RBE 474X: Neural Nemesis Using Four Late Days

Jakub Jandus, Hudson Kortus, Dexter Stark
Worcester Polytechnic Institute
Worcester, Massachusetts

***Abstract*—In this project, we learned how to attack a neural network so that it misjudges the distance between the camera perspective and the closest object. By doing this we are able to better understand what can cause a neural network to produce incorrect results and how to spot these attacks in the future.**

I. Introduction

For project P3: Neural Nemesis, the goal was to attack the data presented to a neural network by adding patches of varying color and shape. This should cause the network to believe that objects in the data are shown as being closer or further away than they actually are. Through this process we will gain insight into just how vulnerable these networks can be, as well as the danger that this can pose in real world scenarios.

II. Patch Creation

In order to make the attack on the network as effective as possible, a random patch will be applied and trained on sample images from the KITTI dataset. When attempting to create these patches, we ran into several issues. A lot of them with the provided started code which made this assignment take longer than usual. There was difficulty getting the dataloader to function properly as the error disguised itself as an error in the source neural network python file. When attempting to fix another bug with the backpropagation gradient, the PyTorch library error message would simply respond with "... Good Luck!", indicating the incredible difficulty to get to a workable result in this project.

Before applying these patches, they were augmented for each image. The kinds of augmentation that we used were perspective transformations, scaling, rotation, adding Gaussian noise, slightly changing brightness and contrast, and hue variation. These augmentations aid in generalizing the patch for real world scenarios and imperfect cameras.

III. Patch Training

Given the capabilities of the adversarial patches, a means of directing the way in which it affects neural networks was necessary to gauge its effectiveness. For the sake of this project, patches were trained to simulate a distance of 10 units and 70 units from the perspective of the network on 100 epochs. The pixels were back-propagated through a custom disparity (distance) loss function. With the final goal of printing the patches and testing in the real world in mind, another function relating neighbouring pixel transitions smoothness

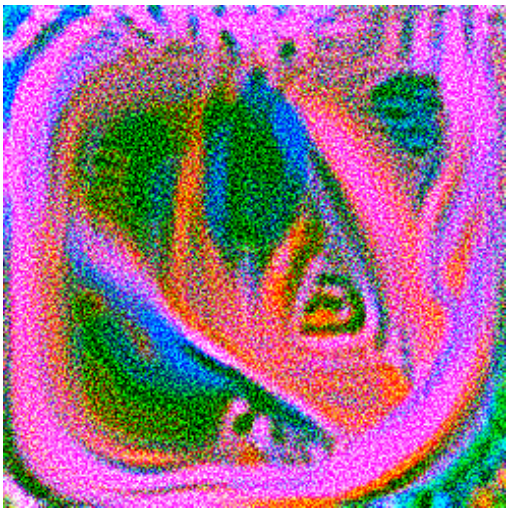

Fig. 1. Patch Trained for 10units 256px

Fig. 2. Patch Trained for 70units 256px

was implemented with a customizable ratio between them. A decision not to implement the loss function that takes into account how close are the patch colors to an ink printer colors was made. The color jitter applied to the patch would invalidate that computed loss. The training of each patch took over two hours. The team experimented mainly with different patch resolutions. In attempts to decrease the training time changing the resolution, loss function ratio, and the other properties was attempted, but had very little effect on both the training time and surprisingly the appearance of the adversary patches.

Fig. 3. Patch Trained for 10units 128px

Fig. 4. Patch Trained for 70units 128px

IV. Patch Validation

In order to verify the impact of the attack patch on the neural network, two different tests needed to be run. One test is done with the patch present, and the other without. To get a clear visual comparison, we analyzed the different detected distances using a plasma heat map of the scene. This set of tests was done for both the 10unit and 70unit patches in order to verify their effectiveness.

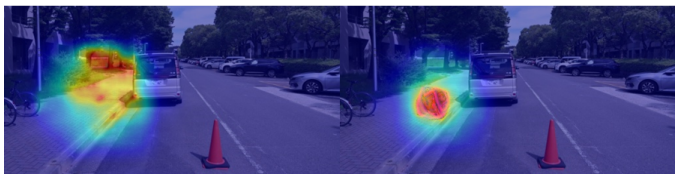

Fig. 5. Heat map for 10units

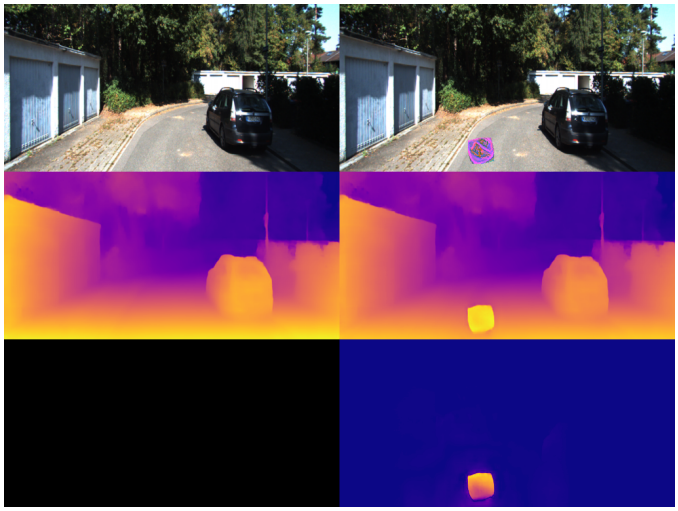

Fig. 6. 10unit Patch Validation (Left: Area of interest before attack. Right: Patch's effect on the area)

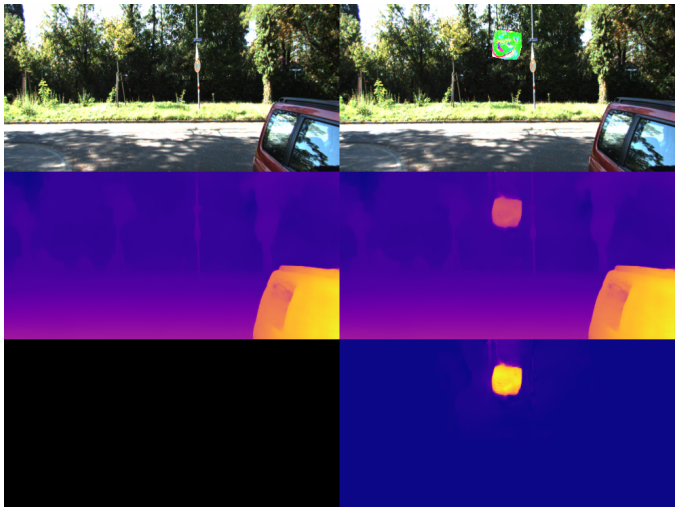

Fig. 7. 70unit Patch Validation (Left: Area of interest before attack. Right: Patch's effect on the area)

Checking to see whether or not the patches were working as intended was not as simple as we anticipated, as the heat map was difficult to read properly. It is clear that the different patches were having an effect on the image. The 10unit patch

is registering as being much closer, while, strangely, the 70unit patch registers also as closer instead of the desired further aways. However, without a key or legend to gauge the true distances, it makes it very difficult to say for sure that the patches were simulating the exact distances that we trained it for.

V. Real World Testing

As a further step of validation, a real world test was conducted see the attack's effectiveness in a live scenario. All six available patches were printed out on letter sized pieces of paper and trimmed to remove the white backgrounds. After taking note of the low quality of the patches that were printed at pixel square sizes of 64 and 128, we realized that these would smear when viewed on camera, so we opted to run our tests with the patches that were created at a pixel value of 256. This lack of quality was mainly due to the limitations of the library printers, which was anticipated. Three different recordings were taken of the patches while taped to a street lamp: One with the 10unit patch, one with the 70unit patch, and a third that contained both. To increase the potential variability in results, we placed the patches at different heights.

Fig. 8. Printed Patches

Fig. 9. Outside Validation of Both 10unit(bottom) and 70unit(top) Patches Showing Some Effect on the Image

From analyzing the results of the effect, it was clear that the 10unit patch was only a little more effective than the 70unit patch. Potential reasons for this behavior could be that the patch was not large enough or clear enough to be registered by the network. In addition, it could be the fault

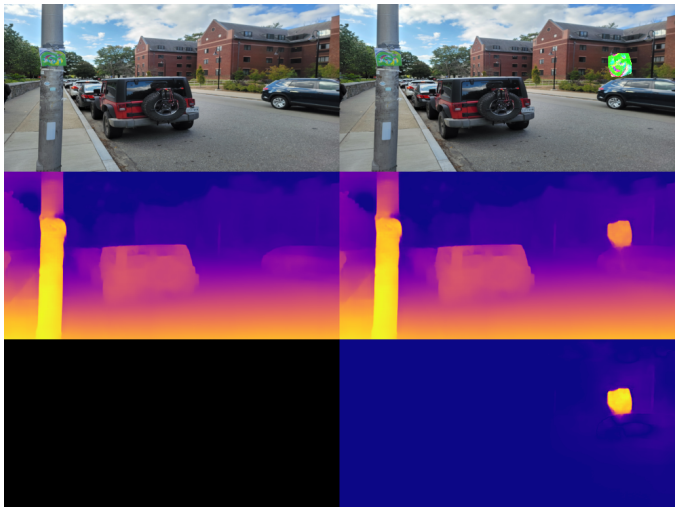

Fig. 10. Comparison of the Effects of our 70units Artificial Patch Printed in Real World on Left vs Additional Code-Added Second Patch on Right. The Far Patch Strangely Shows as Being Close

of the patch training or the individual loss function ratio. Due to various time constraints and conflicting assignments, we are unfortunately missing the control footage of the street that does not include any patches.

VI. Fast Gradient Sign

In considering the implementation of the Fast Gradient Sign method of creating adversarial patches, certain parts of the network need to be analyzed and targeted to maximize the patch's effectiveness. Since this method is built around counteracting the network's loss function, it requires having the input image, input label, model parameters, and the network's loss. We can easily provide the model parameters as they will not change after the model is finished being trained.

As for introducing it into our own code, we already have everything we need but the ground truth label. Upon obtaining such label either with a LiDAR, or from a different network using stereo-pair cameras, we can obtain the difference between the model prediction from the source image and the ground-truth label. Using a simple torch inbuilt loss function we can obtain the loss for that particular image with respect to the ground truth and the corresponding gradient. Next, we run the gradient through the signum function, to obtain values of +1, 0, or -1 only. In the last step, we scale those unit values by a small epsilon value and add it to the original source image, thus slowly obtaining our adversarial image. This image will inherently look like noise to a human eye, but it is very efficient at fooling the artificial neural networks. The advantage is the speed of the training, with a large disadvantage of having to posses the accurate ground truth.

VII. Conclusion

The use of targeted attacks on a neural network can pose a serious threat under certain circumstances. By learning the required steps to create these attacks, we can gain an

understanding of their severity and consider potential ways to weaken or mitigate them in the future. From executing real world tests, we were able to determine the effectiveness of the adversarial patches natural environment. Even when creating one that was able to be registered by the network, it showed that it would need further improvements before it could truly be effective. By comparison, the tests that were run using the given data proved to have a greater impact. This in turn tells us that patches used in the real world require a lot more time and effort in order to produce the desired results.

REFERENCES

[1] K. Yamanaka, R. Matsumoto, K. Takahashi, and T. Fujii, "Adverasarial Patch Attacks on Monocular Depth Estimation Networks," IEEE Access, Vol. 8, pp. 179094–179104, 2020.

[2] X. Guo, H. Li, S. Yi, J. Ren, and X. Wang, "Learning monocular depth by distilling cross-domain stereo networks," Proceedings of the European Conference on Computer Vision (ECCV), pp. 484–500, 2018.