work, but it's possible that making them different (i.e. conducting optimization with MPRMS to improve generalization and evaluation with RMS to ensure robustness) would prove fruitful. We should also further investigate the effects of translation, rotation, and flow losses on the effectiveness of our loss criterion - it is possible that there is a different combination than the one we chose in this work that would prove more successful.

References

[1] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–358, 2015.

[2] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. *Unknown*, 2018.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *ICLR 2015 Conference Proceedings*, 2015.

[4] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *ICLR 2017 Conference Proceedings*, 2017.

[5] Yaniv Nemcovsky, Matan Yaakoby, Alex Bronstein, and Chaim Baskin. Physical passive patch adversarial attacks on visual odometry systems. *ACCV 2022 Conference Proceedings*, 2022.

[6] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *ICLR 2014 Conference Proceedings*, 2014.

[7] Jin Xu. Generate adversarial examples by nesterov-momentum iterative fast gradient sign method. *ICSESS 2020 Conference Proceedings*, 2020.

5 Appendix A: PNMI-FGSM Algorithm

Algorithm 1 Projected Nesterov Momentum Iterative Fast Gradient Sign Method

```
    Input $VO$: VO model
    Input $A$: Adversarial patch perturbation
    Input $(x, y)$: Trajectory to attack and it's ground truth motions
    Input $(l_{train}, l_{eval})$: Train and evaluation loss functions
    Input $\alpha$: Step size for the attack
    Input $\mu$: Momentum parameter

 1: $P \leftarrow \text{Uniform}(0, 1)$
 2: $P_{best} \leftarrow P$
 3: $l_{best} \leftarrow 0$
 4: $g_0 \leftarrow 0$
 5: for $k = 0$ to K do
        optimization step:
 6:     $\theta \leftarrow P + \alpha \cdot \mu \cdot g_k$
 7:     $\theta \leftarrow Clip(\theta, 0, 1)$
 8:     $\hat{y} \leftarrow VO(A(x, \theta))$
 9:     $g_{k+1} \leftarrow \mu g_k + \frac{\nabla_x l_{train}(\theta, \hat{y})}{||\nabla_x l_{train}(\theta, \hat{y})||_1}$
10:     $P \leftarrow P + \alpha \cdot sign(g_{k+1})$
11:     $P \leftarrow Clip(P, 0, 1)$
        evaluate patch:
12:     $Loss \leftarrow l_{eval}(VO(A(x, P)), y)$
13:     if $Loss > Loss_{best}$ then
14:         $P_{best} \leftarrow P$
15:         $Loss_{best} \leftarrow Loss$
16:     end if
17: end for
18: return $P_{best}$
```