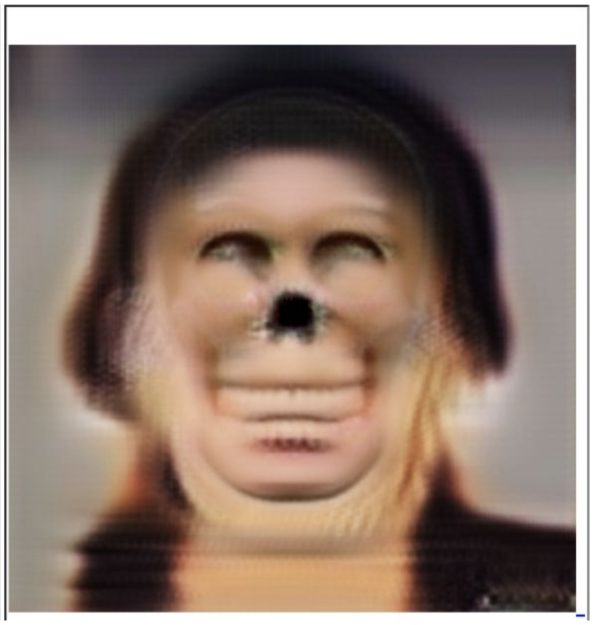
ART AND MACHINE LEARNING
CMU 2019 SPRING
PROJECT 2



Prajwal Prakash Vasisht, Sunil Kumar, Yashovardhan Chaturvedi

Concept:

We went into this project with an open mind and the intent to understand the practical working of GANs. Pokemon was something that all three of us were attached to as kids and so our first thought was to explore how a computer would create a pokemon out of an existing animal, just as the creators of Pokemon, Satoshi Tajiri, Ken Sugimori drew inspiration (mostly) from real-world animals.

Technique:

To accomplish our goals, we first obtained a dataset of pokemon images from Kaggle. These images were primarily composed of in game sprites of the pokemons pulled from the early pokemon video games. In addition to our pokemon dataset, we also utilized the dataset of horses from the class GitHub repo. With regards to the skull to face project, we performed some simple web scraping off of Google images to obtain a relatively small dataset of skull images. For the faces, we used a curated set of 1,000 images from the CelebFaces database, a collection of more than 200,000 celebrity faces. These images were chosen at random from the super set.

In order to address the problem we explored Cycle Gan for unpaired image to image translation. Cycle Gan is a technique that learns mapping as well as an inverse mapping between two class of images by utilizing cyclic loss between two GANs. It is the state of the art technique for addressing image to image translation. We used the official GitHub repository that is open sourced by the authors of the paper. Should we mention you used unet here?

Process and Results:

Our initial experiments with the Horse to Pokemon model started by simply feeding our datasets to the CycleGan notebook. After letting the model run for 40 epochs, we obtained the images in Figure 1a. Interestingly, we noted that the CycleGan was failing to identify the outlines of the horses in the original images. However, the cutouts in the generated images had unique and plausible outlines of new and original pokemon. To help alleviate the issue with the detection of the horses, we tried to limit the pokemon sprites in the dataset to just pokemon with the same general silhouette as a horse, thereby hopefully allowing for a more precise fit on the horses. As you can see in Figure 1b, this was somewhat successful, but the bounds on the horses were still quite loose. We then tried this experiment again with a different dataset of pokemon images also sourced from Kaggle. This time, instead of using just the pokemon sprites, we included images of the pokemon from the cartoon as well. Our hope was that by placing the pokemon into an image with a background, it would more closely resemble the original horse to zebra task and potentially produce a better result. However, as seen in Figure 1c, this actually resulted in a more stylistic transfer on the original image instead of generating a new concept. The generated images also seemed to lose some of their focus on the horses themselves, with some images only focused on the background.

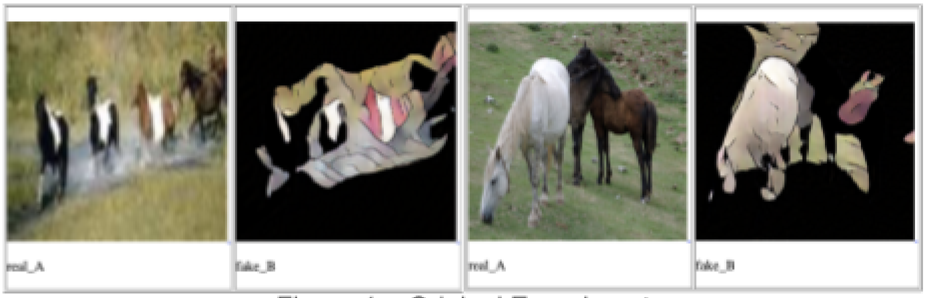


Figure 1a: Original Experiment

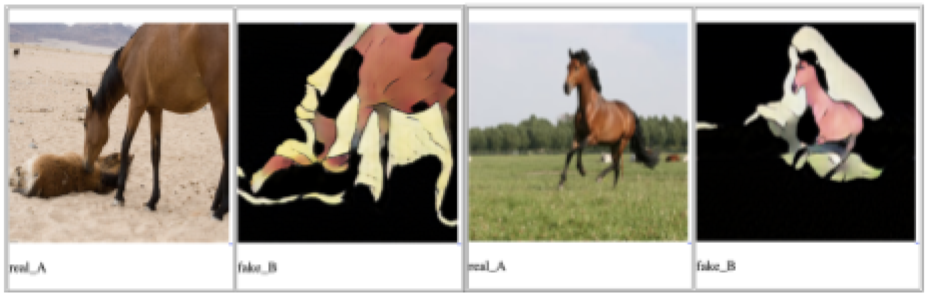


Figure 1b: With Pokemon similar to horses

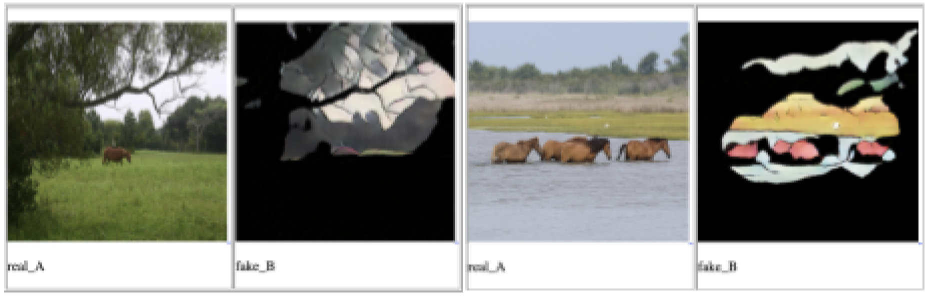


Figure 1c: With 2nd Pokemon Dataset

As we did not end up with very conclusive results for our first experiment, primarily due to the difficulty in obtaining a relevant dataset, we decided to pursue another idea we had initially. This idea was to try to generate faces from images of skulls to see if a computer could pick up on the subtle differences in skulls in order to generate faces that fit the size and shape of the skull. For our initial experiments, we again simply fed the collected skull dataset and CelebFaces dataset to our CycleGan. However, after training for 50 epochs, we noticed that there was very little variation to the images produced by the model, given a starter image. In addition, the results were also less than satisfactory in that very little change was being applied to the original skulls as seen in Figure 2a. To address this, we first cut down on the size of the skull dataset such that we had a smaller variety in angles for the skulls so that most were front-facing. This would allow the model to focus on reconstructing faces primarily from the front, which aligned with the orientation of most of the face in the CelebFaces dataset Then, to help offset the smaller data size, we introduced random noise to each skull image before it was fed to the model. The idea behind this was that the random noise factor would prevent the model from simply memorizing the output face for a given input image, thereby allowing it to be far more robust and hopefully more original. The results of this can be seen in Figure 2b.

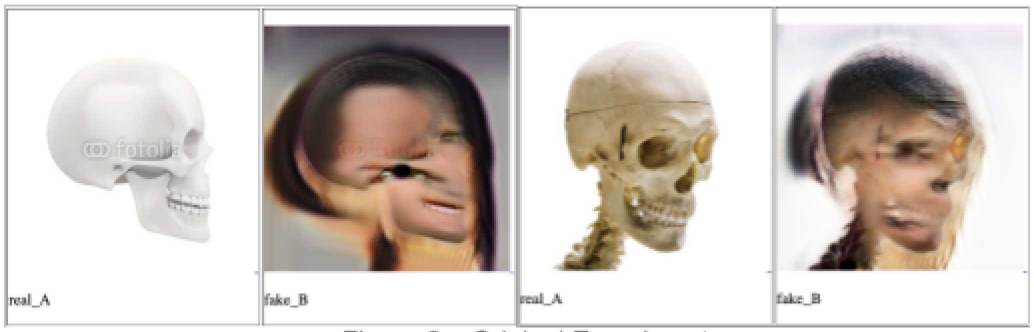


Figure 2a: Original Experiment

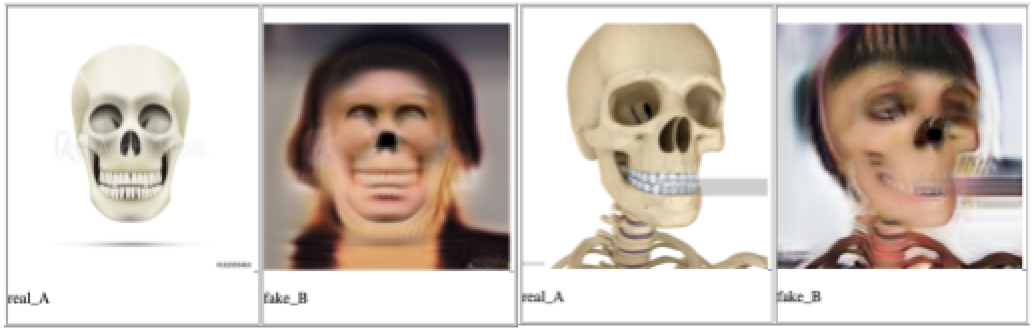


Figure 2b: Revised Experiment

Reflection:

Although the final images we ended up with aren't as convincing as we'd hoped for, we learnt a lot along the way. For each experiment, we trained the models for about 50 iterations, which translates to close to 20 hours per experiment. We believe that letting the models train for many more iterations could potentially improve the results, as we continued to see improvements even at the 50 or 60th epochs. Due to the number of experiments we ran, as well as restriction in terms of AWS credits, we unfortunately did not have the resources to do so. In addition to the huge training times, the need for a carefully curated dataset also stood out to us. With both the pokemon models and the skull to face models, we noticed that having a noisy/messy dataset resulted in huge performance decreases by the model over the same number of iterations. All in all, getting our hands dirty in trying to produce quality results helped us understand the impressiveness and notoriety of GANs. Although the pokemon experiment did not come out as well as we hoped, we are pleased with results from the skull to face experiments given that we had to collect our own dataset as well as the limited training time.

Code: https://github.com/yshvrdhn/CycleGan

References:

1. https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
2. https://www.kaggle.com/kvpratama/pokemon-images-dataset - Pokemon dataset
3. https://github.com/hardikvasa/google-images-download - Google Image Scraper