Project P3 : Neural Nemesis
Used 3 Late Days

Farhan Seliya
Department of Robotics Engineering
Worcester Polytechnic Institute
Email: faseliya@wpi.edu

Sarthak Mehta
Department of Robotics Engineering
Worcester Polytechnic Institute
Email: smehta1@wpi.edu

***Abstract*—This report presents our project on testing the vulnerability of a Depth Estimator Neural Network to targeted adversarial attacks. Starting with randomly initialised rgb values, a small patch is applied to images in a pre-trained Depth Estimator Network to estimate disparity. Using objective functions to minimise the difference in disparity from the target disparity in the patch region, the patch is trained to assume better patterns. Finally, the trained patch is then printed out and is tested on a real world scene and we compared the results of the Depth Estimator network with and without the patch to verify the extent to which the adversary could fool the network. The comparisons of the disparities of train and test dataset with and without the patches and the trained patches images are also provided in the report.**

I. Introduction

Neural Networks have demonstrated how successful they can be in various problems in computer vision, ranging from image classification to semantic segmentation etc. However, despite the impressive accuracy and robustness, these are not totally fail-proof. A key weakness of neural networks is their vulnerability to adversarial attacks. [1] These attacks work by making subtle alterations to input data—such as adding very slight, nearly invisible modifications to an image or a data sequence—that can mislead the neural network into producing incorrect predictions. This usually happens due to the underlying linearity of the way neural networks function. This vulnerability raises serious concerns about the dependability and security of AI systems, especially in high-risk applications such as autonomous driving, healthcare, and cybersecurity. Understanding how and why neural networks are vulnerable to such attacks is critical for increasing their robustness and assuring safe deployment in real-world contexts.

Specifically, Monocular Depth Estimator Neural Networks are very vulnerable to adversarial attacks, as the networks rely too much on non depth features like colour and shadows. [2]. In this work, one such Depth Estimator network produced by Guo et al. [3] has been used to train and test adversarial patches that can fool the network into estimating different depths for the patch regions. The patch can be trained to make the the network estimate a wrong depth using an objective function that minimises the difference between the target disparity and the disparity output by the network.

This work is the implementation of the procedure presented by Yamanaka et al. in [2]. In similar ways, we have trained targeted patches to make the network estimate the depths at 10 and 70 units respectively. Since the patches are also meant to be implemented in a real world setting, along with the disparity loss, two other losses were considered, namely the **Non-Printability Score (NPS)** and the **Total Variation (TV)** loss, primarily to ensure that the patch can be printable and applied in the real world by smoothing the pixel changes and using printable colours. Our trained patches are shown below in Fig[1]

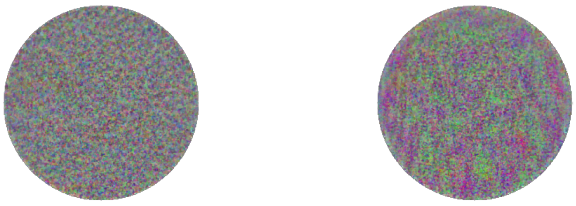

Fig. 1. (Showing our trained patches here) Left: Patch for disparity 10. Right: Patch for disparity 70

To reach the objective of applying it to the real world, many augmentations such as affine transform, perspective scaling and random colour jitter are applied to the patch so as to mimick the lighting and shadow effects a printed patch faces in the real world. As a result, the imaging process of the patch is modeled to account for these diverse shooting conditions.

The further report is divided as follows. Section II describes the process of Patch Augmentation, Section III described the Patch training and Validation and Section IV describes the real world implementation of the patch.

II. Patch Augmentation

In the context of the depth estimator networks, a patch is nothing but a region of pixels with different rgb values from the rest of the image and these values can be trained.

We started off with a patch with its rgb values initialised randomly and performed augmentation further.

In the patch augmentation process, the patch is initially generated with random values to ensure diversity. To simulate real-world conditions, various augmentations are applied to the patch using `torchvision.transforms`. These augmentations include adding random noise, adjusting color properties like brightness and contrast through color jitter, and applying geometric transformations such as random affine and perspective changes. These manipulations help the patch adapt to different physical environments when placed in a scene. The augmentations are shown below in Fig[2]

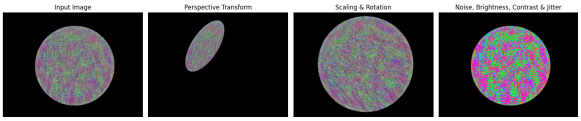

Fig. 2. Augmentation using Perspective Transform, scaling, noise and color jitter..

After applying the augmentations, the patch is integrated into the image using a mask. The mask controls where the patch is applied on the image according to the following equation:

$$(1 - M) \cdot Image + M \cdot Patch \tag{1}$$

where M is the binary mask that determines the location of the patch on the image, ensuring that only specific areas are modified while the rest of the image remains unchanged. This method allows for a seamless blending of the patch into the target image, mimicking how it would appear in real-world scenarios. A sample image for attack is as shown in Fig[3]

Fig. 3. Sample Image after Augmenting the patch with the training image.

III. Patch Training and Validation

A. Mathematical Representation of the Attack

For patch training and validation, we trained the patch over a given image dataset, focusing on simulating specific depths at 10 and 70 meters, respectively. The training process involved optimizing the patch using three key loss functions to ensure it effectively manipulates the network's depth estimation. Similar to the method presented in [2], we first attacked the images from the training dataset with our patch and pass it into the Depth Estimator network of [3]. This estimate is then corrected to show the target depth in the patch region using our loss functions. This is mathematically represented as follows:

$$\arg\min_{P} \sum_{(i,j)\in R_\theta} ||d_t - F_{i,j}(I +_{R_\theta} T_\theta(P))|| \tag{2}$$

where d_t is the target depth for which we trained the patch, $F_{i,j}$ represents the neural network F applied to each pixel (i, j) of the image. The term $(I +_{R_\theta} T_\theta(P))$ entirely represents the image I attacked with the patch P adjusted to the transformations T_θ over the region R_θ. The $argmin$ returns the patch P for which the depth estimate is the most closest to the target.

B. Loss Functions

In order to train the patch with the objective mentioned in the previous section, we defined the patch as a `torch.tensor` and its transformations as a `torch.nn.module` with gradients enabled for the patch tensor. Now with this, we can train the patch in the same way we train a neural network by defining loss functions and performing backpropogation steps to optimise the parameters (the pixel values of the patch in our case). The loss function used in this work is as follows:

$$L = L_{\text{depth}}(P) + \alpha L_{\text{NPS}}(P) + \beta L_{\text{TV}}(P) \tag{3}$$

It is a weighted sum of three individual losses which are described as follows. The weights are implemented directly from []:

1) ***Depth Loss****:* The first loss function aims to minimize the error between the estimated depth and the targeted depth. Specifically, we employ an L_1 loss to reduce the disparity error between the preset depth and the network's output within the patched area. This forces the network to generate a fixed, constant disparity where the patch is applied, ensuring the patch manipulates the depth perception consistently. Mathematically, this is just implementing the objective function in the earlier section and can be shown as:

$$L_{\text{depth}}(P) = \sum_{(i,j)\in R_\theta} ||d_t - F_{i,j}(I + R_\theta T_\theta(P))|| \tag{4}$$

2) ***Non-Printability Score(NPS)****:* To ensure that the patch remains printable and realistic, we incorporated a Non-Printable Subspace (NPS) loss function. This function minimizes the L1 error between the patch's color values (r, g, b) and the closest colors within a printable color gamut. The purpose of this loss is to ensure that the patch's color remains within the range of colors that can be accurately printed, preventing distortions that could arise from non-printable hues.

$$L_{\text{NPS}}(P) = \sum_{i,j} \min_{\vec{c}\in C} |\vec{p_{i,j}} - \vec{c}| \tag{5}$$

Here the $p_{i,j}$ denotes the color vector of a pixel (i, j) in the patch P. We seek the closest color vector c from the set of printable colors C.

*3) **Total Variation (TV)**:* Finally, to ensure smoothness in the patch and avoid harsh pixel transitions that would not print accurately on physical media, a smoothness loss function was introduced. This loss penalizes sharp changes between adjacent pixels by minimizing the variation between successive pixels. This encourages a more gradual and continuous texture, ensuring that the patch can be printed with high fidelity on a physical sheet. Together, these loss functions ensure that the patch is effective in influencing depth estimation while maintaining printability and visual smoothness.

$$L_{\text{TV}}(P) = \sum_{i,j} \sqrt{(p_{i,j} - p_{i+1,j})^2 + (p_{i,j} - p_{i,j+1})^2} \quad (6)$$

Once the losses are defined, the patch can be trained and since its gradients are set to true, the patch updates its pixel color values in order to minimise the loss.

The patches after training are shown in Fig.[4]

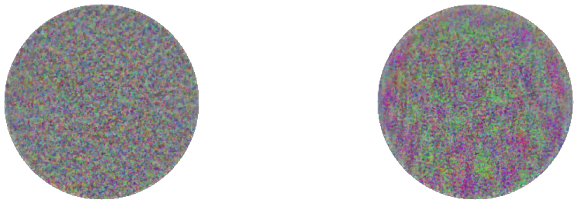

Fig. 4. Patches after training. Left: Patch for disparity 10. Right: Patch for disparity 70

C. Validation

For images given in the validation set, we calculate disparity using the model and the patches given for disparity values 10 and 70. The comparison results with respect to original images are as shown in Fig[5 - 12]

As demonstrated in the images, the depth estimator network struggles to accurately predict the depth in the area containing the patch. Instead, it estimates the depth of the patch as either near or far, depending on the target depth for which the patch was trained. Although visually the disparity images almost look the same due to low contrast, a clear difference can be observed in the depth estimation as shown in the difference image for each validation image. This shows that the patch caused the network to make an error in the depth judgement.

IV. Real World Testing

For real-world testing, we printed the trained patch and attached it to a physical object within a real-world environment. A video of the scene was captured with the

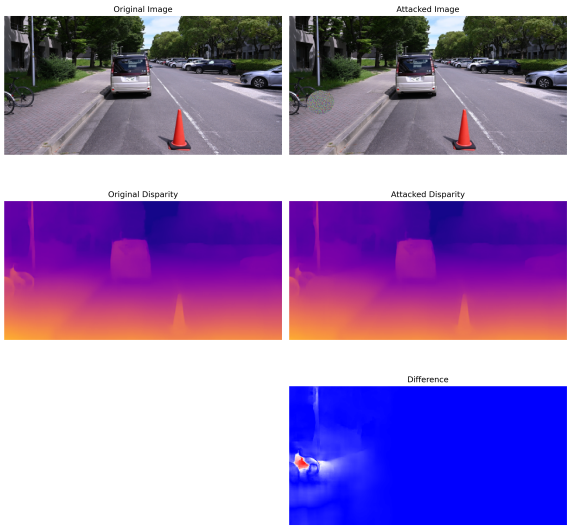

Fig. 5. Test Image with patch for disparity 10

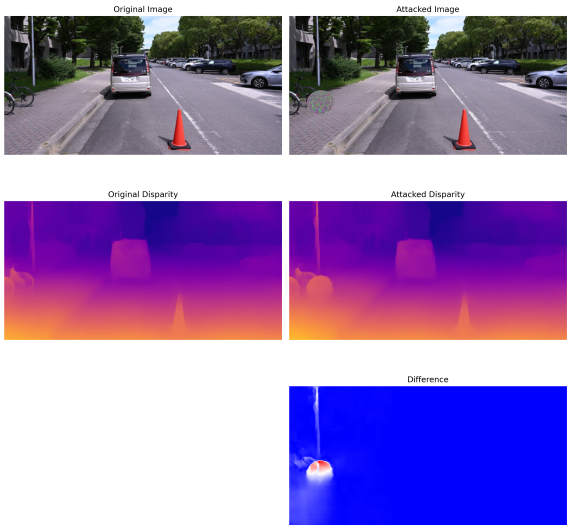

Fig. 6. Test Image with patch for disparity 70

patched object, ensuring various angles, lighting conditions, and perspectives were included. We then processed each frame of the video through the depth estimator to assess the patch's effectiveness outside of the controlled training environment. The comparison in real-world image is as shown in Fig[13] . Now since there is small difference (slight position variation, slight camera movement) between images when we use the patches and when taken without patches, so the disparity has other errors as well, apart from the error caused due to patch.

By comparing the depth estimation results between frames with and without the patch, we observed that the adversarial

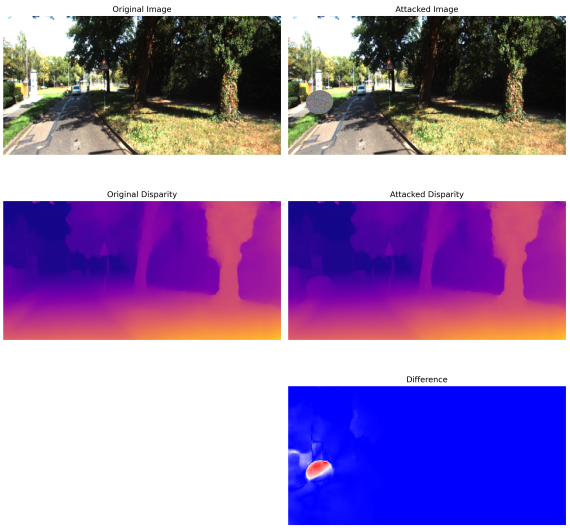

Fig. 7. Test Image 2 with patch for disparity 10

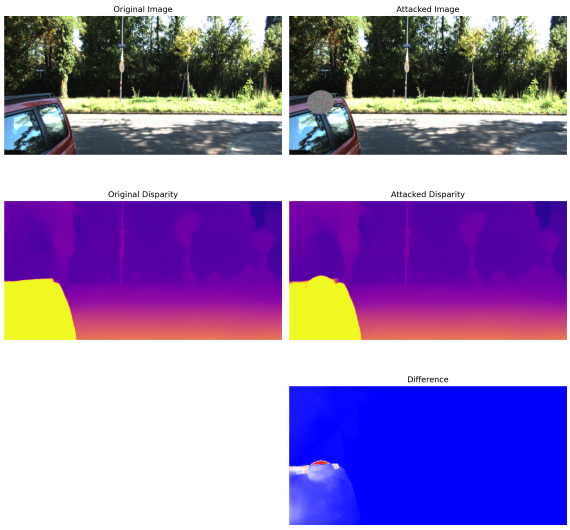

Fig. 9. Test Image 3 with patch for disparity 10

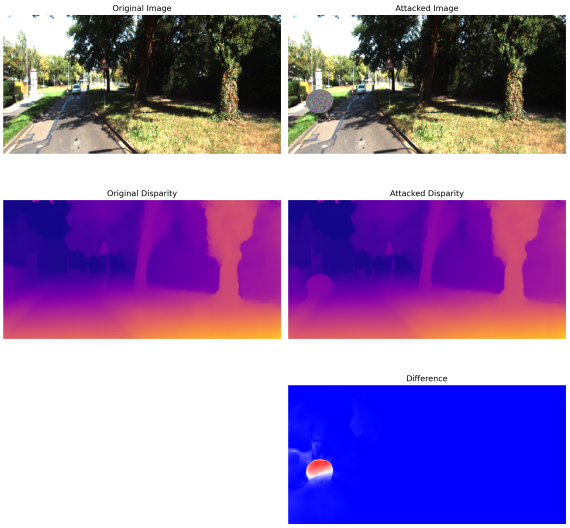

Fig. 8. Test Image 2 with patch for disparity 70

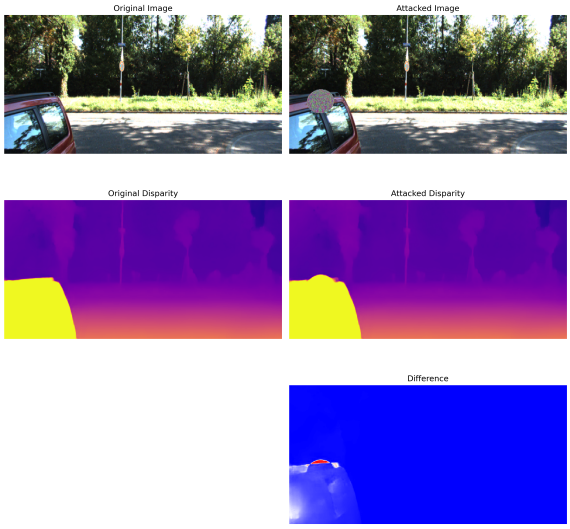

Fig. 10. Test Image 3 with patch for disparity 70

patch successfully influenced the network's predictions, even in real-world conditions. The depth estimator consistently miscalculated the depth around the patch, aligning with the target depth values for which the patch was trained. This confirms that the patch maintained its adversarial effect despite real-world transformations like lighting changes, camera noise, and object motion, demonstrating its robustness and applicability beyond synthetic or controlled setups.

V. Attack using the Fast Gradient Sign Method (FGSM)

The adversarial attack presented in this work is a targeted adversarial attack where the network is made to predict a desired target value (depth in this case). The training of the adversary is carried out through gradient descents and by minimising a total loss function. However, this is not the only way we can train an adversary to attack a network. Another type of attack called the **Fast Gradient Sign Method (FGSM)** [4] trains the adversary in an opposite way, by performing **gradient ascent** and **maximising** a loss with respect to the input image.

Mathematically, it can be expressed as finding $argmax$ of the loss function and performing the gradient ascent to modify the image within the attack budget. This method is fast since the

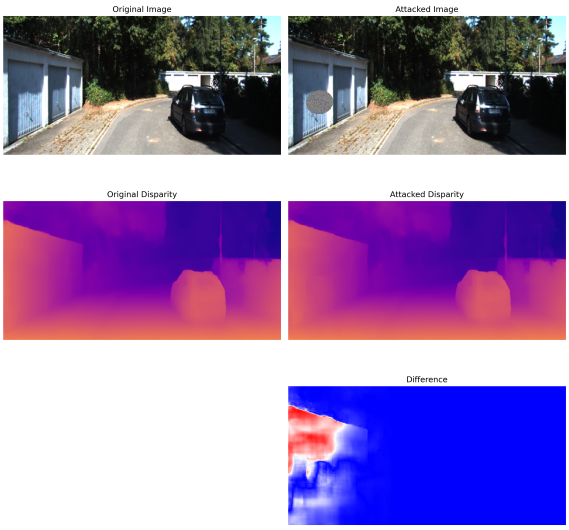

Fig. 11. Test Image 4 with patch for disparity 10

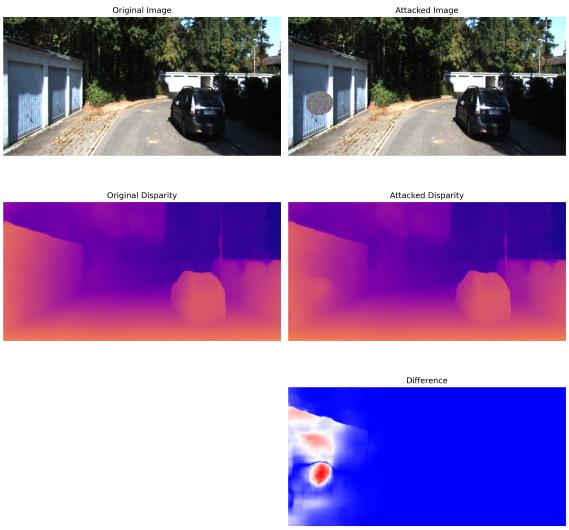

Fig. 12. Test Image 4 with patch for disparity 70

ascent step is carried out only once.

$$x_{\text{adv}} = x + \arg \max_{\delta:\|\delta\|_p \leq \epsilon} (L(\tilde{y}, \hat{y}, x + \delta \mid \theta)) \tag{7}$$

To implement the FGSM method to our depth network, we can modify the loss function as follows:

$$L = L_{\text{new depth}}(P) - \alpha L_{\text{NPS}}(P) - \beta L_{\text{TV}}(P) \tag{8}$$

where

$$L_{\text{new depth}}(P) = -\lambda_1 \|d_{\text{target}} - \hat{d}_{\text{estimate}}\| + \lambda_2 \|d_{\text{truth}} - \hat{d}_{\text{estimate}}\| \tag{9}$$

in this, d_{target} and d_{truth} are the target and true depth values of the patch region in the image that is being attacked.

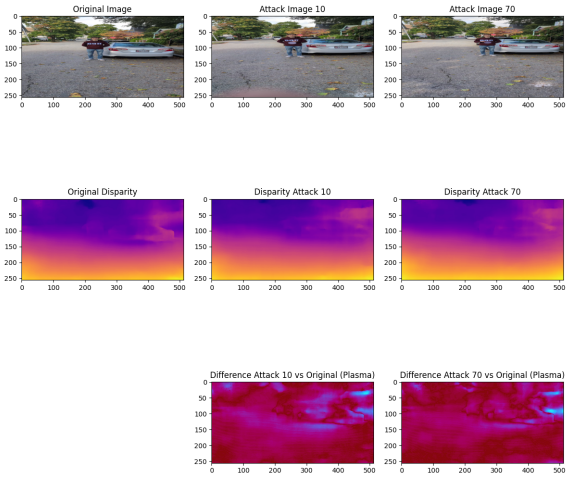

Fig. 13. Real world testing

$d_{estimate}$ is the depth output produced by the network given the attacked image. λ_1 and λ_2 are parameters that can weigh the losses according to their importance. Rest all losses are as discussed above. The positive and negative signs in front of all losses are present such that performing a maximisation over the loss function would help in training the adversary to fool the network better.

VI. Conclusion

This project showed the vulnerability of neural networks to adversarial attacks. We presented that by training an adversarial patch with a targeted depth estimate, we can make a depth estimator neural network to make a wrong prediction about the depth of the scene. This was done by defining loss functions which ensured that the patch region returns a single depth value and by ensuring the printability of the patch for real world application. A comparative study was also presented between the depth images of the scenes both with and without the adversarial patch, to show the effect of the patch. Finally, another method of attack, namely the Fast Gradient Sign Method, was also discussed and we showed how this could be used to attack the network in this work.

References

[1] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[2] Yun, Gyungeun, Joo, Kyungho, Choi, Wonsuk, and Lee, Dong Hoon. "Poster: Unveiling the Impact of Patch Placement: Adversarial Patch Attacks on Monocular Depth Estimation." In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 3639–3641. New York, NY, USA: Association for Computing Machinery, 2023. DOI: https://doi.org/10.1145/3576915.3624400.

[3] Guo, Xiaoyang, Li, Hongsheng, Yi, Shuai, Ren, Jimmy, and Wang, Xiaogang. "Learning Monocular Depth by Distilling Cross-Domain Stereo Networks." In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part*

XI, 506–523. Berlin, Heidelberg: Springer-Verlag, 2018. DOI: https://doi.org/10.1007/978-3-030-01252-6_30.

[4] Goodfellow, Ian J., Shlens, Jonathon, and Szegedy, Christian. "Explaining and Harnessing Adversarial Examples." In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. Available at: http://arxiv.org/abs/1412.6572.