EE641: Final Project Report

Kyle Murphy, Kaustav Chakraborty

August 10, 2025

Abstract

Vision-based robotic systems are seen as a crucial tool in modern-day autonomy. Advances in deep learning have allowed robotic systems to employ cheap sensors like cameras to perform remarkably well in tasks such as navigation or manipulation. Such deep learning algorithms, however, are trained using tremendous amounts of data to perform such tasks appreciably. Even then, these systems often fail if the run time environments differ from those in training. This leads to problems of bad *generalization* and *out of distribution* data. A promising way of resolving such a shortcoming could be to *augment* the training data with additional samples that could be adversarial and cause the learning system to fail, resulting in a richer dataset. Yet searching for such counter-examples or edge cases for failure would be a painstaking and laborious task. This work proposes an automated self-supervised algorithm to generate novel failure examples of such learned pipelines. This will allow us to augment the training dataset of such systems appreciably to remedy such out-of-distribution failures, resulting in higher generalization capabilities of such algorithms. Our work employs a two-step modular approach to generate counterexamples - a failure detection mechanism that uses tools from reachability theory and a scenario generation module that learns the latent embedding from the detection mechanism to generate novel counterexamples.

1 Introduction

1.1 Motivation

Modern robotics use multiple sensor modalities in localization and navigation algorithms; however, this comes at the cost of added complexity. Cameras-based visual sensors are easy to use and inexpensive in terms of procurement costs. Hence they have emerged as one of the go-to options for such tasks. A prime example of such a visual-controller-guided robot can be seen in [1], where the authors presented a computer vision approach to creating paths to a goal. The algorithm employs RGB images as inputs, from an indoor environment, like an office, to a wheeled autonomous robot for navigating to different goal locations. While the proposed approach worked well, it had many issues regarding environments that were slightly out of its training scope. These environments typically consist of the exact location with different lighting. Such failure due inability of the navigation pipeline to generalize to small shifts in the data distribution can be seen as a common failure in learning-based systems. To combat such issues [2], employed tools from reachability analysis to discover closed loop failures of vision-based controllers. Even though the method exposed the standard failure modes for a vision-based pipeline, the number of actual examples of such failures was too few to be of any immediate use. Another example of vision-based navigation [5] employs a simple ResNet-based network to take in a continuous stream from an RGB image of a virtual camera to localize a virtual aircraft and generate control commands to keep the flight on the runway. The system's task is to keep the virtual aircraft on an airport runway without veering off its path. It was, however, prone to errors with the changing light conditions of the runway at different times of the day. Hence a network trained with images from, say the morning was seen to be inefficient for the network to reliably function at dusk in low-light conditions Subsequent development in GANs [3] allows preferential generation of images similar to original training data but with new characteristics to cause the network to fail. Inspired by these existing works, we propose a novel and automatic pipeline for generating a large corpus of any learned vision-based architecture failure modes. The outputs from such a generator can then create adversarial datasets that will allow for better generalization while training such vision-based networks.

1.2 Literature Review: Cycle Gan and Taxi Net

1.3 CycleGAN

CycleGAN (fig.1) is a generative network that allows for style transfer to be trained with unpaired data sets. Instead, the network only needs images in Domain A and Domain B with no requirements for the image set to be paired[3]. CycleGAN uses two generative networks, one generator A to B takes images in domain A and transfers them into domain B. The second is generator B to A. This takes images in domain B and transfers them into domain A [3, 7]. The network is trained by using cycle consistency. This is achieved by inputting an image into Generator AB and then inputting the generated image into generator BA to reconstruct the image. The final image is then compared to the input image. This is using transitivity as a regularizer.CycleGAN being an adversarial network, also use a discriminator. The discriminator is used on the generated outputs from generator AB.

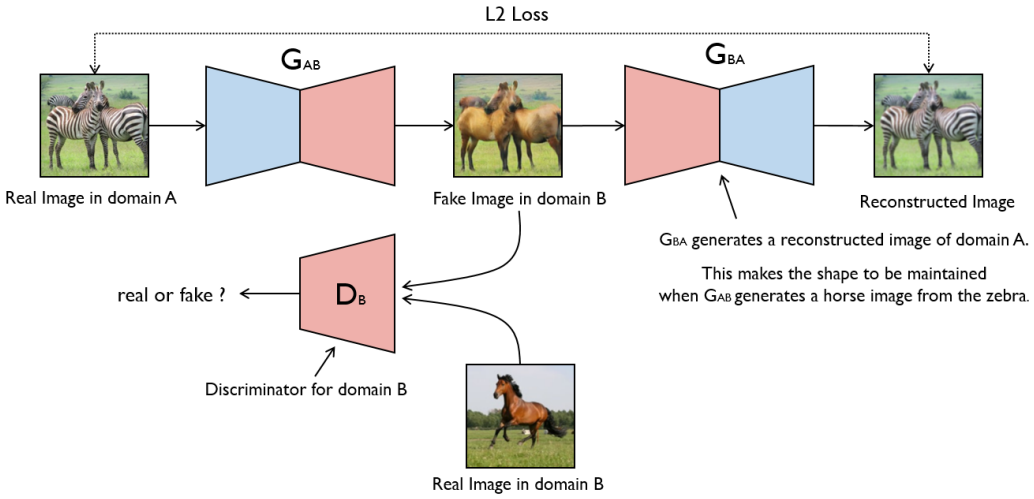

Figure 1: The architecture of Cycle Gan

1.4 TaxiNet

Now we introduce the aircraft taxiing problem [5] that we will use as a running example to illustrate the key concepts.

TaxiNet is part of an autonomy program at Boeing. Here, the robot is a Cessna 208B Grand Caravan. The dynamics of the aircraft are:

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\,cos(\theta) \\ v\,sin(\theta) \\ u \end{bmatrix} \tag{1}$$

The state is given by $\mathbf{x} = (p_x, p_y, \theta)$, where p_x is the crosstrack error (CTE), measured as the lateral offset of the aircraft from the centreline of the trackway, p_y is the downtrack position (DTP), which is the position along the length of the runway, and θ is the heading error (HE) of the aircraft in degrees from the centreline (Fig. 2 shows how these quantities are measured). v is the linear velocity of the aircraft kept constant at 5 m/s, and the control u is the angular velocity. The goal of the aircraft is to stay on the runway of an airport (for the entire episode) while moving with a constant velocity, using only the images that it sees through a camera mounted on its right-wing using a Convolutional Neural Network (CNN), which returns the estimated CTE, $\hat{p}_x$, and the estimated HE, $\hat{\theta}$. A proportional controller (P-Controller) then takes these predicted values to return the control input fed to the aircraft allowing it to change its heading. We use the X-Plane simulator that returns a rendered RGB image from a virtual camera mounted on the aircraft's right wing at any state.

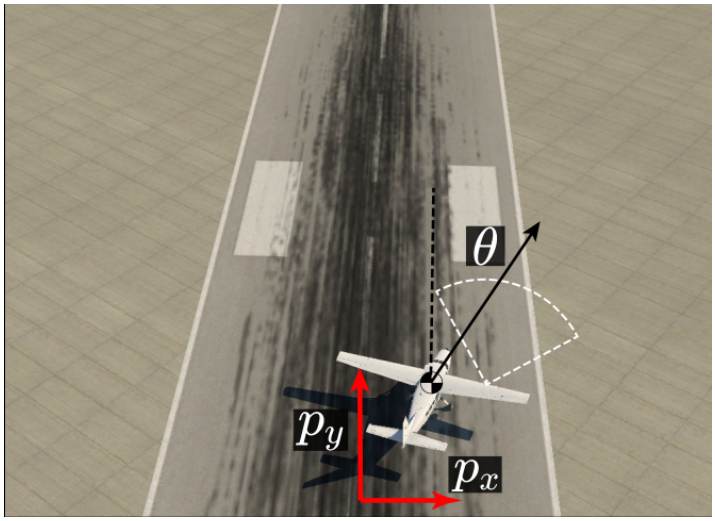

Figure 2: Measurements from the XPlane Simulator

2 Materials and Methods

2.1 TimeLine and Milestones

Training and testing our model was an iterative process. Our first objective was to simply get a working CycleGAN code. This was aided by code from GitHub [7]. Once we had a working code for CylcleGan, the next step was to modify the code to replace the discriminator with a static TaxiNet model and test if the generators would produce reasonable images. The TaxiNet was used from a NASA project [6]. The first version of TaxiNet used was a simple 4-layer MLP. A simpler model was preferable for testing. Once we generated acceptable results, the TaxiNet was replaced with a ResNet18 version. This is also from the same NASA repo [4]. We then repeated the same steps until we were getting good image results. The next step was to then set the TaxiNet as trainable and modify the code to train TaxiNet alongside the generators. When this was working, we then trained two final models, one with the AB identity loss weight being 0.1 and one with no AB identity loss weight. The final milestone of our project was testing the final models against NASA's best TaxiNet model.

2.2 Model

The model architecture is based heavily on CycleGAN with some modifications, as shown in fig. 3. The major difference is that the discriminator has been replaced with TaxiNet. This will allow us to adversarial train the TaxiNet. The model keeps the two generators, AB and BA. The first generator takes in the training image, and the second generator takes in the generated image. Another modification is instead of inputting the domain B image into the TaxiNet, the input image is also fed into TaxiNet. This allows the network to train on both adversarial and good images.

2.3 Data

In order to sample the data, we use the Xplane simulator. We simulated the conditions for the daytime and sampled around 50K images with the corresponding actual state of the aircraft. Thus, the aircraft's image and corresponding ground truth location form our dataset. We also use a previously available online dataset to supplement our dataset as part of another project [4].

2.4 Data Augmentation

Some data augmentation was performed on the data set. The images are resized to 224x224 pixels and then normalized. After which, they are converted to greyscale and a further crop to reduce the resolution to 256x128. Furthermore, we added a bias of 0.5, which shows a better performance of the CNN at different times of the day. These are the only augmentations performed on the data.

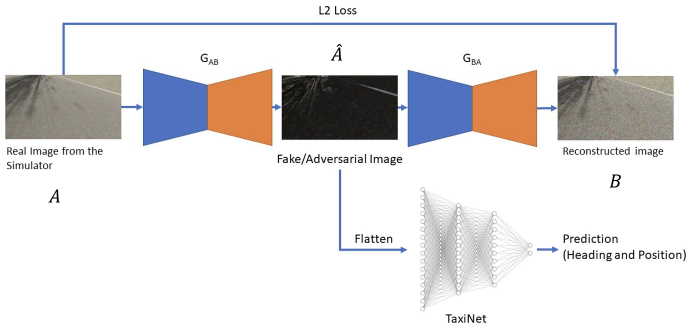

Figure 3: The architecture of Our Model

2.5 Loss Functions

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

$$L1 = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}|I(i,j) - K(i,j)|$$

Our model used a combination of multiple loss functions. Loss functions for the generators that involve images use the L1 loss, and losses involving the TaxiNet use MSE. The Generator loss is a combination of 3 losses: The Adversarial Loss, the Cycle Loss, and the Identity Loss. The Adversarial Loss is the loss of TaxiNet with the generated bad images and the given velocities. The Cycle loss is the loss with respect to the images reconstructed from bad images. This is done by running the input image through generator AB and then the generated output image through generator BA. The L1 is taken of the difference of the original input image and the reconstructed input image. The identity loss is a combination of two losses. The first is generator BA given the original image and taking the L1 loss of the generated and original images. The second is the same process but generator BA is switched with generator AB. They are combined with the weights 0.9 for BA and .1 for AB for the first model. The second model sets the weight to 1 and 0. This removes AB from the identity loss. The total generator loss is combined with weights of 10 for the adversarial loss, 10 for the cycle loss, and 5 for the identity loss. These weights were decided after trial and error. TaxiNet loss is the average of the network prediction with the bad and good images. The network uses MSE as its loss function.

2.6 Training

The model was trained using Pytorch as the framework. The model was trained on Google Colab with an NVIDIA A100 40GB. ADAM was used as the optimizer for training with a learning rate of 0.0002. The models were trained for five epochs each. The mini-batches in the epochs contained 20 images for 2240 mini-batches per epoch. Training time was slightly under 100 minutes.

3 Results and Discussion

We were successfully able to train two models using this method. The first model that used the AB identity loss had a slightly smoother loss graph and achieved a lower loss. This can be seen in a comparison of Figure 5 and Figure 7. This is not a surprising result as the adversarial images were much more similar to the original images in the first model. The adversarial loss seemed to converge for the first model much more than the second one. The second model had a much larger oscillating for the adversarial loss. The cycle and identity losses seemed to converge for both models.

The images generated by the two models are significantly different. The images with AB identity loss were much closer to the input images than the other. The no AB identity loss model had a blue tint through out the images and created weird patterns in the center of the images. The other model instead caused slight tweaks to the image. This included changing the corner of the image to yellow and

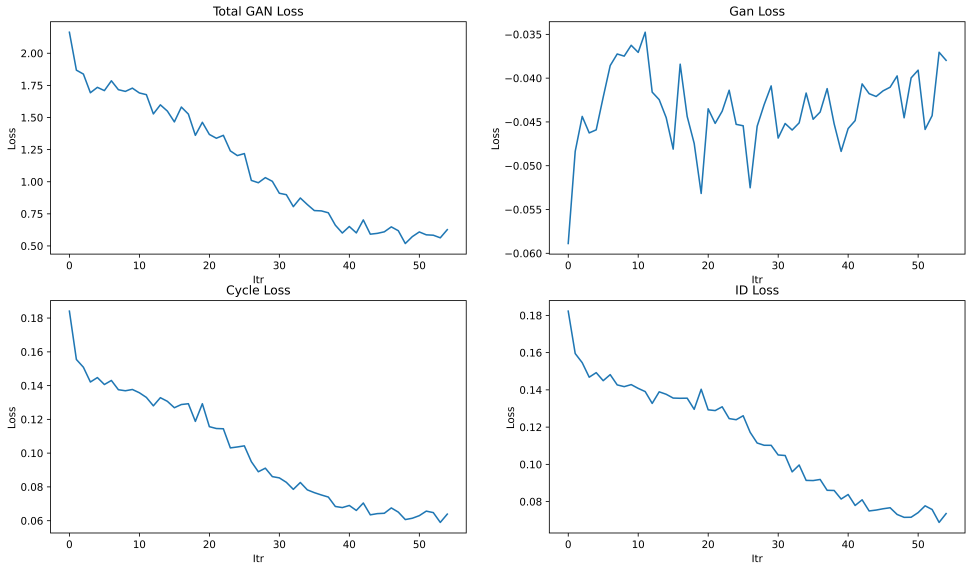

Figure 4: Generative Losses with AB Identity

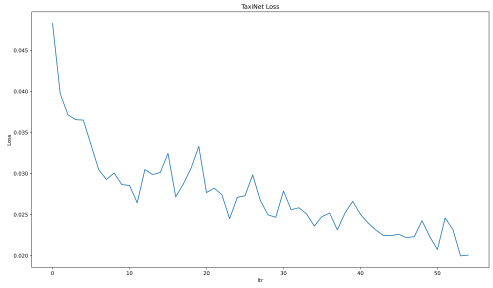

Figure 5: TaxiNet Losses with AB Identity

creating maroon and cyan highlights around the central and side white lines. This is interesting since it is seen in prior works that the TaxiNet relies on the position of these line markers to determine its position on the runway. And our model seems to provide adversarial artifacts to confuse the TaxiNet or add masks to the line marker to hide them from the images seen by the CNN. This results in an effective loss of tracking abilities of the TaxiNet network! Looking at the generated images and comparing them with the original also provides insights into the critical visual cues for the CNN.

4 Extension

An extension that can be done for this project is using a second domain to probe the generator to produce images closer to known failure cases. Currently, the model uses no or original images for the identity loss for generator AB. This could train the generator to create images even more effectively at fooling the network. If these images are scarce, having a random chance of replacing the original image with a bad image will allow it to learn this behavior without overfitting. Another extension is to update the generator architecture to follow the larger modern GAN models like GigaGAN. This would require more training resources but could produce much better results.

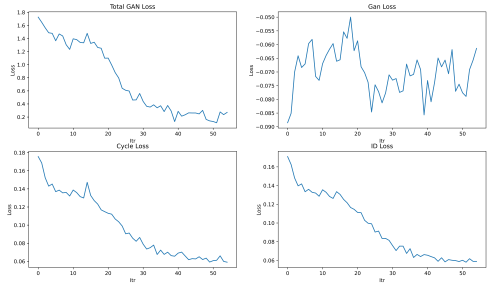

Figure 6: Generative Losses with no AB Identity

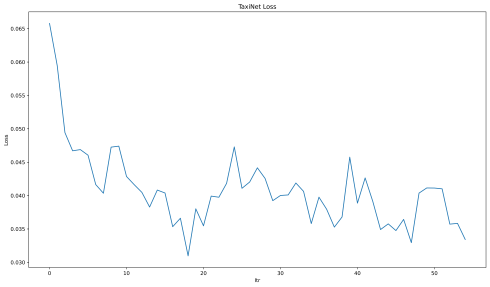

Figure 7: TaxiNet Losses with no AB Identity

5 Conclusion

We have designed a generative model that can be used to create an adversarial dataset for vision-based robot navigation pipelines. Such datasets can be used to augment the training of navigation networks to generalize better and remedy some of their common failure cases. This work would be beneficial in finding the long tail of failure in commercial autonomous systems like self-driving cars, indoor robots, and autonomous drones. Training of GANs, however, has been traditionally known to be tricky. Diverging losses and exploding gradients make it challenging to provide proper error bounds. Hence we envision this to be the most uncertain part of our project and could result in unsatisfactory results. Our results show that we can indeed generate input images that could cause an adversarial attack on the base network.

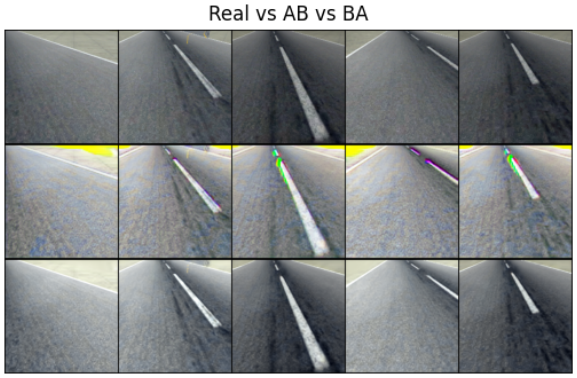

Figure 8: Comparison of Original and Generated images for Model with AB identity Loss

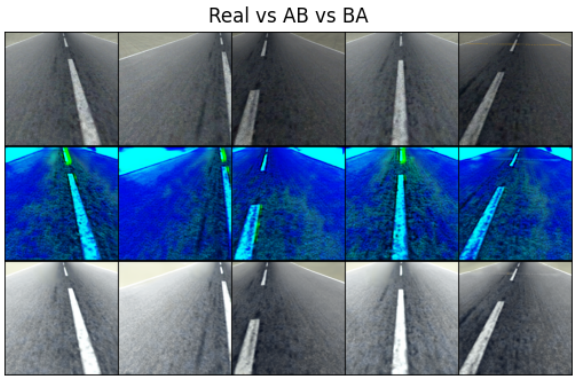

Figure 9: Comparison of Original and Generated images for Model without AB identity Loss

References

[1] Somil Bansal, Varun Tolani, Saurabh Gupta, Jitendra Malik, and Claire Tomlin. Combining optimal control and learning for visual navigation in novel environments. In *Conference on Robot Learning*, pages 420–429. PMLR, 2020.

[2] Kaustav Chakraborty and Somil Bansal. Discovering closed-loop failures of vision-based controllers via reachability analysis, 2023.

[3] Et al. J Y Zhu. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2014.

[4] SM Katz, A Corso, S Chinchali, A Elhafsi, A Sharma, M Pavone, and MJ Kochenderfer. Nasa uli aircraft taxi dataset. *2021, stanford Research Data*, 2021.

[5] Sydney M Katz, Anthony L Corso, Christopher A Strong, and Mykel J Kochenderfer. Verification of image-based neural network controllers using generative models. In *DASC*, pages 1–10. IEEE, 2021.

[6] Laminar. Laminar Research: X-Plane 11 (2019). https://www.x-plane.com/.

[7] S. SongSong. CycleGAN Tutorial from Scratch. https://www.kaggle.com/code/songseungwon/cyclegan-tutorial-from-scratch-monet-to-photo.