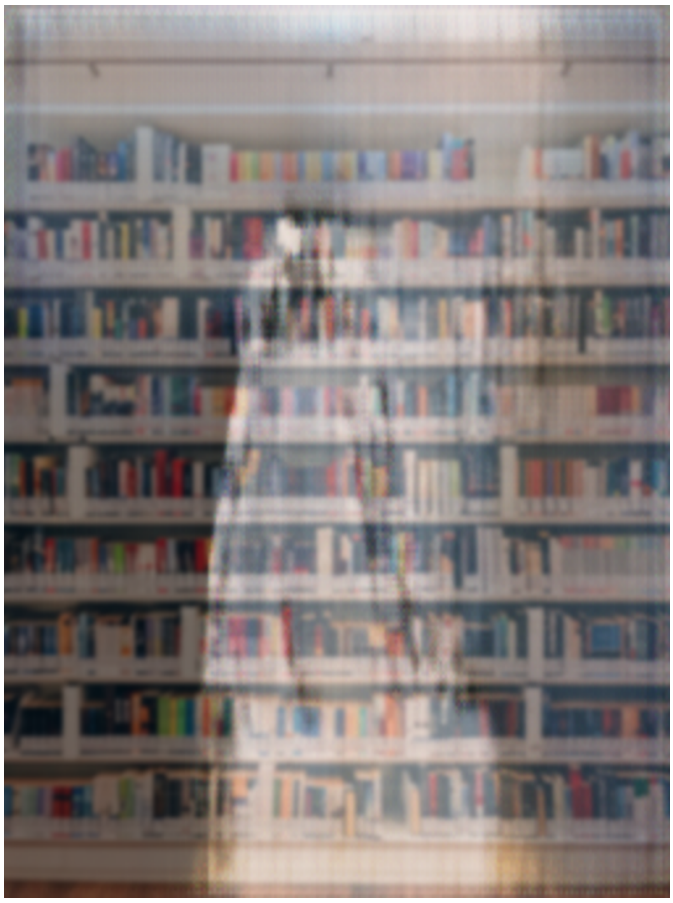
Art and Machine Learning
CMU 2022 Spring
Project 2

Ghost Lens



Group 15
Elena Gong (yezheng), Zixuan Zou (zixuanz)
Carnegie Institute of Technology
Master in Electrical & Computer Engineering
(Background applies to both members)

Description

Concept

Not everyone believes in the existence of ghosts, but we hear about the stories of ghosts or see them in various forms of artworks from time to time. In the depictions of these stories, ghosts are always considered to be related to fear, sadness and horrors. Despite knowing that ghosts are the embodiment of the human soul after death, we still feel the distance due to the difference between humans and ghosts. However, just like people who are alive, ghosts may just switch to a new form of life and continue to exist in this world, and they have their own normal emotions and lives just like humans. Therefore, through this project we want to get rid of the stereotypes of ghosts and provide people with a chance to take a look at ghosts from a different perspective - through Ghost Lens, we are able to view ghosts just as all the other creatures around us. In general, we want to discover the existence of ghosts in empty scenes around us and portray their figures in the scenes. We hope that this project will remind the audience of the existence of ghosts, change people's perception of ghosts, their fear of them, and see their presence as another form of companionship and a new routine of daily life.

Technique

Image scraper - We tried to use RiddlerQ's Google Image Downloader [1] to scrape images with specific keywords.

Edge extraction - We preprocessed the images using OpenCV's Canny Edge Detection method cv2.canny(), and Adrian Rosebrock's auto_canny() method [2] to automatically select the maximum and minimum threshold values based on the median of the single channel pixel intensities.

Augmentor - We used Augmentor [3], an image augmentation library in Python, to perform various operations on our input images, such as crop, flip, distort, and resize.

pix2pix - We trained several pix2pix [4] models over various sets of input images, mapping from objects to ghosts, and from extracted edges of different types of ghost images to the original images.

Random Ghost Image Generator - We drew several images of ghosts using lines for better edge extraction results, and used numpy's random choice to select one of these images and Python's Pillow library to paste the image on a white canvas with size of background images.

Image Filter - We generated the final results through resizing and adjusting transparency of the output image from the pix2pix model and applied it as the filter to the original background image using Python's Pillow library.

Process

We first tried to use RiddlerQ's Google Image Downloader to collect images with the keyword "ghost" from the internet, since we could not find any existing dataset of ghost images that we wanted. However,

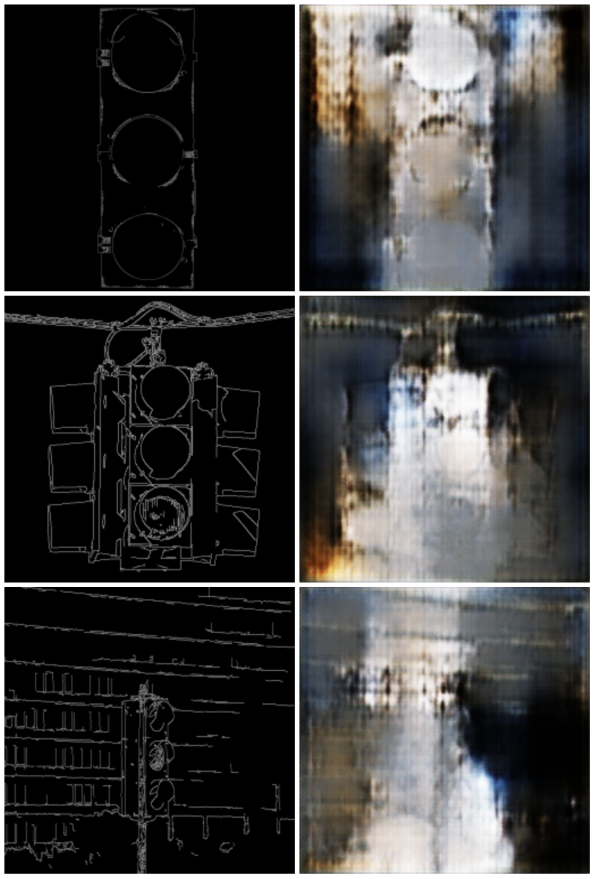
this image scraper could only download the images on the first page of the google search results. The images from the google keyword search also did not meet our requirements: most of them are either too gory, creepy, horrifying, or weird. Our goal is to create ghosts that blend into the images, as if they exist in the real world, so we would not want to use images that are too scary as our training data. Thus, we decided to collect our own dataset of ghost images.

We collected ghost images with different styles, ran edge extraction on them, and compared the results. We found that ghost images that have clear edges and are not too scary best fit our needs. Here are some sample images from our dataset:

Since searching the internet to collect ideal ghost images by ourselves was pretty time consuming and disturbing - there were many weird images online, we ended up having a relatively small dataset. For better training results, we did some research and found that we could apply data augmentations to our training set to increase the size [5]. We used Augmentor, an image augmentation library in Python, to apply various augmentations to the images in our dataset, such as random distortion, rotation, and resizing, etc. In the end, we had 300 ghost images in our training set.

Original Distortion Rotation Cropping

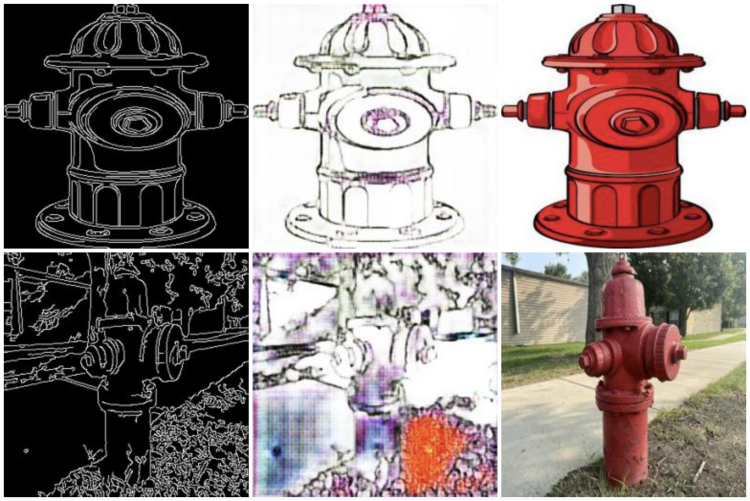
At the beginning, we wanted to add ghosts into images naturally by mapping some common objects in our daily life to ghosts. We collected a dataset of images of traffic lights, and trained the pix2pix model to learn a mapping from the extracted edges of traffic lights to ghost images. We were hoping the trained model could find some underlying correlation between traffic lights and ghosts, and it simply did not work. Here are some results from this failed attempt:



As we can see from the figure above, the trained model failed to produce meaningful results based on the input of traffic lights edges, which made sense because there was not enough correlation between them. So we decided to train models to map from extracted ghost edges to ghosts.

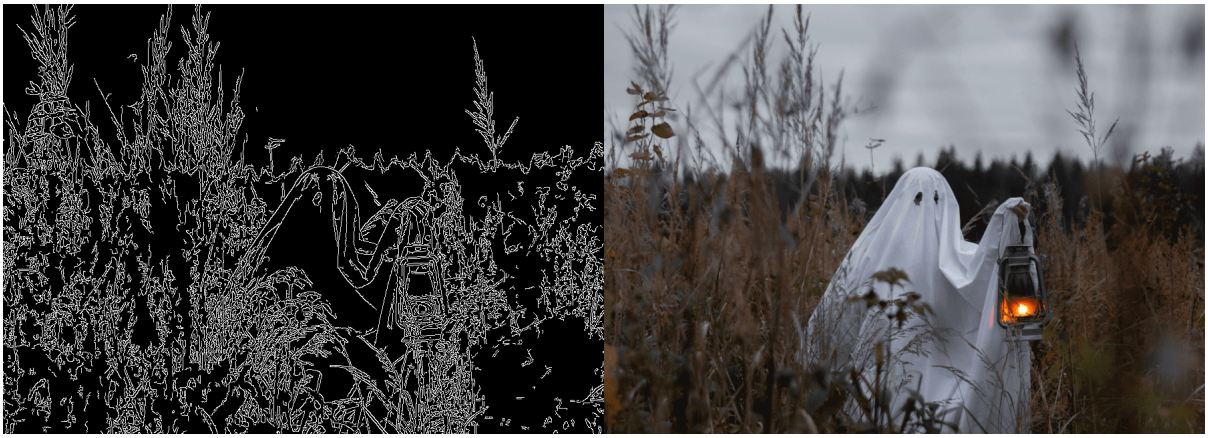
We then attempted to train the models with cute ghost images that map from extracted ghost edges to ghosts. Compared to the ghost images examples shown above, these images usually have monochrome backgrounds and very clear edges; the ghosts also look very similar, with white body and two black holes as the eyes. We hoped that the model would be able to learn about these features from the selected input images.

From the figure above that shows the result of the validation dataset after training 50 cute ghost images for 200 epochs, we can see that the model successfully recovered the ghosts with details. However, when we tested with images that took fire hydrants as the main subject, the model failed to detect the fire hydrants as ghosts and only recovered the edges in the image, as shown in the figure below.

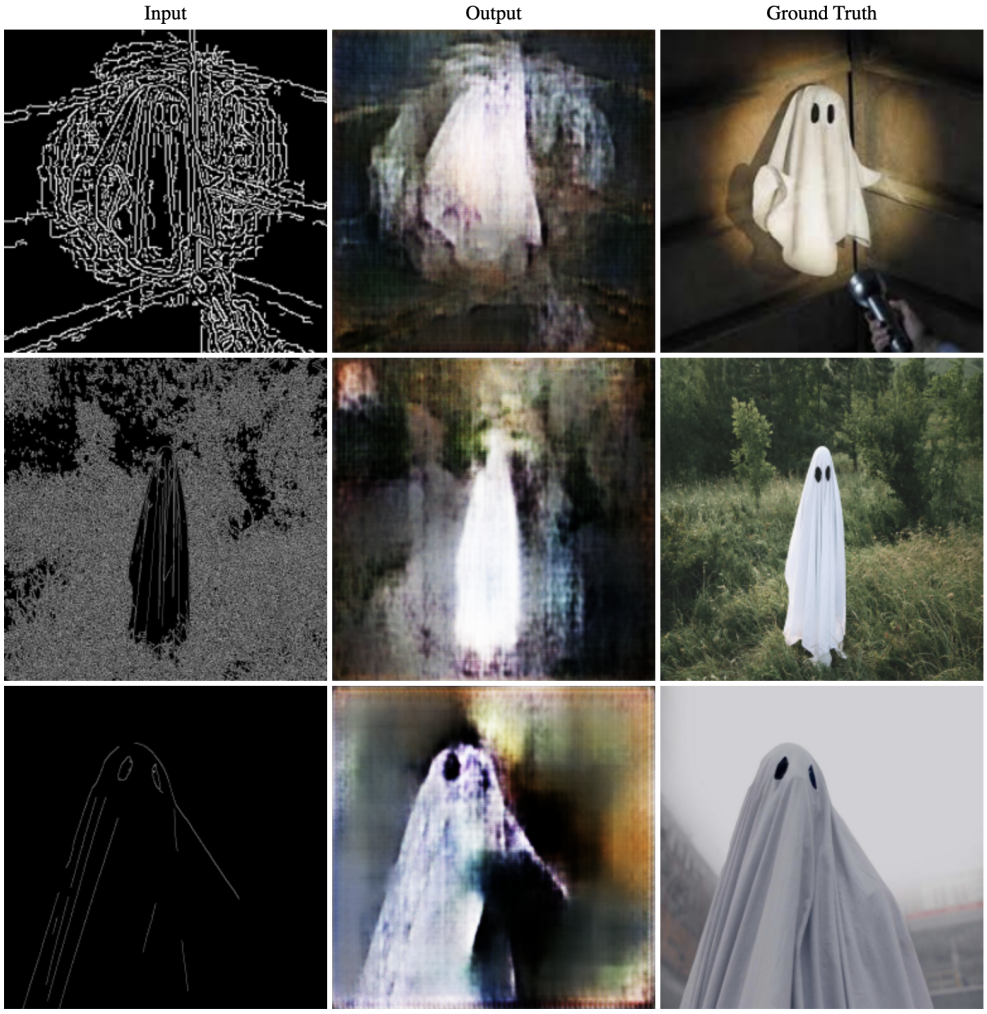


From these results we noticed that the cartoon ghost images might not be the best choice for training data as we thought. Thus, we decided to switch back to the ghost dataset we collected and augmented and eventually used it for our final results.

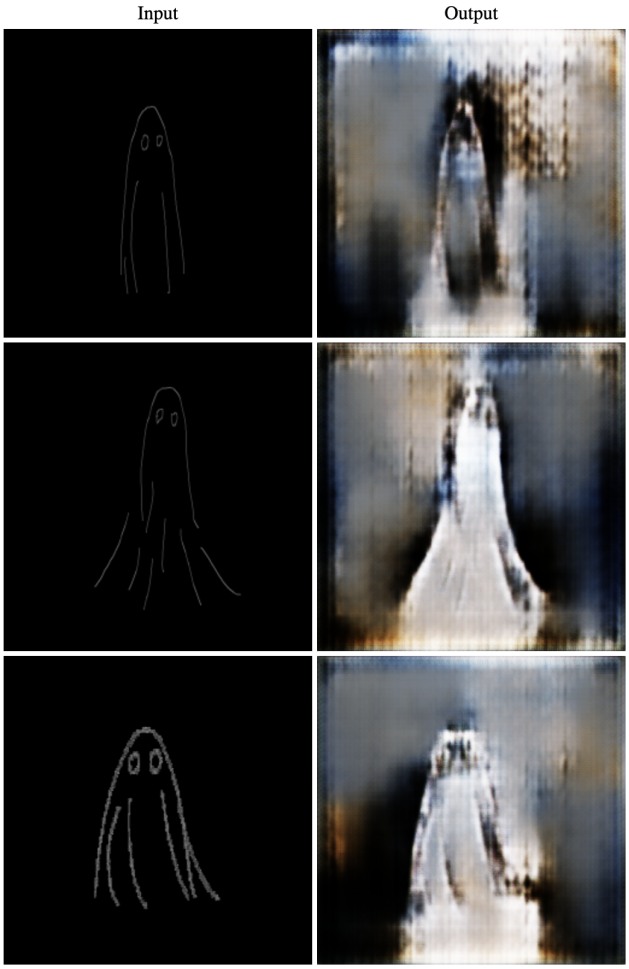
We then used extracted ghost edges and the original ghost images as our training set for pix2pix.



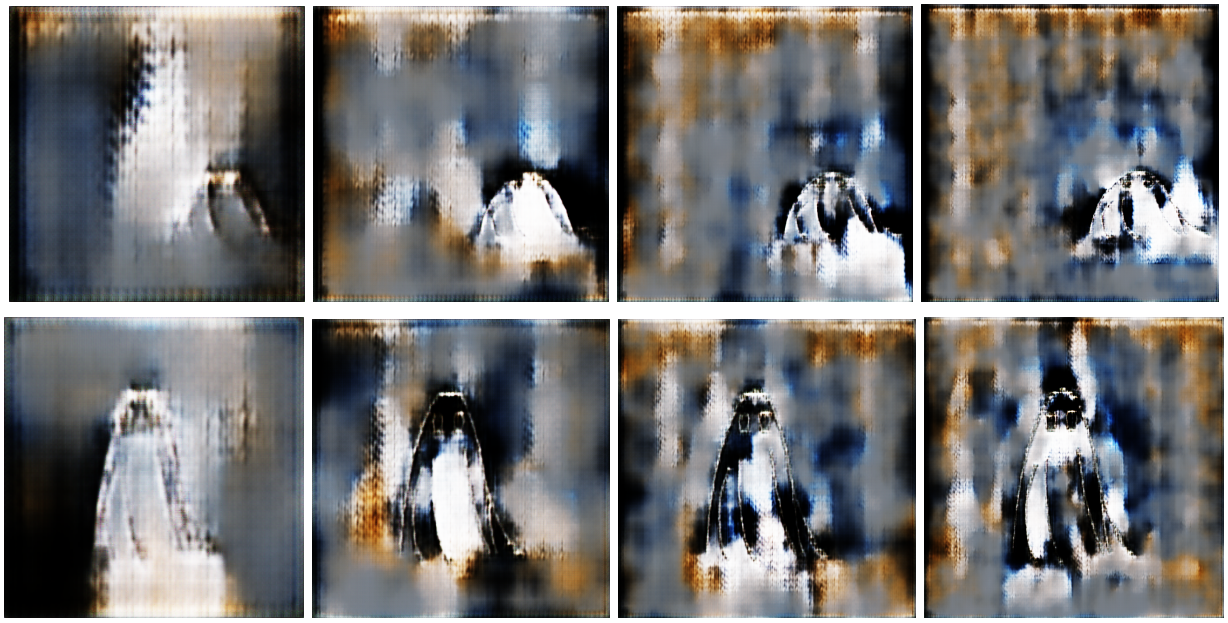
There are 300 images in our dataset, and we used an 80/20 ratio for training and validation. After running for 100 epochs, the trained model was able to map edges of ghosts to ghosts for our test set, as shown in the figure below:



We then experimented with different styles of hand-drawn ghosts - we drew some ghosts by hand, ran edge extraction, and used them as the input. We found that ghosts with simple shapes and clear edges had the best output from this model.

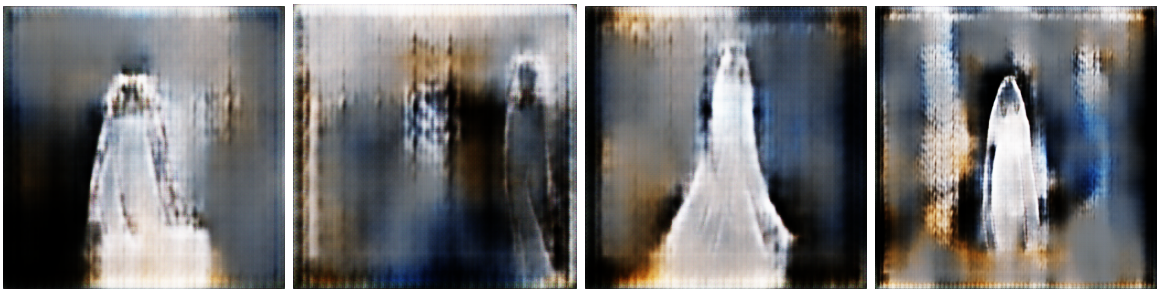


We then experimented with the loadSize and fineSize parameters in the code to test with different output sizes. The output images are 256x256 by default, and we compared the results of sizes 512x512, 768x768, and 1024x1024.



256x256 512x512 768x768 1024x1024

By comparing the output images with different sizes, we noticed that even though the locations of the ghosts can be more easily detected by the trained model for images with higher resolutions, images with lower resolutions seemed to be more natural. We also noticed that some images may look the best with lower resolution, while the others may look the best with higher resolution. We ran the tests for all sizes and models, and selected the best results for later steps.



For the final results, we first tried to generate a white canvas that matches the size of each background, and randomly selected one of our hand-drawn ghosts and also randomized the size and location of this ghost on the canvas. However, this approach did not work well because sometimes the ghosts were resized to small figures or placed at the corner of the canvas so that they were not detected and portrayed correctly. In our previous experiments, ghosts were usually detected successfully when they were the main subjects of the images or they were set at the center of the images.

We next tried to directly resize the best output images to match the background images with different combinations of ghosts and scenes. We found that some of the results looked harmonious, but some resized ghost outputs took up too large a proportion of the entire image, and the location of them didn't match with the background. In order to further improve our results, we generated a "ghostish" background

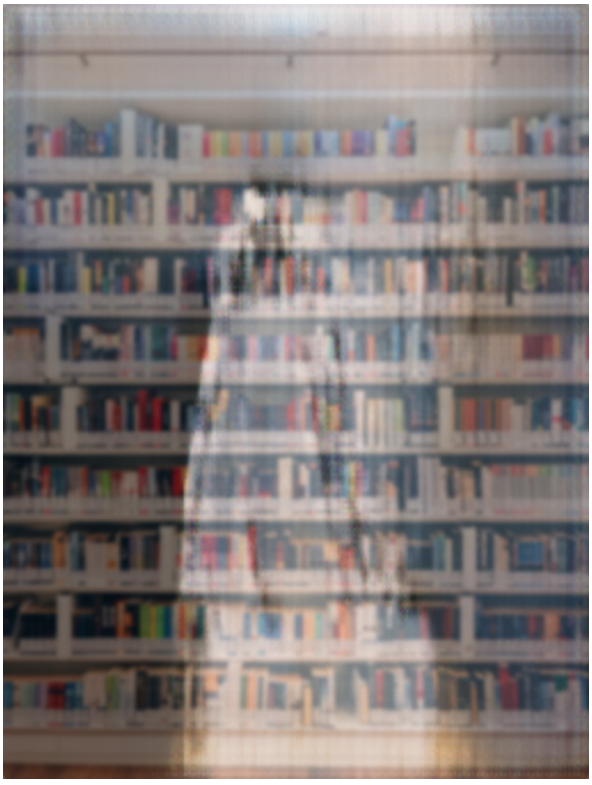
filter by using a white canvas as input. We apply this image as a filter to the background images; then we resized the ghosts and placed them in the image according to the scene and the story it may evoke. Through this method we were able to generate several meaningful images that showed the fusion of ghosts and scenes that the audience can imagine the story behind.

Reflection

We chose the final results based on the following criterias: the position of the ghosts should fit the other objects and characters in the image; the ghosts should naturally blend into the background; the scenes that images depicted should show resonance. In order to achieve these goals, we tried many different techniques that we learned inside and outside the class. We believe that the technique that we chose for our final results served well for our project idea.

During the process of experiments and research, we came up with the new idea that we can also detect the objects that can potentially be transformed into ghosts, which could be a future research idea that would focus more on generating horror images related to ghosts. We have learned a lot about the techniques we used and explored during the process, and more importantly we learned that we need to flexibly adjust our goal according to the problems and available resources we have to achieve the best result we can expect. In terms of our project, we also become more convinced that ghosts are just like all of us, and they are also living their lives around us.

RESULT



CODE

https://github.com/elena-hanni/10615-Project2-GhostLens.git

REFERENCE

[1] “RiddlerQ/simple_image_download: Python script that lets you auto download images from google images using tags.” *GitHub*, https://github.com/RiddlerQ/simple_image_download. Accessed 26 February 2022.

[2] Rosebrock, Adrian. “Zero-parameter, automatic Canny edge detection with Python and OpenCV.” *PyImageSearch*, 6 April 2015, https://pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/. Accessed 26 February 2022.

[3] “mdbloice/Augmentor: Image augmentation library in Python for machine learning.” *GitHub*, https://github.com/mdbloice/Augmentor. Accessed 26 February 2022.

[4] “phillipi/pix2pix: Image-to-image translation with conditional adversarial nets.” *GitHub*, https://github.com/phillipi/pix2pix. Accessed 26 February 2022.

[5] “Ask for help about the size of dataset and the num of epoches. · Issue #283 · junyanz/pytorch-CycleGAN-and-pix2pix.” *GitHub*, 31 May 2018, https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix/issues/283. Accessed 26 February 2022.