**I’m Something of a Painter Myself**

**Deep Learning Project Update**

**Austin Gadiner, Seth Dana**

**Introduction**

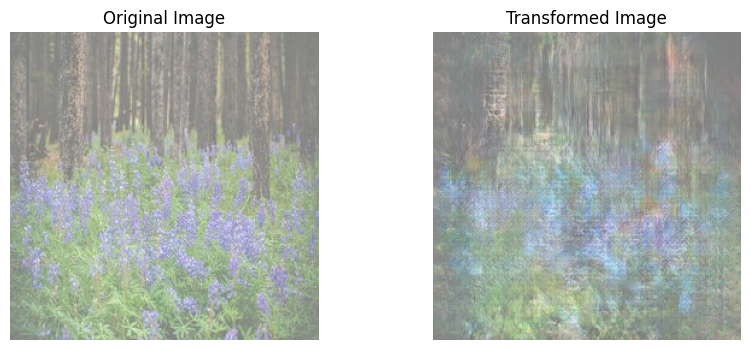
The goal of our project is to build and train a Generative Adversarial Network (GAN) that takes photograph images and transforms them to be in the style of Monet. We have a set of 7,038 images to transform and 300 painting images for the model to learn the style. The GAN works by simultaneously training two models, a generator, which transforms the images, and a discriminator, which classifies the generated images as either real Monet paintings or generated images.

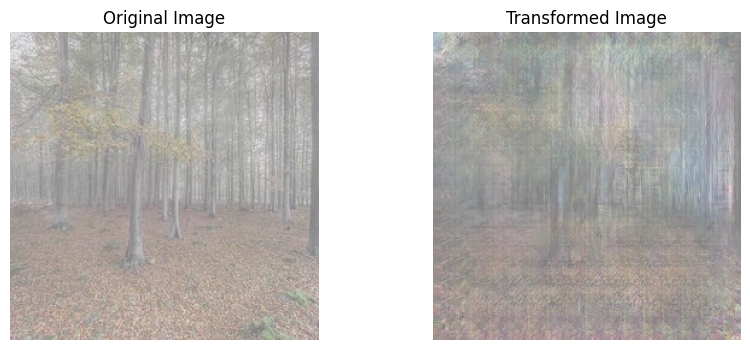
**Data**

We start by reading the image folders into the program and applying a transform that ensures all images are 256x256. In our final implementation, we plan to augment our data using random horizontal flips on Monet paintings. Additionally, we will separate the data into training, validation, and testing sets.

**Model Training Progress**

Unlike other projects, we won’t be comparing our model to basic machine learning methods. Instead we will build multiple GAN models and compare their performance to one another. Our first model is a self constructed GAN with convolutional layers in the generator and discriminator. We have this model working with the following results obtained during the training process. We set the model to train for 500 epochs, which took approximately 22 hours to run. If we need to retrain the model again, we will probably decrease the number of epochs as the improvement after about 100 epochs is marginal. Comparing the images in Fig. 1 to the example in Fig. 2, it is easy to see that the generator is improving during the training process.



****

**Fig. 1:** Original and transformed images after 380 epochs of training

****

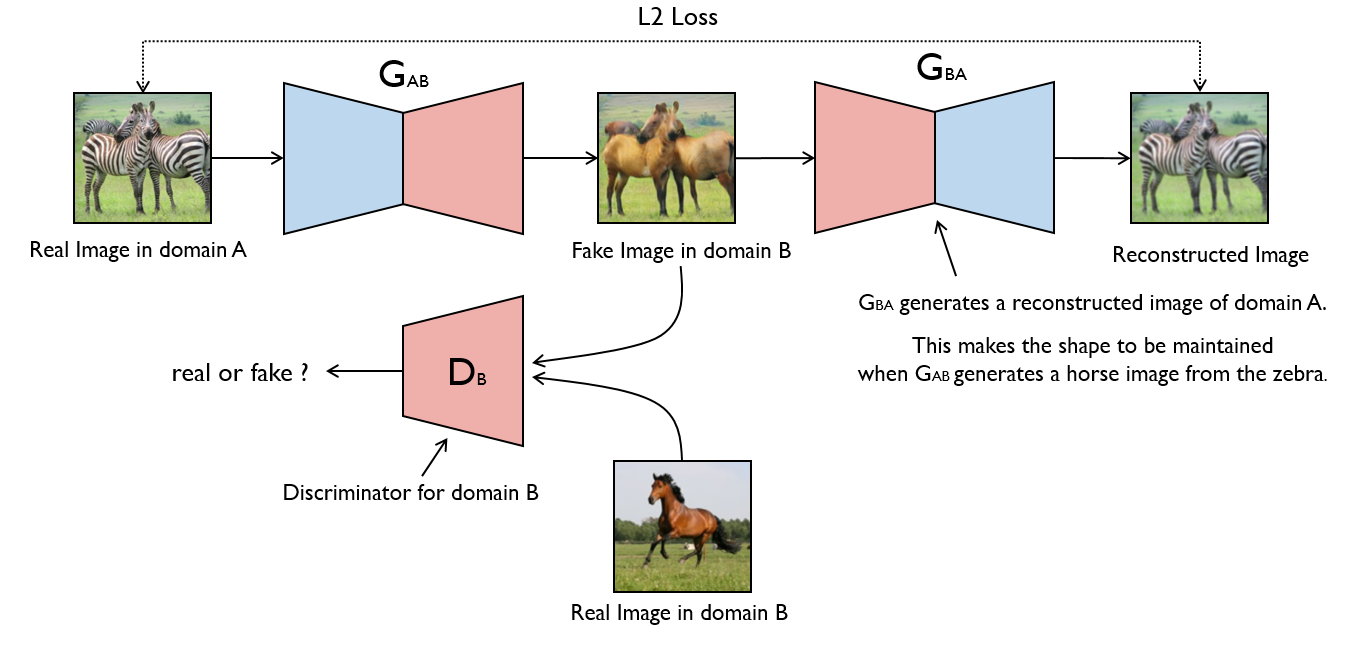
**Fig. 2:** Original and transformed image after 10 epochs of training

**Problems**

The biggest problem getting the model working was making sure the dimensions of layers in the generator and discriminator lined up correctly. Originally, the discriminator was not correctly doing a binary classification and the generator produced images that were not the same size as the input images. However, after some work to find the dimensions at each layer, both models were fixed.

**Planned Steps to Completion**

We have found some success in our foundational GAN model however we believe the transformed images could be better with a different GAN architecture. Our future work will include advancing to a more sophisticated GAN model called cycle GAN. Cycle GAN is designed for unpaired image-to-image translation, meaning, we don’t have pairs of data from both domains. Figure 3 shows a diagram of how cycle GAN works. There are 2 generators and 1 (sometimes 2) discriminators. In some cases, there is an additional discriminator to identify if the reconstructed image coming out of G\_BA is real, however, in this case a L2 loss is used to determine how well G\_BA is performing.



**Fig 3.** Cycle GAN architecture. For our project monet paintings will be domain B and the real photos are domain A.

**Team Contribution**

Both team members have contributed equally to the project. In order to test multiple configurations for the generator and discriminator, both team members independently created and trained their own models. When both models were able to run, the performance of the two models were compared and the better performing model was selected.