Machine Learning for the Arts
UCSD FALL 2019
FINAL PROJECT

Artifying Pictures

Benjamin Chang

DESCRIPTION

The goal for this project is to generate visual images that can have particular types of desired art styles or patterns. This can be useful for creative designers who wish to see what their artwork would look like in a particular style or with specific types of features or patterns. It can be used for the inspiration of new art styles as well. The method for this project is neural networks which is arguably the most popular machine-learning method to date. The three networks used in this project are DeepDream, StyleTransfer, and CycleGAN.

Additionally, some of the output of the DeepDream network are very interesting and could be the basis for some interesting interpretations. The resulting images yielded close to the hoped for results. The output images had interesting distortions added by the DeepDream network, cool pattern styles added by the StyleTransfer network, and noise removal by the CycleGAN network. Some future work for this project include finding a way to use different patterns from the DeepDream network and making the effect of the pattern from the StyleTransfer network stronger on its input image.

Concept:

Growing up, I loved art and took many art lessons. I would learn how to use different mediums such as pencil, color pencil, acrylic, chalk, watercolor, and more. It fascinated me how many diverse art styles each medium could be used for. While looking for inspiration online, I came across some images of Native American paintings of the impasto style which reminded me of how it is one of my favorite painting styles. The impasto painting technique where paint is laid in a thick layer on the surface of the canvas so that the brush texture shows up strongly. Seeing these images inspired me to want to crate my own art style conversion generator which both others and I can use to manipulate drawings to have the appearance of different styles or mediums.

Technique:

The primary technique used for this project is neural networks. In particular, the details of the CycleGAN network will be discussed. The model architecture is composed of two generator models and two discriminator models. One of each is used to go from the input image to a target domain and another one of each is used to go from the target domain back to the input domain. Specifically, the generators generate images going from one domain to another and the discriminators predict whether the generated images are real or fake. All 4 models are trained like GAN models. An identity mapping generator is also trained and used so that input images transformed to the target domain can be reverted back correctly.

The discriminator model uses least squares loss (L2 Mean Squared Error) with a one-half weighting to how quickly it updates the model. Using one-half weighting helps slow how quickly the discriminator updates compared to the generator when both are being trained. This works best since the generator model is more complex. In addition to the L2 adversial loss, the generator models also update based on an identity loss (L1 Mean Absolute Error), a forward cycle loss (L1 Mean Absolute Error), and a backward cycle loss (L1 Mean Absolute Error. Cycle loss is weighted 10-times that of the adversial loss and identity-loss is weighted half of the cycle loss. On the other hand, the generator model updates based on the loss output of the discriminator model which it tries to minimize. It also updates based on how well it is able to regenerate the original input image.

Training/Testing images were collected with the following directory structure:

```
Human2animal
├── testA
├── testB
├── trainA
└── trainB
```

Here, ‘A’ represents the input domain and ‘B’ represents the target domain. The dataset is zipped and later unzipped for use. In this project, the input domain is humans and the target domain is animals.

The model treats the number of dataset images corresponding to ‘A’ as the number of batches which is also the number of training iterations. It saves the model every five epochs which is five times the number of training iterations. The number of epochs attained the desired result is dependent on the output quality one is looing for and how complex the mapping from the input domain to the target domain is for the model to learn. The code finally randomly selects images from the dataset of the ‘A’ type or a custom dataset to test the mapping of the images from the input domain to the target domain and vice-versa with a desired model.

Process:

This project was developed in several phases. The various stages included: idea brainstorming, planning and consideration of feasibility, network selections, parameter selection, code debugging, preparing a training/testing dataset, training the networks, and finally testing the networks. Some of these stages took much longer than others, particularly the planning stage and the code debugging stage. It was quite surprising that planning took quite as long as it did and showed me the importance of developing a solid idea and plan before launching into the technical aspects of a project. Part of the reason it was difficult to come up with a project I could complete in a short window is the limited knowledge I’ve gained regarding machine-learning through this course.

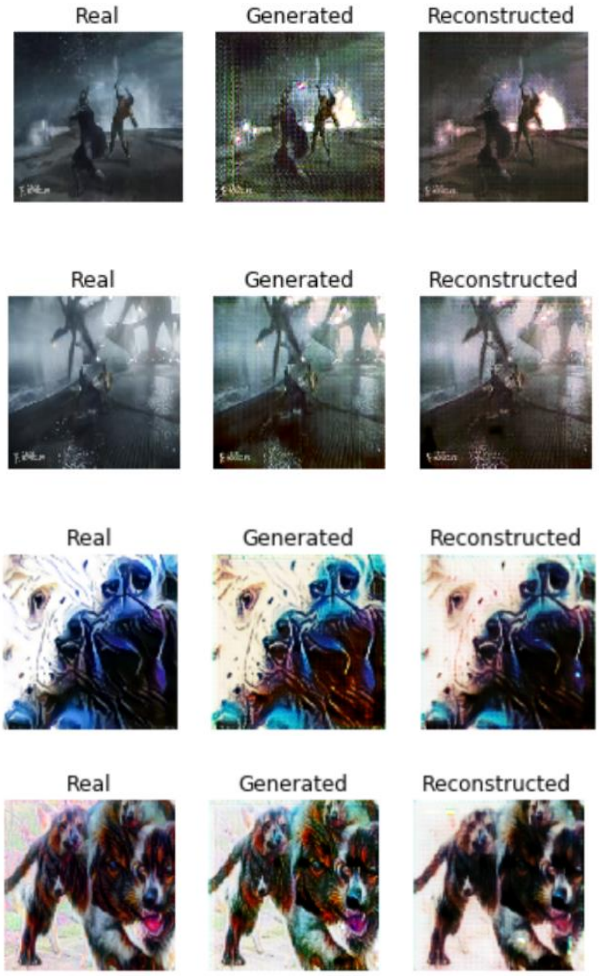
Selecting networks was also difficult as many of the models suggested in course material was buggy and difficult to run. As a result, I used Google as a means for selecting network implementations which were easier to setup and run. Code debugging was a large pain as much of the code which was a couple years old at least constantly had problems being compatible with various things such as tensorflow version and other outdated modules/code.

Result:

The resulting output images of this multi-network system is shown below. For the CycleGAN network used in this final project, it is notable that when the images are passed through models which have been trained on fewer iterations, the output images are more blurry and also more different from the original input image. The opposite is true for models which have been trained for more iterations.

Final Result:

Below are some interesting output results.



Reflection:
This project is quite interesting and works well, but it is difficult to determine how the images will turn out after passing through each network. It is definitely much better suited for artistic exploration than for rigid, technical art with a particular goal. Perhaps in the future, a version of this project can be created which allows for greater control over or predictability of the outcome of this stylization image generator.

REFERENCE:
Brownlee, Jason. *How to Develop a CycleGAN for Image-to-Image Translation with Keras*, Machine Learning Mastery, 8 Aug. 2019, https://machinelearningmastery.com/cyclegan-tutorial-with-keras/.

CODE:
https://github.com/ucsd-ml-arts/ml-art-final-benchang

LINKED RESULTS:
Video Results
- Aquaman Clip
https://www.youtube.com/watch?v=jwP2OJn-2Cc

- Processed Aquman Clip
https://www.youtube.com/watch?v=Mt5g_ZIgcpI