1. Abstract

Adversarial attacks on deep learning networks involve generating a slightly perturbed version of the input data so that the network's classifier will be fooled. Universal perturbations are perturbations that achieve this feat across a set of inputs.
Visual odometry models aim to estimate the position and orientation between two viewpoints.
Patch adversarial perturbation attacks on Visual odometry models can be used against autonomous vehicles.
In this project we were tasked to produce the best universal perturbation to attack a given VO model.

The VO model produces estimated; motion, rotation and optical flow based on a given trajectory.
Our task was to integrate the rotation and optical flow into the patch adversarial perturbation attack so that the estimated position and orientation produced by the VO will be as far as possible from the actual ground truth.

During our research we've compared several different ways to evaluate the loss of the optical flow, the End-point error gave us the best results.
We've experimented with different formulas for the rotation evaluation.
We used cross-validation to determine the best model for the task.

2. Introduction

In our research, we've encountered several relevant articles.
To determine the best way to evaluate the loss of the optical flow we've referred to two articles:
The first article, "Optical Flow estimation using Deep Learning", written by Maxim Kuklin (Xperience.AI) **[1]**, suggested the approach of the End-point error loss, this is a compelling approach since the articles state that several state-of-the-art optical flow estimation models use this.
The second article regarding optical flow evaluation was: "Optical Flow Estimation Using Total Least Squares Variants", written by Maria A. De Jesus & Vania V. Estrela (UFF) **[3]**, which suggested the use of several different functions to evaluate the optical flow loss.
When we tried to determine the best way to evaluate the loss of the quaternion rotation we found only one article addressing this matter.
"Geometric loss functions for camera pose regression with deep learning", written by Alex Kendall & Roberto Cipolla (University of Cambridge) **[2]**,
This article also speaks about combining several loss functions into a single loss criteria using weighted sum.
This is no easy task as optimization of the weight factors, according to the paper written by Alex Kendall & Roberto Cipolla's **[2]**: "Following this intuition it is possible to fine-tune β using grid search. For the indoor scenes, it was between 120 to 750, and for outdoor scenes between 250 to 2000. This is an expensive task in practice, as each experiment **can take days to complete**.".
Overall, in our project, we have to deal with 4 weight factor parameters, and optimize all of them together. As mentioned in the article,this could take several weeks, for this project we could not afford to spend weeks on the optimization of these parameters.
So, our solution to this difficulty is to take a wide sample of numbers, in big leaps, between 0.5 to 2500, and to find the best fit values range we can optimize in a reasonable time.

3. Methods

The original approach given to us by the course's staff, was to compute the loss of the neural network was dependent only on the deviation in the physical translation. In our work we've altered the loss function so that it will take advantage of the rotation and optical flow elements produced by the VO model as well.

3.1

Denote:

$OF - ground\ truth\ optical\ flow.$

$\widehat{OF} - VO\ estimated\ optical\ flow.$

$QR - ground\ truth\ quaternion\ rotation.$

$\widehat{QR} - VO\ estimated\ quaternion\ rotation.$

$M - ground\ truth\ motions.$

$\widehat{M} - VO\ estimated\ motions.$

$l_{OF} = Loss_{OPTICAL\ FLOW}(OF, \widehat{OF}).$

$l_{QR} = Loss_{QUAT\ ROTATION}(QR, \widehat{QR}, \gamma).$

$l_{T} = Loss_{T}(M, \widehat{M}).$

$l_{TTARGET} = Loss_{T}(M, \widehat{M}).$

$l = Loss_{WEIGHTED\ SUM}(l_{OF}, l_{QR}, l_{t}, l_{TTARGET},).$

$\gamma - quat\ rotation\ norm.$

$\beta_{quat\ rotation} - quat\ rotation\ weight\ factor.$

$\beta_{optical\ flow} - optical\ flow\ weight\ factor$

$\beta_{t} - t\ weight\ factor.$

$\beta_{target} - target\ t\ \ weight\ factor$

3.2

First we've tried to evaluate the value of $Loss_{OPTICAL\ FLOW}$, so that we can later add it to the loss function.

We've decided upon three different loss formulas for the optical flow, so that we can cross validate and choose the best approach:

The first formula is used in both RAFT and FlowNet is architecture which are known motion estimations models based on optical flow as described in [1]:

$$l_{OF} = EPE_{loss}(OF, \widehat{OF}) = ||OF - \widehat{OF}||_2$$
$$= \sqrt{(\Delta x_{OF} - \Delta x_{\widehat{OF}})^2 + (\Delta y_{OF} - \Delta y_{\widehat{OF}})^2}$$

The second formula is the known MSELoss formula which was used in many optical flow estimation models as described in [3] we've used the MSELoss built in implementation by pytorch [5]:

$$l_{OF} = MSE_{loss}(OF, \widehat{OF}) = \frac{1}{N}\sum_{i=0}^{N}(OF - \overline{OF})^2$$

The third and final approach that we wanted to try is the RMSELoss formula, simply the square root of the MSELoss formula. We were curious to test this approach to see if it will produce better or the same results as the MSELoss formula:

$$l_{OF} = RMSE_{loss}(OF, \widehat{OF}) = \sqrt{\frac{\sum_{i=0}^{N}(OF - \overline{OF})^2}{N}}$$

3.3

Now in order to evaluate $Loss_{QUAT\ ROTATION}$ we've used a quatrinon regression loss formula as described in [2] the output of the formula is then normalized using a γ norm. As stated in the article the best values for the γ are 1 or 2. In part 4.2 we've cross validated these options to find the optimal model for our goal.

The formula:

$$l_{QR} = Loss_{QUAT\ ROTATION}(QR, \widehat{QR}, \gamma) = ||QR - \frac{\widehat{QR}}{||\widehat{QR}||}||_\gamma$$

3.4

Finally we must define the Loss function of our model to include all of the relevant elements.

The loss function must cause our model to learn simultaneously all of the elements so therefore as described in article [1] a recommended approach is a weighted sum of all the elements:

$$l = Loss_{WEIGHTED\ SUM}(l_{OF}, l_{QR}, l_t, l_{TTARGET}) =$$
$$\beta_{optical\ flow} \cdot l_{OF} + \beta_{quat\ rotation} \cdot l_{QR} + \beta_t \cdot l_T + \beta_{target} \cdot l_{TTARGET}$$

4. Implementations and Experiments

As discussed in 3.4 we've implemented all the optional loss formulas.
We've implemented a cross validation to test the different approaches to the loss formula, and ultimately chose the best model for our given task.
We've split randomly (inorder to prevent overfitting) the given data into train and validate sets with a 6:4 ratio;
So the model can learn 3 datasets and then evaluate its performance on the remaining 2.
The objective of this process was to determine the following parameters:

$l_{OF} = Loss_{OPTICAL\ FLOW}(OF, \widehat{OF})$.

$\gamma - quat\ rotation\ norm.$

$\beta_{quat\ rotation} - quat\ rotation\ weight\ factor.$

$\beta_{optical\ flow} - optical\ flow\ weight\ factor$

$\beta_{t} - t\ weight\ factor.$

$\beta_{target} - target\ t\ \ weight\ factor$

$l_T, l_{TTARGET} - Loss_T$

We will determine the best model based upon the highest

$l = Loss_{WEIGHTED\ SUM}(l_{OF}, l_{QR}, l_t, l_{TTARGET},)$

During the whole process we used a fixed step size of 0.02

All tests ran with a random seed

4.1

Firstly we wanted to find the $Loss_{OPTICAL\ FLOW}$ with the best results so we cross validated the different formulas as stated in 3.2 $[EPE_{loss}, MSE_{loss}, RMSE_{loss}]$

the rest of the parameters need to fixed for now:

$\gamma = 1$

$\beta_{quat\ rotation} = 1$

$\beta_{optical\ flow} = 1$

$\beta_{t} = 1$

β_{target} = 1

$l_{T,} l_{TTARGET} = RMS_{loss}$

we ran 50 epochs in each iteration.

4.2

Now fix $Loss_{OPTICAL\ FLOW}$ as a constant in the optimization and we will perform a new cross-validation to determine the best quat rotation norm γ for our model .

- $\gamma\ will\ be\ 1\ or\ 2$

again the fixed parameters are:

$\beta_{quat\ rotation} = 1$

$\beta_{optical\ flow} = 1$

$\beta_{t} = 1$

β_{target} = 1

$l_{T,} l_{TTARGET} = RMS_{loss}$

Again we ran 50 epochs in each iteration

4.3

Now that we have determined the optimal γ that provided the best result for our model's loss formula, we now fix γ as a constant in the optimization and we will perform a new cross-validation to determine the best $Loss_T$ for our model .

- $Loss_T \ will\ be\ partial\_RMS_{loss}\ or\ mean\_partial\_RMS_{loss}\ or\ Cumultive_{Loss}$

again the fixed parameters are:

$Loss_{OPTICAL\ FLOW}$

$\beta_{quat\ rotation} = 1$

$\beta_{optical\ flow} = 1$

$\beta_t = 1$

β_{target} = 1

This time we ran 100 epochs to get a better deviation in the results

4.4

Now that we have determined the optimal l_{TCRIT} We can proceed to the final experiment so that we can determine our model.

We will cross-validate the weight factors. As stated in article [2] this is no easy task especially when our loss formula is the summation of 4 different weighted losses:

$l = Loss_{WEIGHTED\ SUM}(l_{OF}, l_{QR}, l_t, l_{TTARGET},)$ =

$\beta_{optical\ flow} \cdot l_{OF} + \beta_{quat\ rotation} \cdot l_{QR} + \beta_t \cdot l_T + \beta_{target} \cdot l_{TTARGET}$

We've decided upon on our own technic for this cross-validation,
All parameters are fixed:

$l_{T,} l_{TTARGET}$

$Loss_{OPTICAL\ FLOW}$

$Loss_T$

γ

We will experiment with β values :

[0.5, 1, 5, 10, 50, 100, 500, 1000, 1500, 2000, 2500]

We've decided upon these values because in article [2] it is said:

"For the indoor scenes, it was between 120 to 750, and for outdoor scenes between 250 to 2000."

So we want to see if this will work in our model as well.

4.5

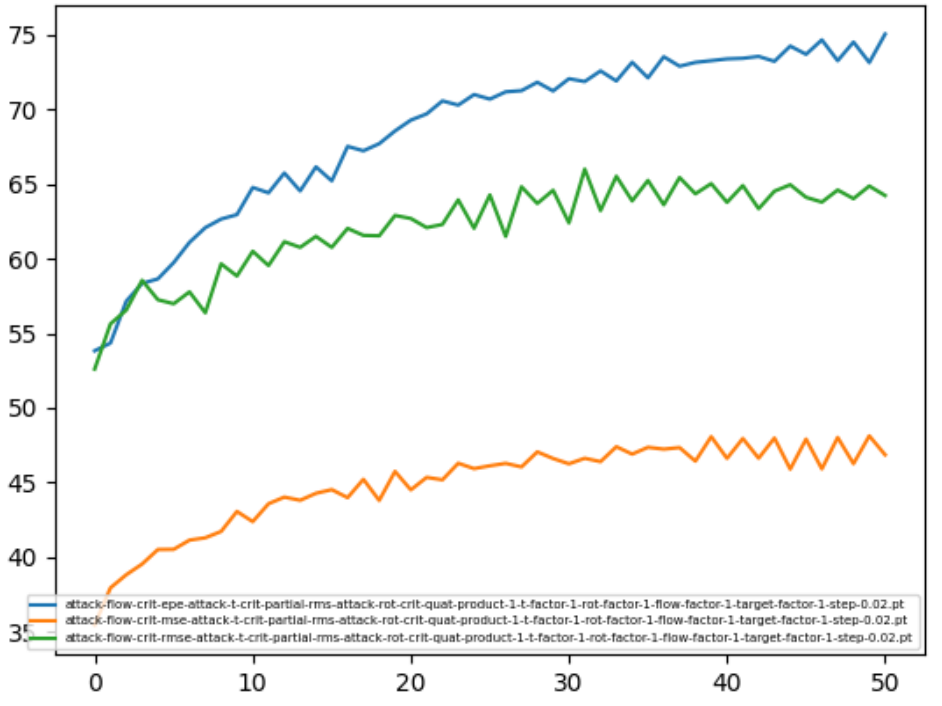
In 4.1 - 4.4, we've used cross-validation to find our best fit model.
In this part, we optimize our fit model hyper-params, to find the best step size for our model.
Before doing optimization on our fit model, we decided to use 80% of datasets (4) for **learning** our fit model and 20% of datasets (1) for **testing** our fit model.
We've found on github additional datasets [4] which we used for this optimization step.
We've also determined that the longest traj possible for the gpu is 21.
Now we will perform new cross-validation to determine the best step size for our fit model.

- The step size will be optimized with the following values:
  [0.01, 0.02, 0.03, 0.1, 0.2, 0.3, 0.001, 0.002, 0.003]

5. Results and Discussion

In this chapter we will discuss the results of the experiments in part 4 as well as the end results of our project:

Results of experiment 4.1:



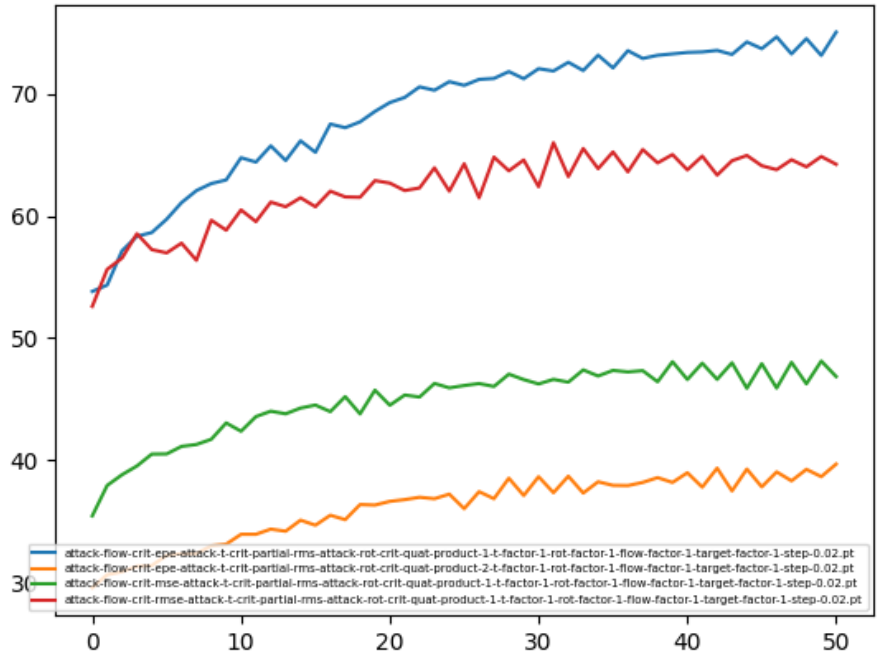
We can immediately see that the MSE_{loss} preformed much poorer than the other two, we can infer that this is because EPE_{loss} and $RMSE_{loss}$ are quite similar to each other.

Although the results of EPE_{loss} and $RMSE_{loss}$ are quite close we can see from the graph the best option for the optical loss evaluation criteria is the EPE_{loss}.

We think that maybe running on a different seed would result with $RMSE_{loss}$ being the better option.

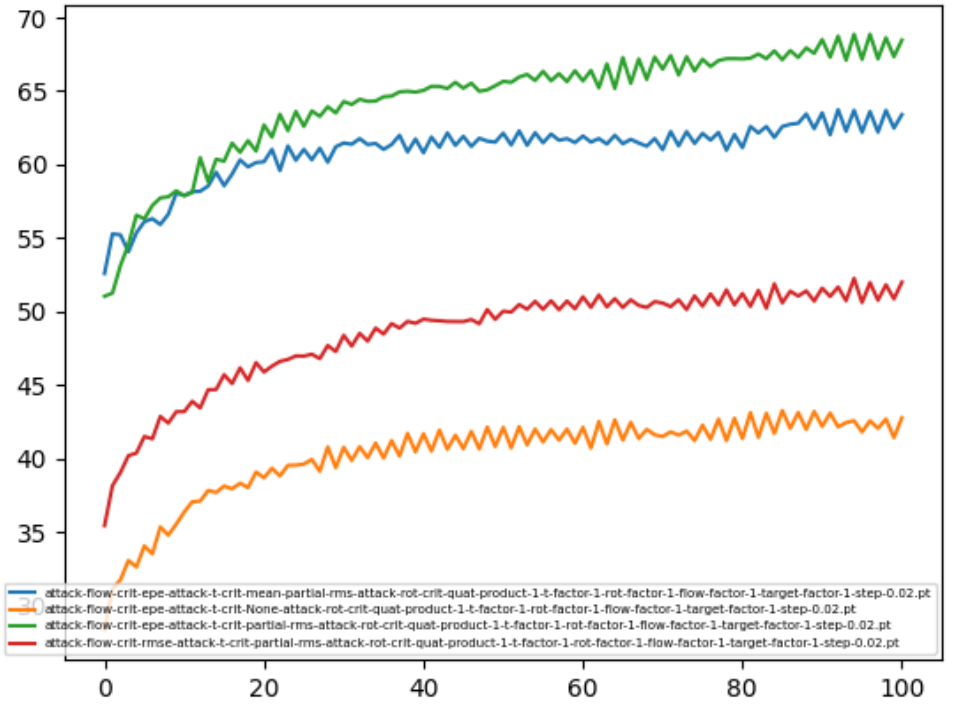
We've decided to proceed with EPE_{loss} in our model since it produced the best results in our test and is used in state of the art models as discussed in article [1].

Results of experiment 4.2:



In this experiment we can see that the best γ value is 1.
γ value of 2 drastically decreased the results in comparison to the same model with γ value of 1.
Therefore we've decided to chose γ norm for the quaternion rotation loss to be 1.

Results of experiment 4.3:

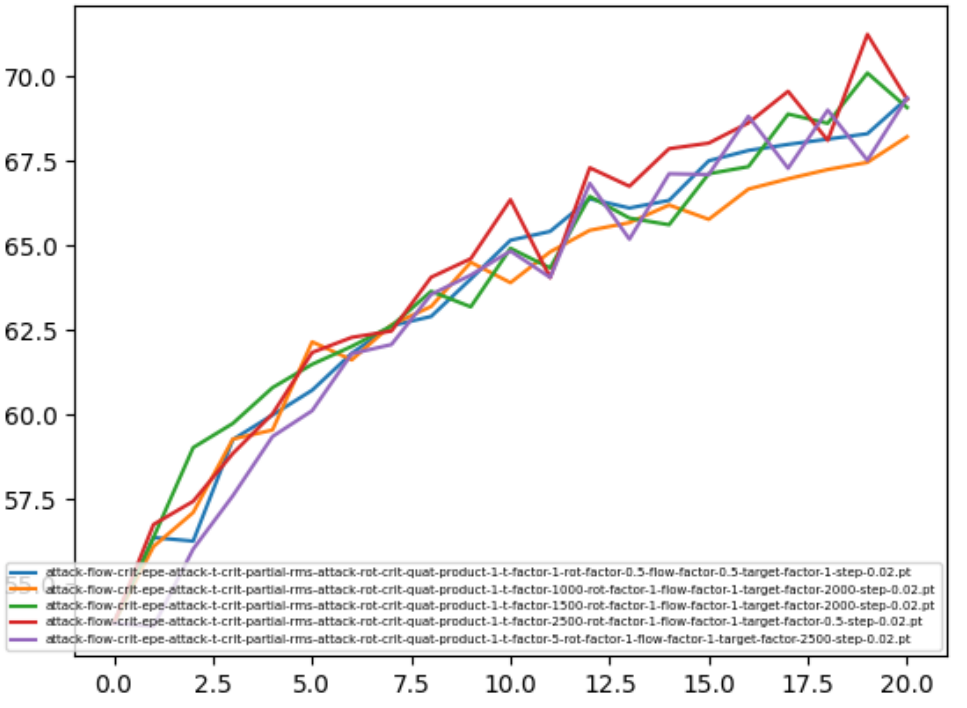


In this experiment, we can see that $partial\ RMS$, as $Loss_T$ category, produces the best results.

Since experiment 4.2's results weren't conclusive we ran the model with $partial\ RMS$, as $Loss_T$, and with $RMSE_{loss}$ instead of the EPE_{loss}

To validate that the trend of the graph continues for more advanced epochs as well, we ran these models with 100 epochs (instead of 50), and we got the results we expected about the EPE_{loss}.

Results of experiment 4.4:



In this experiment our goal was to find the best weight factors, we've cross-validated many different configurations.

In the graph above we can see the top 5 best models.

We've chosen are final model with the values:

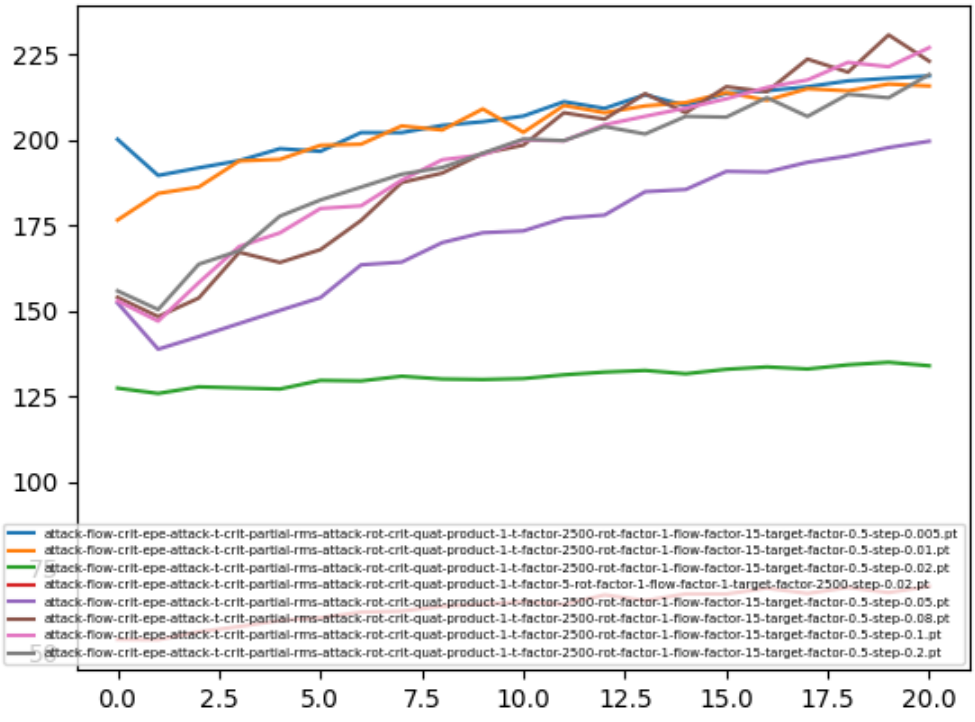
$\beta_{quat\ rotation} = 1$

$\beta_{optical\ flow} = 1$

$\beta_{t} = 2500$

β_{target} = 0.5

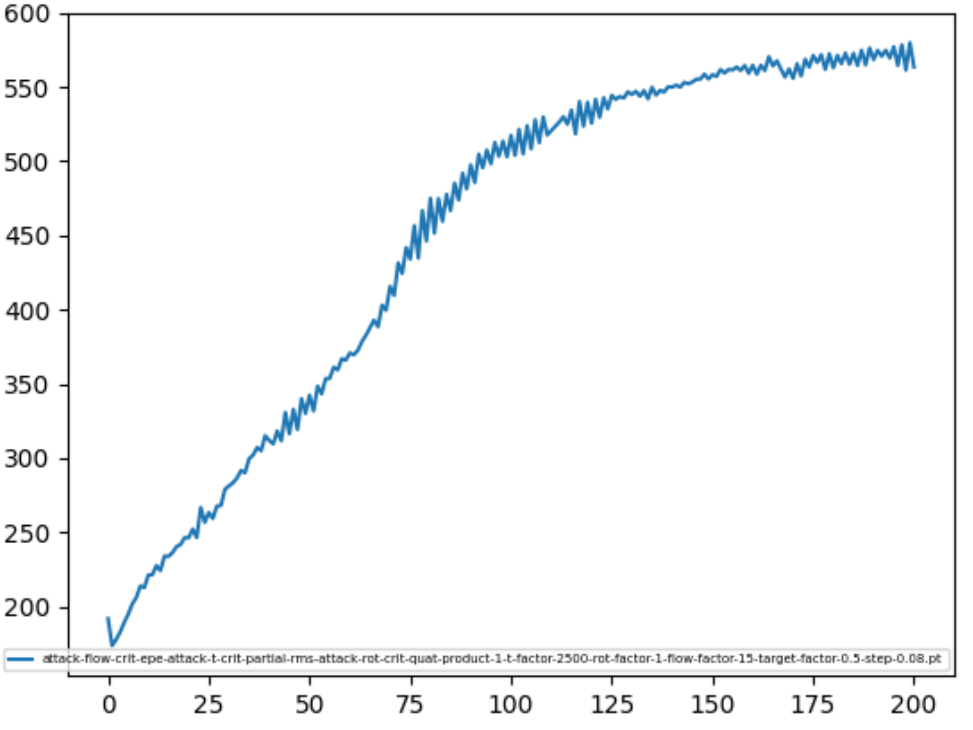
Results of experiment 4.5:



In this experiment our goal was to optimize the step size, we've ran several tests with varying step values. In the graph we can see the top 5 best values. Step size on 0.08 has performed the best.

We can also see in this experiment in comparison to the orders we can see a boost in performance because more data was inputted to the model [4].

Overall end results:



After cross validation to find the best model and hyper-parameters for our task, we let the model run for 200 epochs on the data set with a 9:1 train:test ratio.

Step size was set to 0.08

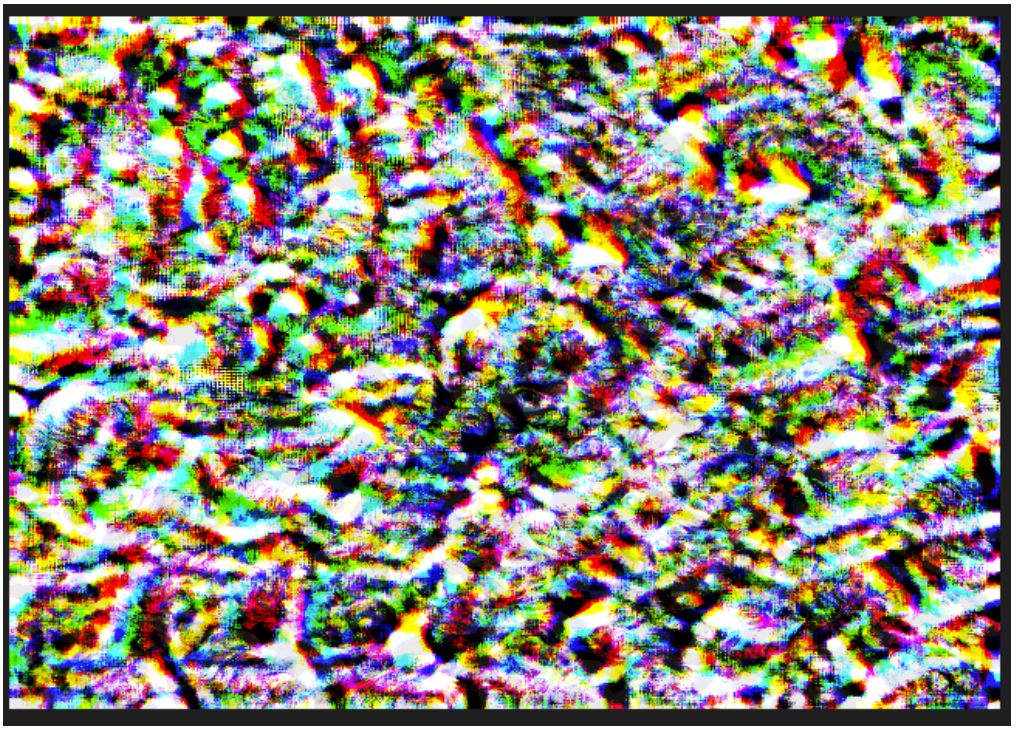
We've also determined that the highest possible trajectory length possible is 21, this is due to insufficient gpu memory.

We've managed to achieve a great results of

$l = Loss_{WEIGHTED\ SUM}(l_{OF}, l_{QR}, l_{t}, l_{TTARGET},)$ = 580

As we can see our results have much improved in comparison to the original model's loss funcion, we believe this is due to the inclusion of the quaternion rotation and optical flow elements to the loss formula.

And finally we have produced the best patch:



References

[1] Maxim Kuklin (Xperience.AI).: **RAFT: Optical Flow estimation using Deep Learning.** (2021)
https://learnopencv.com/optical-flow-using-deep-learning-raft/#raft-loss

[2] Alex Kendall & Roberto Cipolla (University of Cambridge).: **Geometric loss functions for camera pose regression with deep learning**. arXiv:1704.00390. (2017)
https://arxiv.org/pdf/1704.00390.pdf

[3] Maria A. De Jesus & Vania V. Estrela (UFF).: **Optical Flow Estimation Using Total Least Squares Variants.** ISSN: 0974-6471. (2017)
http://www.computerscijournal.org/vol10no3/optical-flow-estimation-using-total-least-squares-variants/

[4]: Patch Adversarial Attacks (Github).: **Additional Datasets.** (July 2022)
https://github.com/patchadversarialattacks/patchadversarialattacks