UNIVERSAL ADVERSARIAL PERTURBATIONS ON VISUAL ODOMETRY SYSTEMS

Tal Peer
Technion - Israel Institute of Technology
tal.peer@campus.technion.ac.il

ABSTRACT

As we seek to deploy machine learning systems not only on virtual domains, but also in real systems, it becomes critical that we examine not only whether the systems don't simply work "most of the time", but which are truly robust and reliable. Deep neural networks are known to be susceptible to adversarial perturbations, small perturbation which alter the output of the network and exist under strict norm limitations. While such perturbations are usually tailored to a specific input, it is also possible to produce universal perturbations that achieve a high rate of miss-classification on a set of inputs. Universal perturbations present a more realistic use case for adversarial perturbations, as the adversary does not require to be aware of the model's input. this work, We focus on one of the most challenging cases —monocular VO—where the only input is a monocular video stream,with the task of estimating a robot's position and orientation from visual measurements.

1 Introduction

A VO model aims to infer the motion (position and orientation) between two respective viewpoints. Such models are frequently used by visual based autonomous navigation systems. In this project, we produced an universal adversarial perturbations for a given VO model, aiming to maximise its physical 3D deviation.

Papers Review:

1."Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-free Attacks": The paper discuss and propose number of extensions to the PGD-attack overcoming failures, such as sub-optimal step size, and non-curved objective functions. Combining the proposing extensions, the authors proposing the APGD method. The APGD method maintaining the idea of PGD attack optimizer, with step size halving in every few-iteration called check points. APGD is a parameter-free, computationally affordable and user-independent ensemble of attacks to train or test adversarial robustness. The authors compared APGD to over 50 models from papers published at recent top machine learning and computer vision venues, and achieved lower robust test accuracy in 98% of the reported in the mentioned papers.

2. "Metrics for 3D Rotations: Comparison and Analysis": The paper present a detailed analysis of six functions for measuring distance between rotation matrix, and demonstrate the importance of those distance functions to be invariant. Finaly, the paper conclusion is that it is more efficient to use quaternions for 3D rotations. I chose the 4 criteria for the rotation loss.

3. "Guided Optical Flow Learning": The paper represent the most common use optical-flow loss function, the End-Point-Error loss, alse reffered as EPE. In many more papers, such as "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow", and "EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras", the EPE loss is highly relevant for testing the discussed optical-flow based models. Hence, I chose to use the EPE loss for the optical-flow loss.

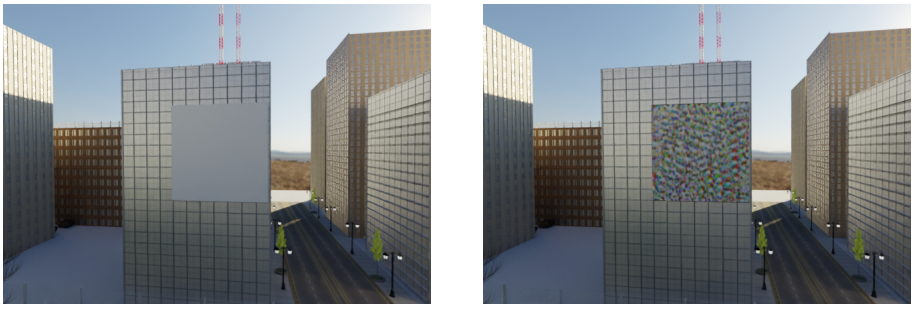


Figure 1: Original Image VS Perturb Image

2 Methods

The original approach for the task was as follow:

1. Optimization Scheme: All Data were used for training, without manging any partition for eavluation-set or test-set.

2. Attack Optimizer: The pre-implemented PGD optimizer were used as optimizer, with default hyper-parameters:

Step Size	0.05
Number of Epochs	100
Number of Multiple Restarts	8
epsilon	1
Batch Size	1

project requirements were to keep other hyper-parameters in there default values such as - eps, batch size, num of workers for data-loaders, etc.

3. Loss Criterion for Training,Evaluation and Testing: All of the mentioned phases were using the RMSE loss with respect to the translation loss only.

Hence, based on the following pre-existing approaches, I have implemented the next modifications:

1. Optimization Scheme: Visual Odometry domain suffers from lack of data. Hence, in order to achieve the best results on the given 50 trajectories, constructed from 5 different data-sets, I have split the data into 5 folds and iterate over them by the following manner: each iteration, 4 data-sets were used for training process, and the fifth for the evaluation. In this way, data-set were used for training phase 4 times total, and once for evaluation.

2. Attack Optimizer: In order to implement the APGD attack, I have used the PGD frame-work, then added the required parameters and code implementation for the APGD. Any implementation nor modification is based on the reviewed paper "Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-free Attacks".

3. Loss Criterion for Train, Evaluation and Test Phases: In order to take advantage of rotation loss and optical-flow loss, I have implemented the MSE loss for optical-flow-loss and quat-product loss for the rotation-loss. Since the task pre-existed code suggesting the conversion of the rotation defined by SO(3) group matrix to quaternions, I have find this objectives loss functions the best-fit for the optimization process, since they are taking the quarternions representation of rotation-matrix in advantage.

Algorithm 1 APGD

Input: f, S, $x^{(0)}$, η, N_{iter}, $W = \{w_0, \ldots, w_n\}$
Output: $x_{\max}$, $f_{\max}$
$x^{(1)} \leftarrow P_{\mathcal{S}}\left(x^{(0)} + \eta \nabla f(x^{(0)})\right)$
$f_{\max} \leftarrow \max\{f(x^{(0)}), f(x^{(1)})\}$
$x_{\max} \leftarrow x^{(0)}$ **if** $f_{\max} \equiv f(x^{(0)})$ **else** $x_{\max} \leftarrow x^{(1)}$
for $k = 1$ **to** $N_{\text{iter}} - 1$ **do**
  $z^{(k+1)} \leftarrow P_{\mathcal{S}}\left(x^{(k)} + \eta \nabla f(x^{(k)})\right)$
  $x^{(k+1)} \leftarrow P_{\mathcal{S}}\Big(x^{(k)} + \alpha(z^{(k+1)} - x^{(k)}) + (1-\alpha)(x^{(k)} - x^{(k-1)})\Big)$
  if $f(x^{(k+1)}) > f_{\max}$ **then**
    $x_{\max} \leftarrow x^{(k+1)}$ and $f_{\max} \leftarrow f(x^{(k+1)})$
  end if
  if $k \in W$ **then**
    if Condition 1 **or** Condition 2 **then**
      $\eta \leftarrow \eta/2$ and $x^{(k+1)} \leftarrow x_{\max}$
    end if
  end if
end for

Figure 2: Figure 1: APGD Pseudo Code.

3 Implementation and Experiments

My experiments focused on three parameters:
- Optimizer
- Objective Function Weights for Rotation Optical-Flow
- Initializing and Updating Optimizer Step Size

3.1 Optimizer

As described above, My chosen Optimizer is APGD. The implementation is in "personalize attack" module. I used the pre-existing code of the PGD attack, with the following modification:

Additional Methods:

compute checkpoint: The function create the check points as described before, using initial parameters for p-values: 0, 0.11. Note that the paper author recommend to initial the second p-value to 0.22, and my chosen value of 0.11 is due to the number of conducted iteration is 50, instead of 100 as the paper describe. For each checkpoint we save the best objective function loss result, and the adversarial perturbation which led to the high result loss.

update step size: in The function setting the two mentioned conditions. In each check point, we call the method, and the optimizer step size is halved if 1 of the 2 conditions is preformed.

new gradient ascent step: The function implementing the APGD optimization scheme, by implementing the rows 7-9 in the pseudo code, presented at Figure 1. we set alpha=0.75, which regulated the influence of the previous update on the current one. This for keep a bias from the previous step. Note that in addition to the new perturbation, this function returns the accumulated loss of the train process. This for optimization inner-checks such as early stopping.

3.2 Rotation and Optical-Flow Objective Loss Function

In this experiments, I checked for different weights for the rotation and optical-flow loss in training phase, in order to take advantage of those while optimizing the adversarial patch.
The chosen Rotation Criterion is the quat-product loss,as mentioned above:

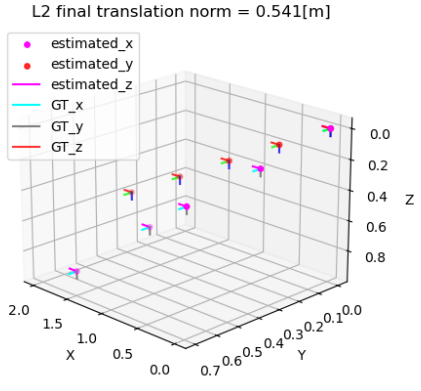
$$\Phi_4 : S^3 \times S^3 \to \mathbb{R}^+,$$
$$\Phi_4(\mathbf{q}_1, \mathbf{q}_2) = 1 - |\mathbf{q}_1 \cdot \mathbf{q}_2|.$$

The chosen Optical-Flow criterion, is the EPE-Loss which I implemented using the nn.MSE loss from PyTorch Library.

Experiments conducted by running 50 iteration of each of the following weights:

rotation factor,flow factor	0,0
rotation factor,flow factor	0.5,0.5
rotation factor,flow factor	0.5,1.0
rotation factor,flow factor	1.0,0.5
rotation factor,flow factor	1.0,1.0

The best conducted with rotation factor weight of 1.0, and an optical-flow factor weight of 1.0.



3.3 Initializing and Updating Optimizer Step Size η

Initializing η In this experiments, I checked for different values in order to find the optimal for the APGD optimizer. I run 50 iteration of APGD with Momentum. In figure 2, the evolution of the best evaluation loss is plotted for each $\eta 0.05, 0.10, 0.15, 0.2, 0.75$.

Updating η We start with step size. With 50 iterations per optimizing phase, we identify 50 checkpoints at which the algorithm decides whether it is necessary to halve the current step size.
We have two conditions:

1. $\sum_{i=w_{j-1}}^{w_j-1} \mathbf{1}_{f(x^{(i+1)})>f(x^{(i)})} < \rho \cdot (w_j - w_{j-1}),$
2. $\eta^{(w_{j-1})} \equiv \eta^{(w_j)}$ and $f_{\max}^{(w_{j-1})} \equiv f_{\max}^{(w_j)},$

Condition 1: counts in how many cases since the last checkpoint w[j1] the update step has been successful in increasing

the objective function f. If this happened for at least a fraction of the total update steps, then the step size is kept as the optimization is proceeding properly (I use = 0.75).
Condition 2: holds true if the step size was not reduced at the last checkpoint and there has been no improvement in the best found objective value since the last checkpoint. This prevents getting stuck in potential cycles.

4 Results and discussion

To summarize, The chosen attack is APGD with step size = 0.15, using momentum = 0.75,early stopping = 20, rotation and optical-flow factors of 1.0. I use train-eval split for 5 folds to adjust the initial step size, since I find the hyper-parameter the most significant one.

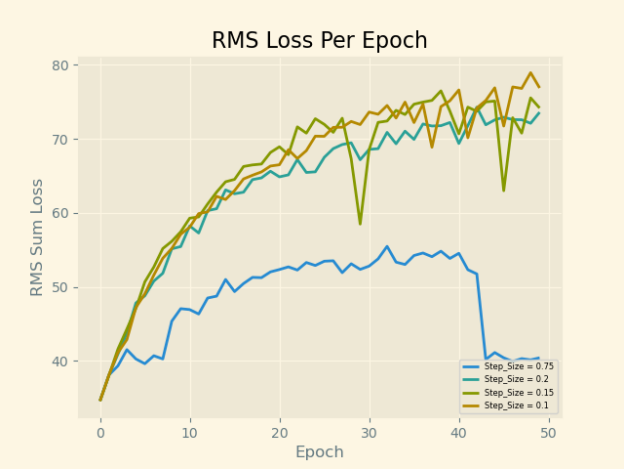


Figure 3: Initial Attack with different Step Sizes

We can see that the most suitable Step Size to start with is 0.15. Note that Step Size 0.75 was set for assuring step-size range.

Optimizer	Initial Step Size	Mean Loss of Adversarial RMS
APGD	0.05	0.419
APGD	0.1	3.604769
APGD	0.15	3.798201
APGD	0.2	3.157261
APGD	0.75	2.264978

Acknowledgments

This project is part of the final assignment for the course "Deep Learning on Computational Accelerators - 236781". Most of the code provided by the course staff, and all rights reserved.
Original Paper - "Physical Passive Patch Adversarial Attacks on Visual Odometry" - https://arxiv.org/pdf/2207.05729.pdf

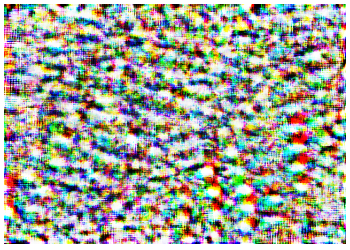


Figure 4: The Final Universal Perturbation Patch