**Technological Institute of the Philippines**

938 Aurora Blvd., Cubao, Quezon City

**College of Engineering and Architecture**

Electronics Engineering Department

Excercise

**Generative Adversarial Network**

Submitted by:

**Gabotero, Clive Jake A.**

ECE41S1

Submitted to:

**Engr. Christian Lian Paulo P. Rioflorido, MSEE**

October 2022

It is the main goal in this activity to show an application of the Generative Adversarial Networks (GANs). Moreover, it is also serves to understand how this neural network works.

As the name implies, the Generative Adversarial Network is composed of two main parts, namely the generator network and the discriminator network. The generator’s role is to generate a fake image out of a random noise initially and then passes it through the discriminator. The discriminator then assesses the image from the generator and decides if it is fake or real. As the evaluation comes out from the discriminator, the feedback is then taken again by the generator for it to be able to generate a better image. This process is called the backpropagation. These processes will continue to happen until the generator is able to produce an image that is deceiving enough to fool the discriminator that this image is real even it is a fake one.

In this activity, an application of Cycle GAN which is the image-to-image translation is demonstrated and images of horses and zebras were used as dataset. Cycle GAN uses the concept of unpaired data, meaning, the input image is not related to the label or the style of what the output would be. An example of this idea is the neural style transfer where an input is an image and the other signifies a style. The idea of using this type of GAN will be further elaborated in this activity.

Below is the code to implement the image-to-image GAN.

**Image – to – image GAN:**

To start the Generative Adversarial Network, it is important to load the necessary libraries.



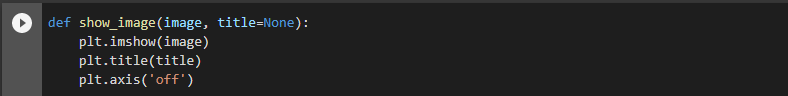
A picture containing text

Description automatically generated

Text

Description automatically generated

In these next lines of code, functions are being customized so that it can be plotted using Matplotlib.



Since Google Colab is being used in this activity, the dataset is uploaded to google drive where it can be imported using the next lines of codes.

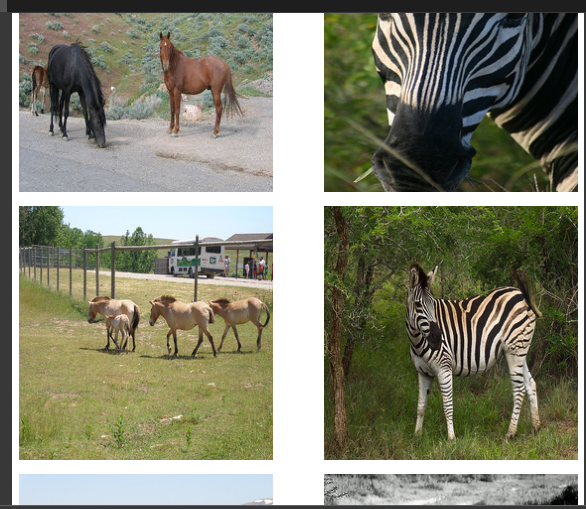
Text

Description automatically generated

Now we visualize the images to see what they look like.

Text

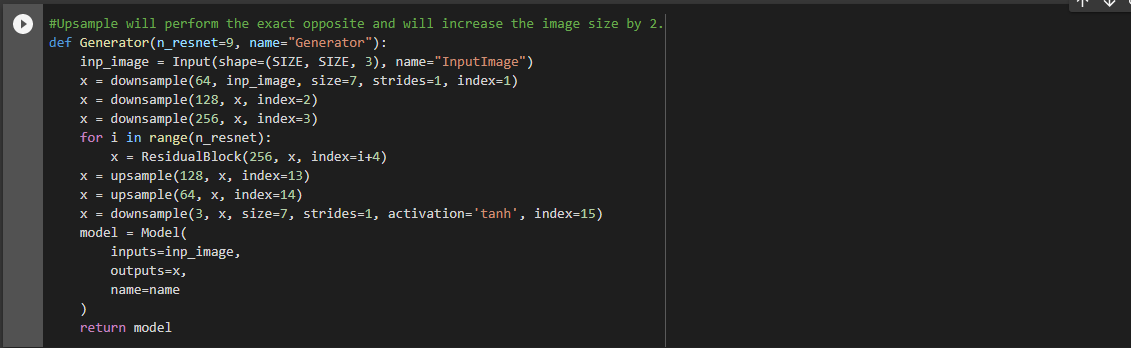
Description automatically generated



From the lines of codes above, we have successfully plotted and checked the images that we have loaded. The images plotted above shows that there are no direct relations between the appearances of horses and zebras.

The next lines of codes below will be for the generator network. Here, the Generator network is built using residual blocks. This residual block is being designed so that the network will be able to learn the features of the input image. The residual block will contain 2 set convolutional layer, instance normalization, and a skip connector.





Moving to the discriminator network, steps are exactly the same with that of the generator network. The only difference now is that instead of using batch normalization, we need to use instance normalization.

Text

Description automatically generated

The next processes combines the generator network and the discriminator network which will be essential in training the whole model.

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

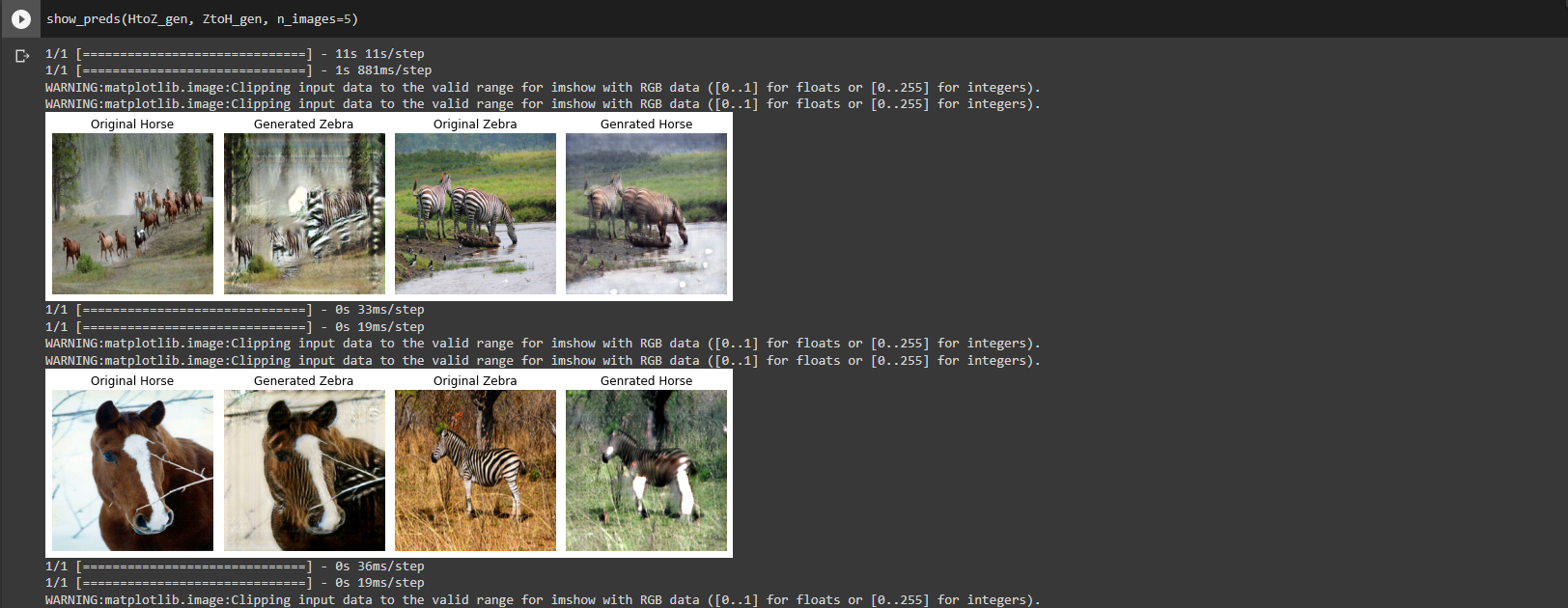
Text

Description automatically generated

Lastly, these next lines of codes will process the image translation after the training. The flow of processes will loop through the model since it is characterized by the generator and the discriminator networks being combined to make the whole final algorithm.

A screenshot of a computer

Description automatically generated with medium confidence



Timeline

Description automatically generated

Graphical user interface, text

Description automatically generated with medium confidence

Graphical user interface, website

Description automatically generated

Graphical user interface, website

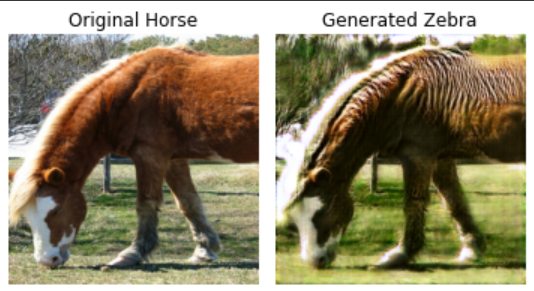
Description automatically generated

**Conclusion:**

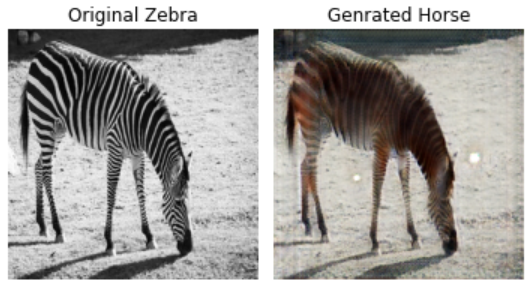
The application of the Cycle GAN in this activity helped me understand ways how wide can machine learning be applied specially in images. Through this, images can be edited to anything or any style for it to be implemented. In this activity, the model was able to make a horse look like a zebra, and a zebra to look like a horse, though the processing of the images were not that good. Still, it is a remarkable experience to observe how neural networks can process anything.

The images shown below are some of the outputs that were obtained from this activity.

**Horse into Zebra**



**Zebra into Horse**

****

**References:**

1. https://www.kaggle.com/code/utkarshsaxenadn/image-to-image-translation-cycle-gan/notebook - Code reference and dataset
2. https://stackoverflow.com/ - Errors, Buggs, and Fixes.