Abstract

Deep learning neural networks are shown to be susceptible adversarial attacks by small perturbations to their input. Most common of these known as Projected Gradient Descent (PGD). In this work, we improve upon PGD in a visual odometry setting by utilizing more sophisticated optimizers that generalize better to universal attacks and make use of an adaptive step size informed by a sparsity heuristic. We combine the adaptive step size and Momentum Iterative Fast Gradient Descent Method (MI-FGSM) into a superior Momentum Adaptive PGD (M-APGD) attack.

1. Introduction

Universal attacks on Visual Odometry (VO) models were introduced by [1] utilizing a PGD attack with an Iterative Fast Gradient Descent Method (I-FGSM) optimizer. We adapt variants of the I-FGSM optimizer such as the ones suggested by [2] and [3], which are shown to produce better generalizing universal attacks for the settings of image classification.

Additionally, we employ an adaptive step size optimization scheme similar to [4], which was originally proposed for L_1 norm. We find this scheme, however, to achieve great results in our settings of the L_∞ norm. An adaptive step size allows an attack to explore better perturbations instead of getting stuck in a local minima, and also allows faster convergence.

1.1. Task Formulation

In this subsection, we aim to formulate the task at hand by first providing background and the settings of the attack. Afterwards, we define a target criterion, which we wish to optimize for.

1.1.1. Attack Settings

[1] proposed the following settings for a VO attack:

$$I^P \triangleq A(I,P) \triangleq H(P) \star (I^1 - I^0) + I^0$$

Equation 1 – applying a perturbation to a patch in an image

$$P, I^0, I^1 \in \mathcal{I} \triangleq [0,1]^{3\times w\times h}, \qquad H: \mathcal{I} \to \mathcal{I}, \qquad A: \mathcal{I} \times \mathcal{I} \to \mathcal{I}$$

where $\mathcal{I}$ is a normalized RGB image space with width w and height h, I is an image sampled from a 3D scene, H is a linear transformation homography such that $H(P)$ maps P into a patch in a plane in the scene where I was sampled, as it is viewed from the same point and angle as I, that is, H represents a patch in $\mathcal{I}$. I^0, I^1 are the albedo images of I, I^0 is the same image as I except for the patch H represents, where I^0 is set to be black. I^1 is constructed similarly but with the patch being white instead. P describes a perturbation applied to the patch by A to finally result in $I^P = A(I,P)$. Note that $\star$ is the elementwise product for $\mathcal{I}$, and the difference $(I^1 - I^0)$ is the patch on the plane as viewed from I in white, with the rest of the image being black.

Further, consider a trajectory $\{\delta_t^{t+1}\}_{t=0}^{L-1}$ where images $\{I_t\}_{t=0}^{L}$ were taken from the scene:

$$T_t \in \mathbb{R}^3, \qquad R_t \in SO(3), \qquad \delta_t^{t+1} = \begin{bmatrix} T_t & R_t \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}$$

$$I_t \in \mathcal{I}$$

and the accumulated motion between time points $t_1 < t_2$ is therefore:

$$\delta_{t_1}^{t_2} = \prod_{k=t_1}^{t_2-1} \delta_k^{k+1}$$

Equation 2 – cumulative motion

Lastly, a VO model is given by:

$$VO: \mathcal{I} \times \mathcal{I} \to \mathbb{R}^3 \times SO(3)$$

1.1.2. Attack Formulation and Target Criterion

For our purposes, a single trajectory with its respective images is considered a sample, where the trajectory is the prediction, and the image sequence is the input. Therefore, our target model is defined by repeatedly applying a VO model on an image sequence $\{I_t\}_{t=0}^{L}$, with the output sequence $\left\{\hat{\delta}_t^{t+1}\right\}_{t=0}^{L-1}$ as the prediction of the target model:

$$\left\{\hat{\delta}_t^{t+1}\right\}_{t=0}^{L-1} = VO(\{I_t\}_{t=0}^{L}) \triangleq \{VO(I_t, I_{t+1})\}_{t=0}^{L-1}$$

Equation 3 – definition of the target model

A non-universal attack on one sample, aims to maximize a loss defined on the ground truth trajectory and prediction. Hence, a non-universal attack patch P_a for a loss $\ell(\cdot,\cdot)$ is:

$$P_a \in \arg\max_P \ell(VO(\{I_t^P\}_{t=0}^{L}), \{\delta_t^{t+1}\}_{t=0}^{L-1})$$

Equation 4 – non-universal attack perturbation

In a similar manner, a universal attack P_{ua} on a set of image sequences $\left\{\{I_{i,t}\}_{t=0}^{L_i}\right\}_{i=0}^{N-1}$ and ground truth motion trajectories $\left\{\{\delta_{i,t}^{t+1}\}_{t=0}^{L_i-1}\right\}_{i=0}^{N-1}$ is defined by the sum of the losses:

$$P_{ua} \in \arg\max_P \sum_{i=0}^{N-1} \ell\left(VO\left(\{I_{i,t}^P\}_{t=0}^{L_i}\right), \{\delta_{i,t}^{t+1}\}_{t=0}^{L_i-1}\right)$$

Equation 5 - universal attack perturbation

In this work, we are to find a universal adversarial patch, that aims to maximize RMS loss between the ground truth accumulated motion translation $T(\delta_0^L)$ and the predicted $T(\hat{\delta}_0^L)$:

$$\ell_{VO}\left(\left\{\hat{\delta}_t^{t+1}\right\}_{t=0}^{L-1}, \left\{\delta_t^{t+1}\right\}_{t=0}^{L-1}\right) = \left\|T(\delta_0^L) - T\left(\hat{\delta}_0^L\right)\right\|_2$$

Equation 6 – RMS loss

2. Methodology

2.1. Previous Work – PGD with I-FGSM

[1] suggests a Projected Gradient Descent (PGD) approach, with an iterative variation of the Fast Gradient Sign Method (FGSM), we will refer to the iterative method as I-FGSM as is in [2] and [3]. This approach will serve as a baseline we wish to improve upon, and can be summarized in the following algorithm:

```
Input   $\boldsymbol{VO}$: VO model
Input   $A$: Adversarial patch perturbation
Input   $X_{train}$: Training trajectories
Input:  $Y_{train}$: Training trajectories ground truth motions
Input   $X_{eval}$: Evaluation trajectories
Input   $Y_{eval}$: Evaluation trajectories ground truth motions
Input   $\ell_{train}$: Train loss function
Input   $\ell_{eval}$: Evaluation loss function
Input   $N_{train}$: Number of training trajectories
Input   $N_{eval}$: Number of evaluation trajectories
Input   $\alpha$: Ascent step size
1.  $P_{best} \leftarrow Uniform[0,1]$
2.  $Loss_{best} \leftarrow 0$
3.  Loop 1    For $k = 1$ to $K$:
4.              $g \leftarrow 0$
5.              Optimization Step
6.              Loop 1.1.   For $i = 1$ to $N_{train}$:
7.                            $\hat{y}_{train,i} \leftarrow VO\left(x_{train,i}^{P}\right)$
8.                            $g \leftarrow g + \nabla_P l_{train}\left(\hat{y}_{train,i}, y_{train,i}\right)$
9.
10.             $P \leftarrow P + \alpha \cdot sign(g)$
11.             $P \leftarrow clip(P, 0, 1)$
12.
13.             Evaluation Step
14.             Loop 1.2.   For $i = 1$ to $N_{eval}$:
15.                           $\hat{y}_{eval,i} \leftarrow VO\left(x_{eval,i}^{P}\right)$
16.                           $Loss \leftarrow Loss + l_{eval}\left(\hat{y}_{eval,i}, y_{eval,i}\right)$
17.
18.             If $Loss > Loss_{best}$:
19.                           $P_{best} \leftarrow P$
20.                           $Loss_{best} \leftarrow Loss$
21.
22. Return $P_{best}$
```

Algorithm 1 – PGD with I-FGSM

where for a sequence of images $x = \{I_t\}_{t=0}^{L}$ we denote:

$$x^P \triangleq A(x,P) \triangleq \{I_t^p\}_{t=0}^{L} = \{A_t(I_t,P)\}_{t=0}^{L}$$

Equation 7

[1] suggests the evaluation criterion to be chosen as $\ell_{eval} = \ell_{RMS}$, a smoother version of ℓ_{VO}:

$$\ell_{RMS} \triangleq \sum_{l=0}^{L-1} \ell_{VO}\left(\{\hat{\delta}_t^{t+1}\}_{t=0}^{l-1}, \{\delta_t^{t+1}\}_{t=0}^{l-1}\right) = \sum_{l=0}^{L-1} \left\| T\left(\delta_0^l\right) - \; T\left(\hat{\delta}_0^l\right) \right\|_2$$

It is also suggested by [1] to use a different, even smoother loss for the training criterion ℓ_{train}, however, we focus in our work on the optimization scheme and evaluating different variants of I-FGSM, as most of the literature focuses on these.

2.2. Optimization

As already mentioned, Algorithm 1 makes use of I-FGSM for updating the perturbation in each iteration, other works suggested other variations of FGSM, such as Momentum I-FGSM (MI-FGSM) [3], Adam I-FGSM (AI-FGSM) and Adaptive Belief I-FGSM (AB-FGSM) [2]. We also experiment with a modified AB-FGSM technique, which we will refer to as M-AB-FGSM in the scope of this work.

2.2.1. MI-FGSM

MI-FGSM is the simplest modification to PGD we discuss in section 2.2, despite that, our experiments show it may well be the most effective. We discuss these results further in section 4. This method saves the accumulated gradient from previous iterations with a decay factor μ, and together with the new gradient defines a new descent direction as described in algorithm 2:

Input	$\boldsymbol{VO}$**:** VO model	
Input	A: Adversarial patch perturbation	
Input	X_{train}: Training trajectories	
Input:	Y_{train}: Training trajectories ground truth motions	
Input	X_{eval}: Evaluation trajectories	
Input	Y_{eval}: Evaluation trajectories ground truth motions	
Input	ℓ_{train}: Train loss function	
Input	ℓ_{eval}: Evaluation loss function	
Input	N_{train}: Number of training trajectories	
Input	N_{eval}: Number of evaluation trajectories	
Input	α: Ascent step size	
Input	μ: Momentum decay factor	
1.	$P_{best} \leftarrow Uniform[0,1]$	
2.	$Loss_{best} \leftarrow 0$	
3.	$g_{acc} \leftarrow 0$	
4.	Loop 1	For $k = 1$ to K:
5.		$g \leftarrow 0$

```
6.      Optimization Step
7.      Loop 1.1.  For $i = 1$ to $N_{train}$:
8.                     $\hat{y}_{train,i} \leftarrow VO(x^P_{train,i})$
9.                     $g \leftarrow g + \nabla_P l_{train}(\hat{y}_{train,i}, y_{train,i})$
10.
11.     $g_{acc} \leftarrow \mu \cdot g_{acc} + \frac{g}{\|g\|_1}$
12.     $P \leftarrow P + \alpha \cdot sign(g_{acc})$
13.     $P \leftarrow clip(P, 0,1)$
14.
15.     Evaluation Step
16.     Loop 1.2.  For $i = 1$ to $N_{eval}$:
17.                    $\hat{y}_{eval,i} \leftarrow VO(x^P_{eval,i})$
18.                    $Loss \leftarrow Loss + l_{eval}(\hat{y}_{eval,i}, y_{eval,i})$
19.
20.     If $Loss > Loss_{best}$:
21.                    $P_{best} \leftarrow P$
22.                    $Loss_{best} \leftarrow Loss$
23.
24. Return $P_{best}$
```
Algorithm 2 – PGD with MI-FGSM

Involving momentum in the update step serves the same purpose as in common optimization problems. Momentum can help with faster convergence and avoiding local minima better than I-FGSM. So, we find experimentally.

2.2.2. AB-FGSM and M-AB-FGSM

AB-FGSM was proposed by [2] as an attack optimizer for image classification models. We adapt the optimize to our settings. The authors of [2] show their optimizer to produce better generalizing perturbations, in many cases beating AI-FGSM. That serves us as motivation to adapt the AB-FGSM optimizer to our task. We also suggest a modified version of the attack M-AB-FGSM. This modification simply normalizes the gradient before proceeding with the optimization step, as we found quite common in the FGSM base attacks literature. This fact also leads us to suspect the authors might have implemented our modification but did not mention it in their paper by mistake. In any case, as we find later in section 4, this modification greatly impacts the performance of the method.

Input	$\boldsymbol{VO}$**:** VO model
Input	A: Adversarial patch perturbation
Input	X_{train}: Training trajectories
Input:	Y_{train}: Training trajectories ground truth motions
Input	X_{eval}: Evaluation trajectories
Input	Y_{eval}: Evaluation trajectories ground truth motions
Input	ℓ_{train}: Train loss function
Input	ℓ_{eval}: Evaluation loss function
Input	N_{train}: Number of training trajectories

```
Input  $N_{eval}$: Number of evaluation trajectories
Input  $\alpha$: Ascent step size
Input  $\beta_1, \beta_2$: Exponential decay rates
Input  $\varepsilon$: Denominator stability parameter.
1.  $P_{best} \leftarrow Uniform[0,1]$
2.  $Loss_{best} \leftarrow 0$
3.  $m, s \leftarrow 0$
4.
5.  Loop 1    For $k = 1$ to $K$:
6.              $g \leftarrow 0$
7.              Optimization Step
8.              Loop 1.1.   For $i = 1$ to $N_{train}$:
9.                            $\hat{y}_{train,i} \leftarrow VO\left(x_{train,i}^{P}\right)$
10.                           $g \leftarrow g + \nabla_P l_{train}\left(\hat{y}_{train,i}, y_{train,i}\right)$
11.
12.             $g \leftarrow \frac{g}{\|g\|_1}$        (M-AB-FGSM only)
13.             $\gamma \leftarrow \sum_{i=0}^{k-1} \frac{\sqrt{1-\beta_2^i}}{1-\beta_1^i}$
14.             $m \leftarrow \beta_1 m + (1-\beta_1) g$
15.             $s \leftarrow \max\{s, \quad \beta_2 s + (1-\beta_2)(g-m)^2\}$
16.             $\hat{m} \leftarrow \frac{m}{1-\beta_1^k}$
17.             $\hat{s} \leftarrow \frac{s+\varepsilon}{1-\beta_2^k}$
18.             $P \leftarrow P + \frac{\alpha}{\gamma} \cdot sign\left(\frac{\hat{m}}{\sqrt{\hat{s}} + \epsilon}\right)$
19.             $P \leftarrow clip(P, 0,1)$
20.
21.             Evaluation Step
22.             Loop 1.2.   For $i = 1$ to $N_{eval}$:
23.                           $\hat{y}_{eval,i} \leftarrow VO\left(x_{eval,i}^{P}\right)$
24.                           $Loss \leftarrow Loss + l_{eval}\left(\hat{y}_{eval,i}, y_{eval,i}\right)$
25.
26.             If $Loss > Loss_{best}$:
27.                           $P_{best} \leftarrow P$
28.                           $Loss_{best} \leftarrow Loss$
29.
30. Return $P_{best}$
```

Algorithm 3 – AB-FGSM without line 12, and M-AB-FGSM with.

2.3. Adaptive PGD (APGD)

[4] suggests a PGD attack with adaptive step size α that changes throughout the attack based on a sparsity heuristic. The sparsity heuristic makes use of what is commonly known as the L_0 norm – although it technically is not a norm. The L_0 norm counts the non-zero elements of a tensor.

Originally, APGD was proposed for attacks where the perturbation is applied directly to an input image for models that receives one image as an input, rendering the method unsuitable for our task in a direct manner.

Follows, is an APGD algorithm adapted to our task, which also supports the option for utilizing momentum. We will refer to the momentum variation as M-APGD:

Input	$\boldsymbol{VO}$: VO model			
Input	A: Adversarial patch perturbation			
Input	X_{train}: Training trajectories			
Input:	Y_{train}: Training trajectories ground truth motions			
Input	X_{eval}: Evaluation trajectories			
Input	Y_{eval}: Evaluation trajectories ground truth motions			
Input	ℓ_{train}: Train loss function			
Input	ℓ_{eval}: Evaluation loss function			
Input	N_{train}: Number of training trajectories			
Input	N_{eval}: Number of evaluation trajectories			
Input	α_0: Initial ascent step size			
Input	α_{min}: Minimal allowed step size			
Input	M: set of checkpoint indices			
Input	μ: Momentum decay factor (M-APGD only)			
Input	κ: Sparsity ratio threshold			
1.	$P_{best} \leftarrow Uniform[0,1]$			
2.	$Loss_{best} \leftarrow 0$			
3.	$q \leftarrow q_0$			
4.	$\alpha \leftarrow \alpha_0$			
5.	$g_{acc\ best},\ g_{acc} \leftarrow 0$ (M-APGD only)			
6.	Loop 1	For $k = 1$ to K:		
7.		$g \leftarrow 0$		
8.		<u>Optimization Step</u>		
9.		Loop 1.1.	For $i = 1$ to N_{train}:	
10.			$\hat{y}_{train,i} \leftarrow VO\left(x^P_{train,i}\right)$	
11.			$g \leftarrow g + \nabla_P l_{train}\left(\hat{y}_{train,i}, y_{train,i}\right)$	
12.				
13.		$g_{acc} \leftarrow \mu \cdot g_{acc} + \frac{g}{\|g\|_1}$		(M-APGD only)
14.		$P \leftarrow P + \alpha \cdot sign(g)$		
15.		$P \leftarrow clip(P, 0,1)$		
16.				
17.		<u>Sparsity and Step Size Adjustment</u>		
18.		If $k \in M$:		

```
19.                 $q_{prev} \leftarrow q$
20.                 $q \leftarrow \frac{\|P_{best} - P\|_0}{1.5 \cdot d}$
21.                 $\alpha \leftarrow \begin{cases} \max\left\{\alpha_{min}, \frac{\alpha}{1.5}\right\}, & \frac{q}{q_{prev}} \geq \kappa \\ \alpha_0, & otherwise \end{cases}$
22.
23.                 If $\alpha = \alpha_0$:
24.                     $P \leftarrow P_{best}$
25.                     $g_{acc} \leftarrow g_{acc\ best}$        (M-APGD only)
26.
27.         Evaluation Step
28.         Loop 1.2.   For $i = 1$ to $N_{eval}$:
29.                 $\hat{y}_{eval,i} \leftarrow VO\left(x^P_{eval,i}\right)$
30.                 $Loss \leftarrow Loss + l_{eval}\left(\hat{y}_{eval,i}, y_{eval,i}\right)$
31.
32.         If $Loss > Loss_{best}$:
33.                 $P_{best} \leftarrow P$
34.                 $Loss_{best} \leftarrow Loss$
35.                 $g_{acc\ best} \leftarrow g_{acc}$ (M-APGD only)
36.
37.     Return $P_{best}$
```

Algorithm 4 – APGD and M-APGD attacks

We note that this algorithm was originally targeted at L_1 norm, however, we found experimentally it also performs well for our purpose.

To produce the M-APGD variant, we simply plant the steps specific to the MI-FGSM optimizer in the APGD algorithm, save the momentum at the iteration the best perturbation was achieved at so far, and continue with the momentum we found with the best perturbation whenever we have to backtrack at a checkpoint.

3. Experiments And Implementation

We are supplied with 5 folders, each containing 10 trajectories. More information about that data is available in [1]. We put aside the 5th dataset as a test set and only use it after we have experimented with all our ideas, by that, we hope to prevent leakage from the test set and prevent our own bias from selecting one method over the other.

For tuning hyperparameters we use 4-fold cross validation on the first four sets $0, 1, 2, 3$

3.1. Experiments Tools Implementation

We implemented a tool in the module run_experiments.py that allows one to run all the experiments below using a CLI, another tool in run_experiments_evlauation.py to evaluate our perturbations on the test set we set aside, and a tool called plot.py to help plot results and extract other information from our experiments easily. The core of these tools we implemented in a package called experiments. This package implements classes to load experiment data and export the results in two modules called loaders.py and exporters.py.

3.2. Evaluation

We evaluate our results using two metrics, the first metric is used to evaluate different hyper parameters for a chosen attack employing the RMS loss ℓ_{RMS}, while the second is chosen to evaluate the attacks themselves and employs the VO loss ℓ_{VO}:

$$M(\ell) \triangleq \frac{\ell^{adv} - \ell^{clean}}{\ell^{clean}} = \frac{\ell^{adv}}{\ell^{clean}} - 1, \qquad M_{RMS} \triangleq M(\ell_{RMS}), \qquad M_{VO} \triangleq M(\ell_{VO})$$

$$\ell^{adv} = \sum_{i=1}^{N} \ell\left(\{\hat{\delta}_t^{t+1}\}_{t=0}^{L-1}, \{\delta_t^{t+1}\}_{t=0}^{L-1}\right), \qquad \ell^{clean} \sum_{i=1}^{N} \ell\left(VO\left(\{I_{i,t}^{0}\}_{t=0}^{L_i}\right), \{\delta_{i,t}^{t+1}\}_{t=0}^{L_i-1}\right)$$

A metric $M(\ell)$ over a loss ℓ describes the improvement in loss relative to the performance of the albedo I^0 as an attack, thereby providing a form of a normalized metric better suitable for comparing between different validation sets.

Using M_{RMS} to evaluate the performance on the validation sets used for tuning hyperparameters only makes sense, since the optimization schemes we suggested all optimize ℓ^{adv}, and maximizing that is equivalent to maximizing M_{RMS}. That, however, differs from maximizing M_{VO}, which is the metric our submitted perturbation would be assessed by and therefore makes sense to perform a final evaluation on our proposed methods.

3.3. Optimization

All optimizers in this section were implemented in pgd.py under the attack folder, under PGD class. The optimization flow is the same for all the optimizers, so we only change the implementation of the update step based on the optimizer chosen via the CLI.

For all experiments discussed in this section, we fix the step size and number of iterations:

$$K = 20, \qquad \alpha = \frac{1}{K} = 0.05$$

3.3.1. Tuning MI-FGSM Decay Parameter μ

For this experiment we vary the decay factor μ in the range $[0,2]$ with jumps of 0.2 starting from $\mu = 0$.

3.3.2. Tuning AB-FGSM Parameters β_1, β_2:

We tune the decay parameters β_1, β_2 for both AB-FGSM and M-AB-FGSM

$$(\beta_1, \beta_2) \in R \times R, \qquad R = \{0, 0.2, 0.4, 0.6, 0.8, 0.9, 0.99, 0.999\}$$

3.4. APGD

We implement the APGD and M-APGD attacks in apgd.py under the attack folder, under APGD class, which itself inherits from PGD. Choosing between PGD and M-APGD is done by choosing an optimizer I-FGSM or M-IFGSM respectively.

[4] suggests a sparsity threshold $\kappa = 0.95$, we experiment with different thresholds:

$$\kappa \in \{0.85, 0.9, 0.95\}$$

For this experiment we fix $\mu = 0.6$ when using M-APGD, as we found it to be optimal from the previous experiments. We fix $\alpha_0 = 1, q_0 = 0.2$ as suggested by [4]. For convenience, we list all the fixed parameters:

$$K = 120, \qquad \mu = 0.6, \qquad \alpha_0 = 1, \qquad q_0 = 0.2, \qquad M = \{5, 10, \dots 115\}$$

4. Results and Discussion

In subsections 4.1, 4.2 and 4.3, we discuss the results of the different optimization schemes we proposed, evaluated by the metric M_{RMS}, which we also use to choose the optimal hyperparameters to evaluate the method using M_{VO} in section 4.3.

4.1. Optimization

From the experiments we conducted on the validation sets, according to M_{RMS}, we find MI-FGSM performed best with M-AB-FGSM as a second, while I-FGSM and AB-FGSM performed worst. Additionally, we find the decay factor μ of MI-FGSM generalizes quite well across different choices of the validation set, as seen in Figure 1, in contrast to M-AB-FGSM and AB-FGSM for which we find only β_2 to generalize well.

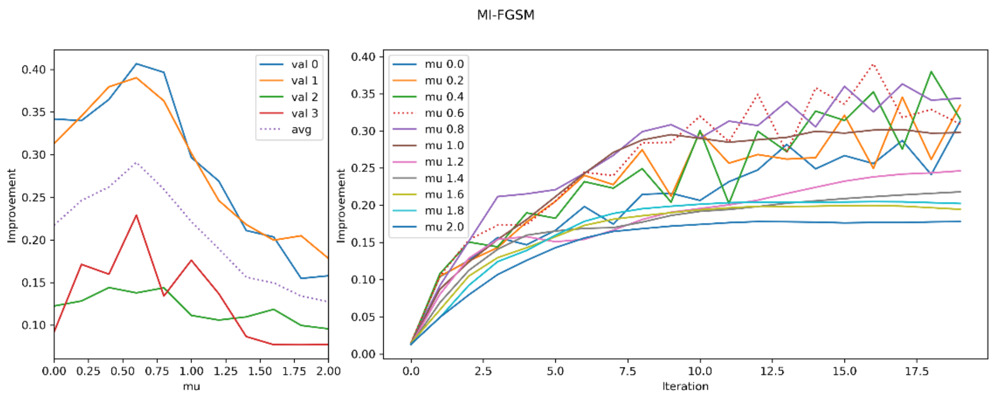


Figure 1:

Left: Performance of MI-FGSM according to validation set (max over all iterations) according to M_{RMS} improvement metric

Right: M_{RMS} improvement metric track along optimization for different decay factors μ on validation set 1

We found the optimal β_1 for AB-FGSM to be $\beta_1^{opt} = 0.2, 0.2, 0.4, 0.4$ for validation sets 0,1,2,3 respectively and for M-AB-FGSM we found $\beta_1^{opt} = 0, 0.4, 0.6, 0.6$ optimal. As for the other decay parameter, we found $\beta_2 = 0.999$ optimal consistently. We also note that for β_2 we see a spike across all the validation sets as we get closer to $\beta_2 = 1$, which matches the findings of [2], unlike β_1 which gives us actually results in a great dip as it approaches $\beta_1 = 1$. We can see these phenomena in Figure 2 and Figure 3.

To compare the performance of different optimizers, we take hyperparameters that performed best for most validation set choices. That is, for MI-FGSM we take $\mu = 0.6$, for AB-FGSM we take $\beta_1 = 0.4$, $\beta_2 = 0.999$, and for M-AB-FGSM we take $\beta_1 = 0.4$, $\beta_2 = 0.999$. We discuss these results later in section 4.3.

We summarize the chosen optimal hyperparameters in Table 1.

Optimizer:	I-FGSM	MI-FGSM	AB-FGSM	M-AB-FGSM
Optimal params	-	$\mu = 0.6$	$\beta_1 = 0.4, \beta_2 = 0.999$	$\beta_1 = 0.6, \beta_2 = 0.999$

Table 1 – Optimal general parameters found for each optimizer

AB-FGSM

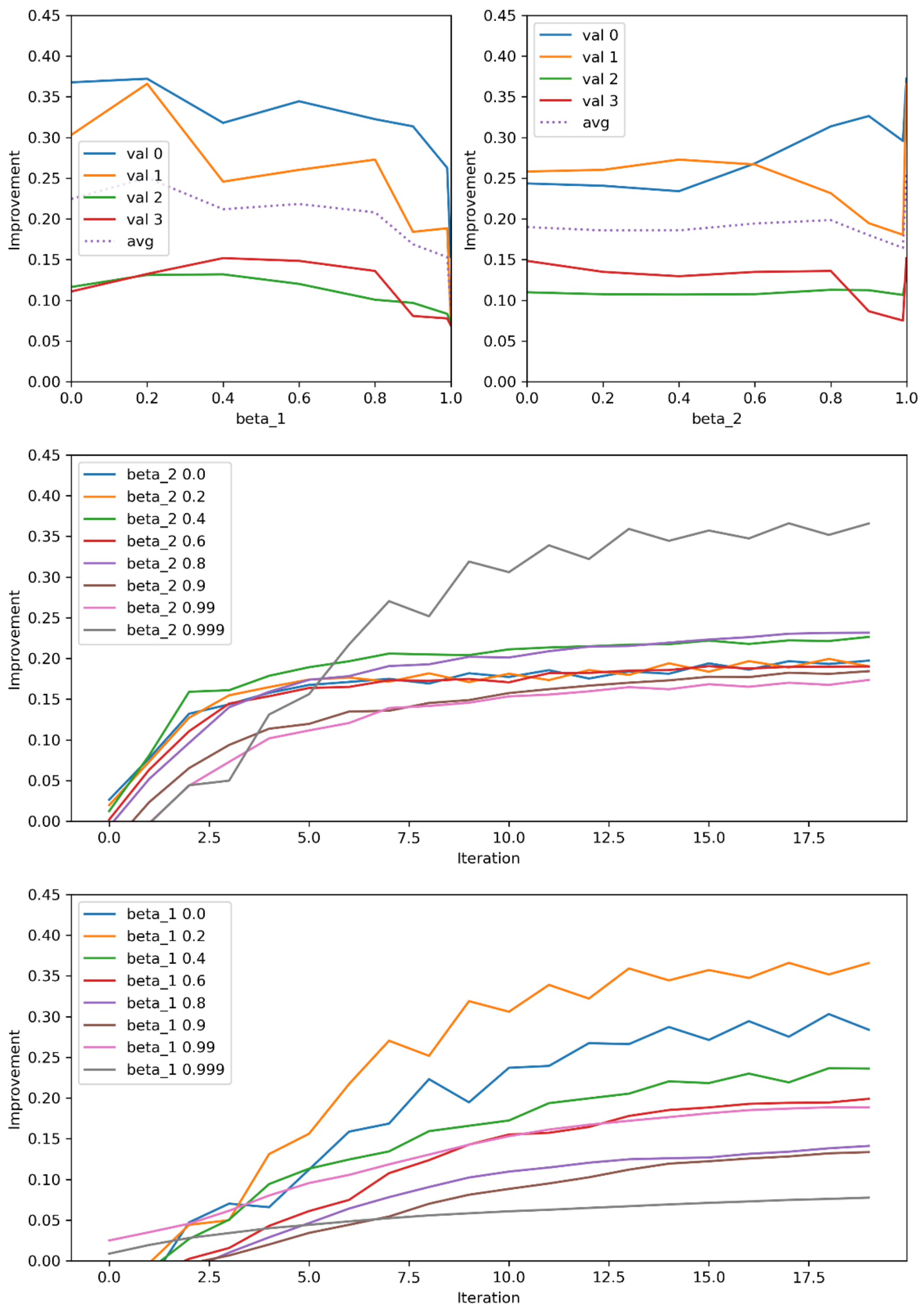
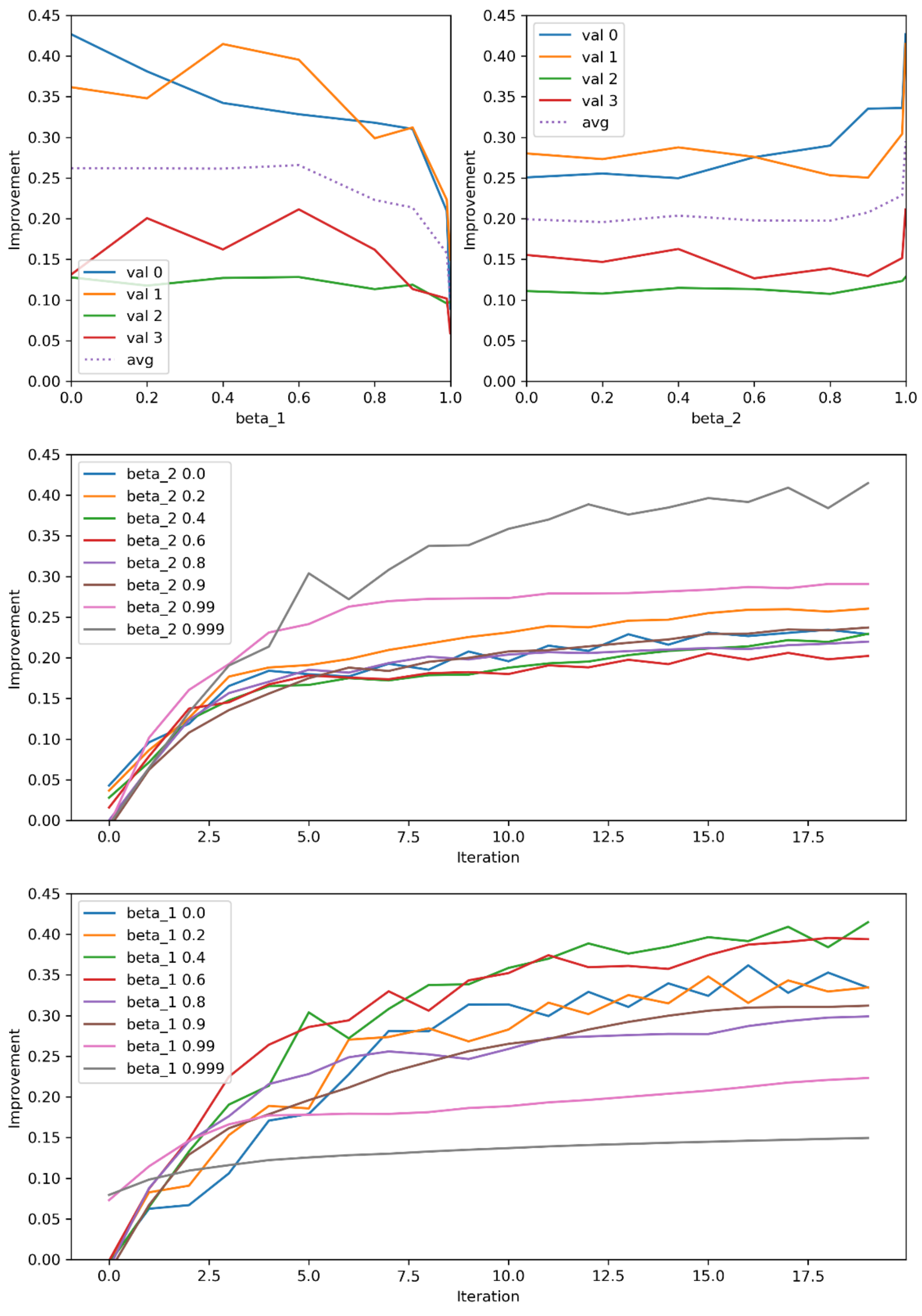


Figure 2: Top graphs with the horizontal axis named beta_1 and beta_2 show the maximal improvement according to M_{RMS} of a hyperparameter β_i over all the other values of the other hyperparameter. Middle graph shows M_{RMS} track over the course of optimization for $\beta_1 = 0.2$ and varying values of β_2. Bottom path shows the same as the middle with $\beta_2 = 0.999$. Middle and bottom graphs concern validation set 1.

M-AB-FGSM



0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Improvement
beta_1
beta_2
val 0
val 1
val 2
val 3
avg
0.0
0.2
0.4
0.6
0.8
1.0
Iteration
2.5
5.0
7.5
10.0
12.5
15.0
17.5
beta_2 0.0
beta_2 0.2
beta_2 0.4
beta_2 0.6
beta_2 0.8
beta_2 0.9
beta_2 0.99
beta_2 0.999
beta_1 0.0
beta_1 0.2
beta_1 0.4
beta_1 0.6
beta_1 0.8
beta_1 0.9
beta_1 0.99
beta_1 0.999

Figure 3: Top graphs with the horizontal axis named beta_1 and beta_2 show the maximal improvement according to M_{RMS} of a hyperparameter β_i over all the other values of the other hyperparameter. Middle graph shows M_{RMS} track over the course of optimization for $\beta_1 = 0.4$ and varying values of β_2. Bottom path shows the same as the middle with $\beta_2 = 0.999$. Middle and bottom graphs concern validation set 1.

4.2. APGD

We evaluate APGD and M-APGD using M_{RMS} on the validation sets. While the sparsity ratio threshold κ seems to not have a notable effect on M-APGD on average, we found $\kappa = 0.85$ performed best on this metric for APGD on average from the range of κ we experimented with.

As expected, M-APGD showed better M_{RMS} of with an average of about 0.395 than APGD with an average of about 0.288 when taking $\kappa = 0.85$. We choose this value of κ since APGD performs with it, and it doesn't really affect M-APGD. We also evaluate APGD using this value later in section $(placeholder)$

	Val\\κ	0.85	0.9	0.95
	0	0.453013	0.398834	0.37786
	1	0.330057	0.336741	0.347092
	2	0.159668	0.147484	0.126596
	3	0.212642	0.221076	0.200622
Avg.		0.288845	0.276034	0.263043

Table 2 – APGD (I-FGSM) performance with different sparsity ratio κ thresholds on validation sets (max over all iterations)

	Val\\κ	0.85	0.9	0.95
	0	0.654478	0.56783	0.716718
	1	0.475076	0.501378	0.433229
	2	0.156401	0.16496	0.155127
	3	0.293242	0.35292	0.287458
Avg.		0. 3948	0.396772	0. 39813

Table 3 – M-APGD (MI-FGSM) performance with different sparsity ratio κ thresholds on validation sets (max over all iterations)

4.3. Evaluation of M_{RMS} and M_{VO} on Validation Sets

In this subsection we compare our different optimization methods and study the effect of the adaptive step size computed by the sparsity heuristic using both the validation set used to choose the best perturbation (M_{RMS}) and also the test set we set aside (M_{VO}).

	M_{RMS}					
	APGD		PGD			
Val	I-FGSM	MI-FGSM	I-FGSM	M-IFGSM	AB-FGSM	M-AB-FGSM
0	0.453013	0.654478	0.341933	0.406769	0.317989	0.328406
1	0.330057	0.475076	0.31288	0.390338	0.236549	0.395483
2	0.159668	0.156401	0.122454	0.137877	0.131905	0.128229
3	0.212642	0.293242	0.091801	0.229079	0.151842	0.211471
Avg.	0.288845	0.394799	0.217267	0.291016	0.209571	0.265897
Max.	0.453013	0.654478	0.341933	0.406769	0.317989	0.395483

Table 4 – Performance of all attack methods with respective optimizers on validation sets (max over all iterations) using M_{RMS}

As discussed earlier, for the PGD attack, in Table 4 we see that I-FGSM and AB-FGSM behave similarly on average and give the worst results. Then follows M-AB-FGSM, which achieves higher improvements on average though still lower than what MI-FGSM with the best results achieves. We also see that APGD achieves only slightly less improvement on average to PGD with MI-FGSM, which would lead us believe that an adaptive step size can greatly help incur a higher loss. M-APGD however, beats all other attacks by a great margin and incurs a relative loss of about as twice as the baseline PGD with I-FGSM. Now that we have discussed the performance of the attacks in terms of M_{RMS} on the validation sets, it is important to make sure these attack generalize well on unseen data and the M_{VO} metric.

	M_{VO}					
	APGD		PGD			
Val	I-FGSM	MI-FGSM	I-FGSM	M-IFGSM	AB-FGSM	M-AB-FGSM
0	0.730767	1.166526	0.519947	0.680051	0.514768	0.562593
1	0.574102	0.989122	0.469947	0.516584	0.354015	0.672454
2	1.035133	0.185787	0.398181	0.881289	0.479509	0.439579
3	0.163288	0.838694	0.083258	0.531395	0.496497	0.538319
Avg.	0.625822	0.795032	0.367833	0.65233	0.461197	0.553236
Max.	1.035133	1.166526	0.519947	0.881289	0.514768	0.672454

Table 5 - Performance of all attack methods with respective optimizers on test set using M_{VO} with best perturbation found on validation set

Analysing Table 5, one concludes that the results regarding M_{RMS} generalize to M_{VO} with the exception of AB-FGSM which seems to perform better on average than I-FGSM despite using the same perturbations. M-AB-FGSM seems to still benefit from our modification and also seems to be most consistent amongst different choices of validation sets. M-APGD still beats all other attacks with good margin. It is also useful in this case to consider the maximal improvement of an attack method since we could choose one perturbation that generalizes well on the test set.

In conclusion, our hybrid method of combining momentum with APGD seems to greatly enhance the performance of the attack. Furthermore, all other methods we experimented with seem to have improved upon the baseline of PGD with I-FGSM, at least on average considering the M_{VO} metric. While we did not experiment with changing the training and evaluation of the attacks as suggested in [1], we think doing so may achieve better results, especially when considering the rotational loss in a direct manner. We also believe the improvement of the loss is orthogonal to the improvement of the methods we evaluated. Thus, we leave that for future work and others to experiment.

References

[1] "Project guideline: Universal adversarial perturbations on visual odometry systems".

[2] "AB-FGSM: AdaBelief optimizer and FGSM-based approach to generate adversarial examples," *Journal of Information Security and Applications,* vol. 68, 08 2022.

[3] "Discovering Adversarial Examples with Momentum," 2018.

[4] "Mind the box: l1-APGD for sparse adversarial attacks on image classifiers".