Artificial Intelligence for the Media
Mini-Project Brief
By Marissa Beaty

The goal of this project was to train a Pix2Pix model on a paired image dataset of architectural buildings taken from Pinterest using PinDown and simplified black-and-white images of those same buildings but run through an edge detection model. The projects purpose was to give the model an image of a building that has been edge detected and have it produce a realistic likeness of that building. As will be discussed later in this paper, the Pix2Pix model was unsuccessful in training on the given dataset. After several attempts with varying Pix2Pix models yielding no results, the data trained on a CycleGAN model. The CycleGAN model successfully trained to give preliminary results with the given dataset.

The Pix2Pix model I had initially chosen - and found in the first notebook - to use on my dataset was based on the research of Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros and adapted into a Pytorch notebook by Aniket Maurya. This model utilized a U-Net-based structure, as is typical for a Pix2Pix model. To get this model to work, several tweaks needed to be made that read and split the dataset into two image tensors (one for the real images and one for the condition or edge detected images). When attempting to train this model, however, several errors ensued, most commonly an error stating, "DataLoader worker (pid(s) 74234) exited unexpectedly." After researching this specific error, I concluded that it stemmed from the DataLoader's inability to read the images into the model in the correct format. The images were read into the model as tensors organized by width, height, and RGB. The model required them, however, to be organized by RGB, width, and height. However, attempts to restructure the image proved fruitless.

This leads to my second notebook and second model. This model was adapted from one posted on OpenCV, which reduces the number of classes defined and called by the model. I had theorized that perhaps the difficulty with reading the images correctly in the previous model was due to the difficulty of the DataLoader reading and cycling through the class that defined and created the tensored images. This model proved effective until it got to training, in which I received the following error message "ToTensor.__init__() takes 1 positional argument but 2 were given." Similar to the first notebook/model, this error was referencing the DataLoader, leading me to believe that eliminating the class was not successful in solving the DataLoader issue. I attempted to fix the error by reducing the number of images and pulling the paired images from separate folders rather than import them as one image that then gets split. I learned quickly, however, that successfully implementing these changes would require the rest of the model to be restructured and recoded. Due to the timeframe of this project, this was not a feasible endeavor.

The next model I attempted was that written by the creators of the Pix2Pix model: the research team of Phillip Isola, et al. This model is within a GoogleColab notebook. Though the colab notebook worked initially, I again found an issue when uploading my dataset into the model, particularly with the inability to find the directory that housed my data and download the model necessary to train the data. Research into these issues provided little understanding. It appeared the issue was common, but the solution remained unknown.

In one last attempt to utilize a Pix2Pix model, I adopted the model posted by Aryan Esfandiari on a personal blog, documented in the fourth notebook. This model implemented Pytorch Lightning and established a different way of splitting the image into a real and condition set. In training this model, however, I came upon a new error only halfway through the code that stated, “Can’t pickle local object ‘SketchDataset.__init__.<locals>.<lambda>’.” This error seemed straightforward, requiring me to alter my object from a local to a global object. However, despite doing this, the error remained. Another recommendation was to bypass pickle by using Dill instead. This resulted in a similar error as when using pickle. Additional research on this error failed to provide a successful solution.

After four unsuccessful attempts at training a Pix2Pix model, I began to wonder if a CycleGAN might produce similar difficulties or would bypass the errors loading in a paired dataset seemed to have. The benefit of a CycleGAN model is that it does not require perfectly paired images, rather it imports files each time and trains itself based on separate folders. I utilized a CycleGAN model published to Kaggle by SW-Song. Due to time constraints, I trained this model with only 800 total images, 400 edge images, and 400 real images (or condition and real as referred to in the Pix2Pix model). Having only tweaked the code that read in the datasets, the model began training and produced 25 total images for each epoch and batch after approximately 23 total hours of training.

The results yielded from the CycleGAN model were not perfect. Though the model did improve over time, the model was designed to train on a dataset between 7,000 and 10,000 images and utilized a larger GPU than I had available. Given more time, I would have enjoyed training the CycleGAN model for longer with a larger dataset to determine its overall success in transferring edge images into realistic depictions and vice versa. Considering the limited dataset and timeframe, the images produced by CycleGAN are promising for the model’s future use. Furthermore, the ease of using the CycleGAN model far exceeds the Pix2Pix model. Though a Pix2Pix model is interesting by design, its usability proved narrow. As stated above, four models were attempted and manipulated without success. Though this is, in part, due to my limited knowledge of the model’s inner workings and structures, there was also little information available on the types of errors produced by the Pix2Pix model. Though I gained a far superior understanding of generative networks and their structure throughout this process, should I attempt this project again, I would begin with CycleGAN training on a larger dataset. I believe this would have provided more interesting results than I was afforded for this Mini-Project.

Works Cited

Pix2Pix Model used in Notebook 1
Aniket Maurya (2021) Pix2Pix – Image to Image Translation with Conditional Adversarial Network [Source Code]. https://librecv.github.io/blog/gans/pytorch/2021/02/13/Pix2Pix-explained-with-code.html.

Pix2Pix Model used in Notebook 2
Aditya Sharma (2021) Pix2Pix: Image-to-Image Translation in Pytorch and Tensorflow [Source Code]. https://learnopencv.com/paired-image-to-image-translation-pix2pix/.

Pix2Pix Model used in Terminal for Trial 3
Phillip Isola and Jun-Yan Zhu et al, (2017) CycleGAN and Pix2Pix in Pytorch [Source Code]. https://colab.research.google.com/github/junyanz/pytorch-CycleGAN-and-pix2pix/blob/master/pix2pix.ipynb#scrollTo=9UkcaFZiyASl.

Pix2Pix Model used in Notebook 4
Aryan Esfandiari (2021) Pix2Pix with Pytorch and PytorchLightning [Source Code]. https://www.aryan.no/post/pix2pix/pix2pix/.

CycleGAN Model used with successful results
SW-Song (2021) CycleGAN Tutorial from Scratch: Monet to Photo [Source Code]. https://www.kaggle.com/code/songseungwon/cyclegan-tutorial-from-scratch-monet-to-photo.