**Technological Institute of the Philippines**

938 Aurora Blvd., Cubao, Quezon City

**COLLEGE OF ENGINEERING AND ARCHITECTURE**

**ELECTRONICS ENGINEERING DEPARTMENT**

**COE 005 – Prediction and Machine Learning**

**ECE41S11**

**Generative Adversarial Network**

**Class Exercise**

**Using styleGAN for Image Generation**

**Submitted to:**

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

**Submitted by:**

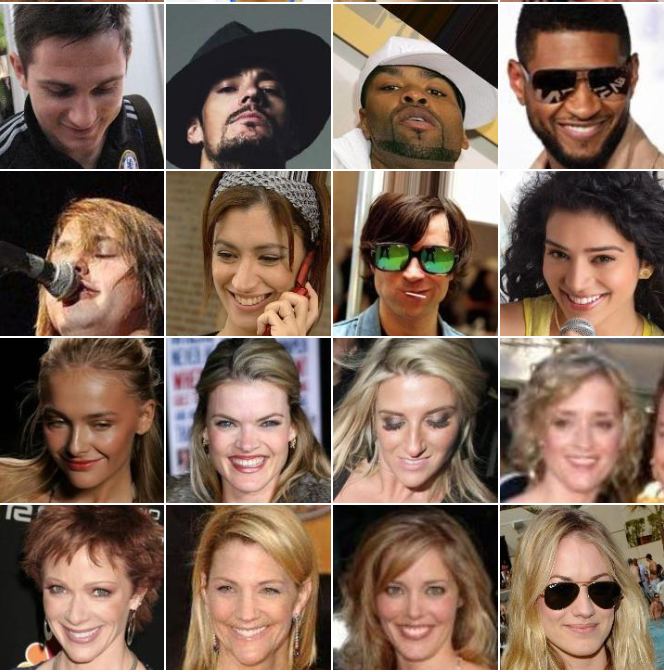
**John Andrei Mercado**

**BSECE – ECE41S11**

**Submitted on:**

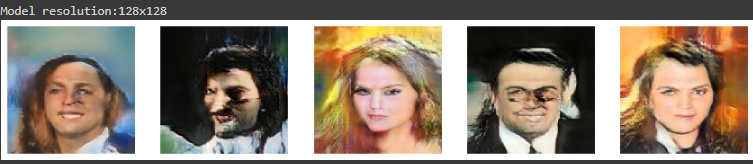
**October 20, 2022**

Dataset



Celeb\_A dataset items

Sample outputs

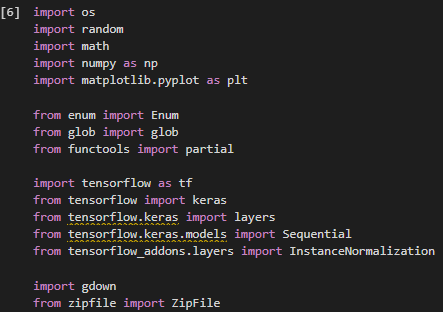




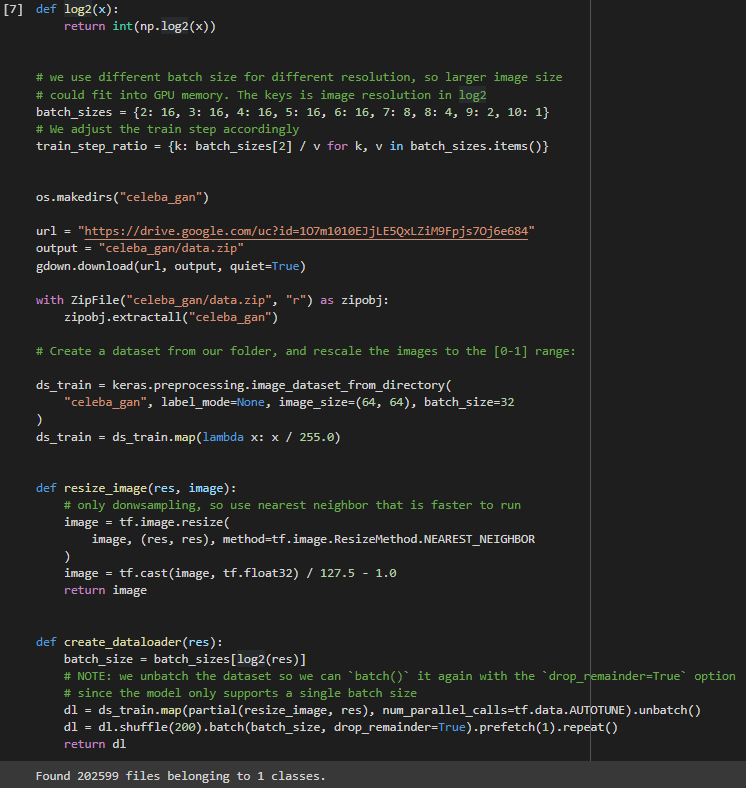
In this activity, a demonstration of face image generation using GAN was conducted. Generative Adversarial Networks is a network for generating data from scratch using 2 different NN models that compete with each other. These 2 networks are knowns the generator and discriminator. The generator is a convolutional NN that creates numerous images with the goal of fooling the discriminator. As the name suggests, the discriminator is a deconvolutional NN that is in charge of detecting if the image generated by the generator is real and fake, and whenever there’s an outcome, it is used to feed back to the generator to train itself in fooling the discriminator better. In this activity, styleGAN is used.

The following are the machine learning code used to perform styleGAN with Google Colab as the compiler.

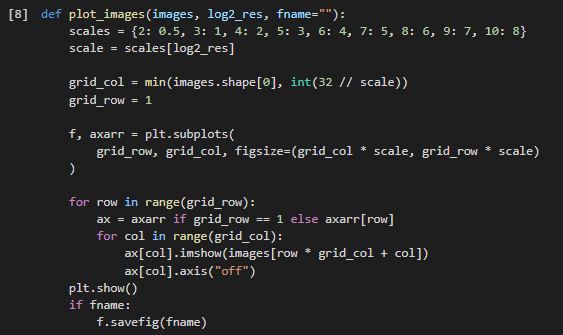
Initial part of the code is to call the general libraries and to mount Gdrive to get the directory for the dataset and to perform the different functions later on related to DCGAN.



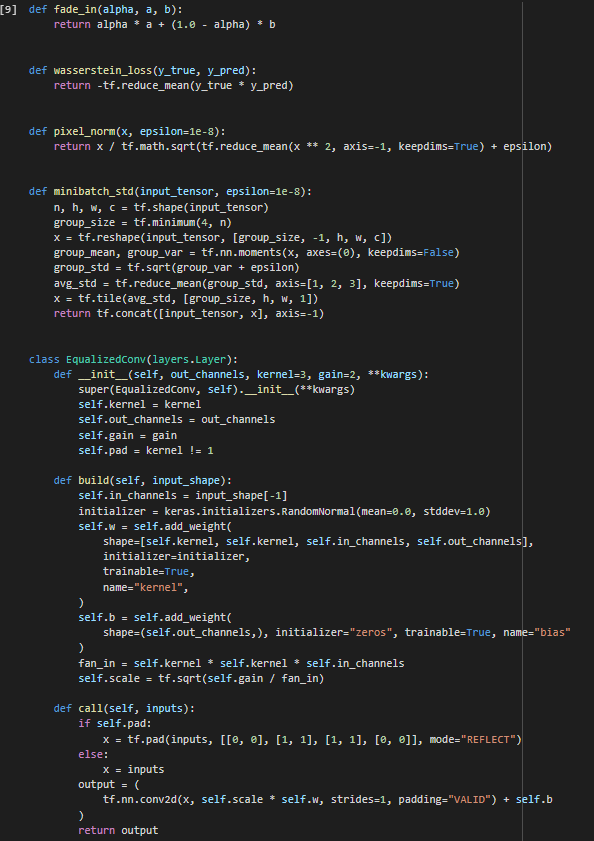
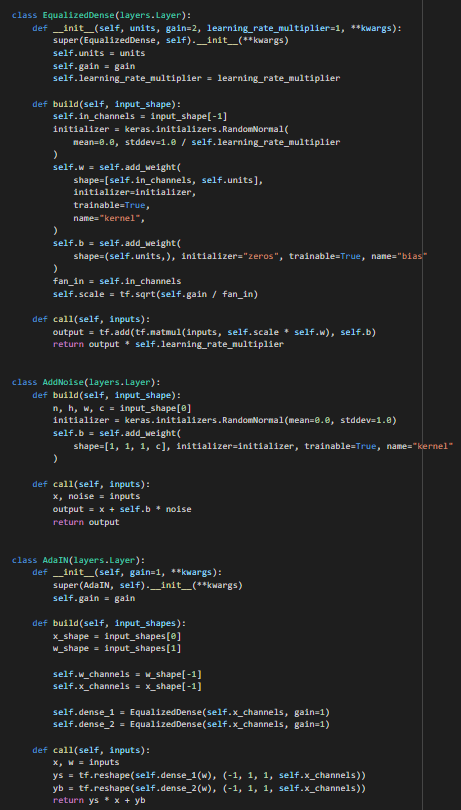
Next, the source images were also imported. The images were resized to 64x64. As seen, the dataset contains 202599 files of different celebrity face images and angles. A function for assigning a batch size depending on resolution is applied, this influences the training step later on.



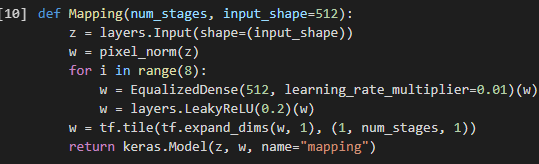
A function for displaying the images after every epoch is defined.



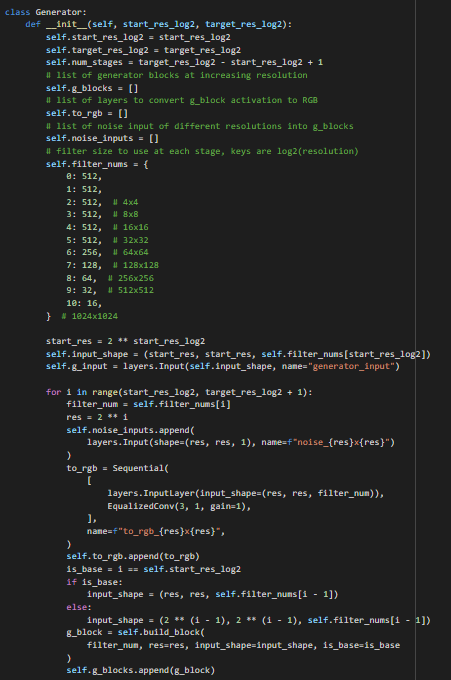
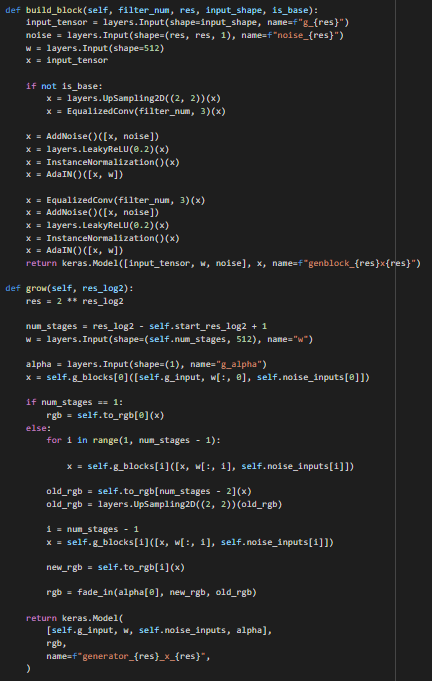
Functions were defined for the different parameters used for building the generator and discriminator.

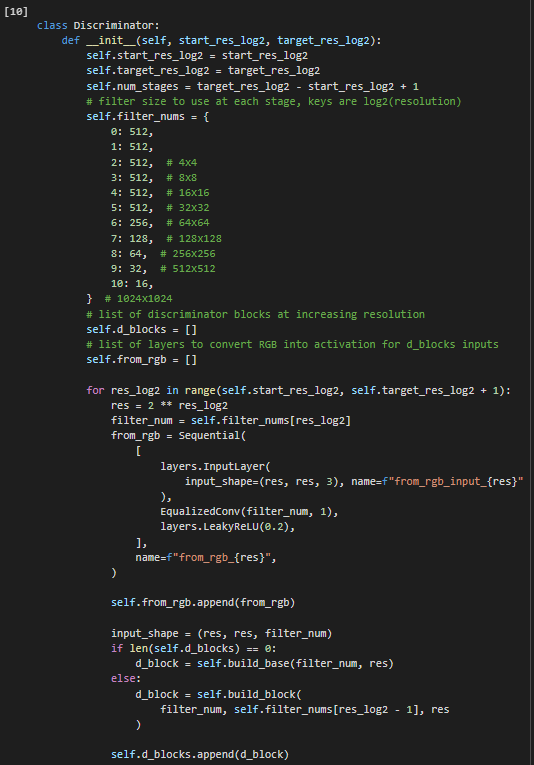


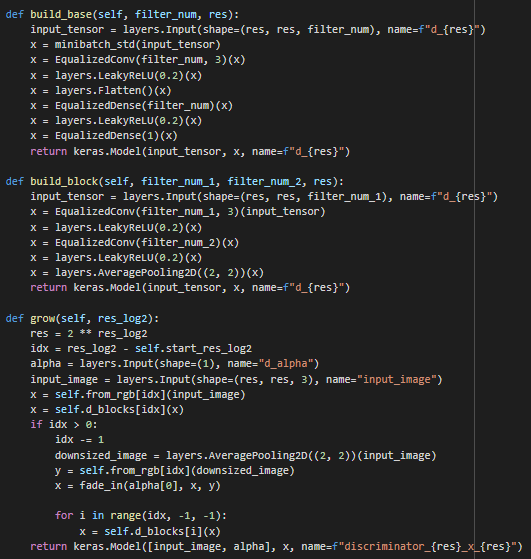
A function for mapping the random noise into the style code is defined.



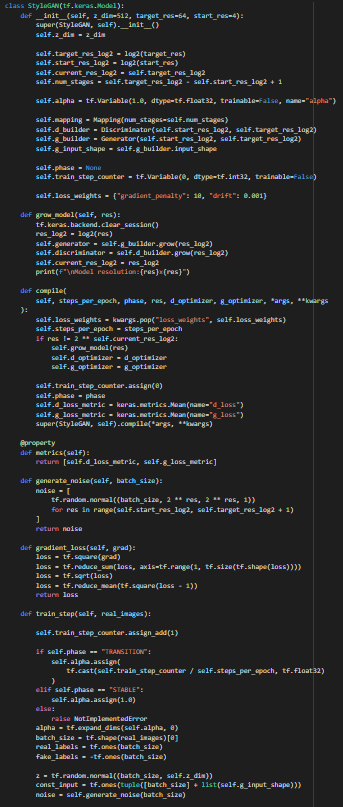
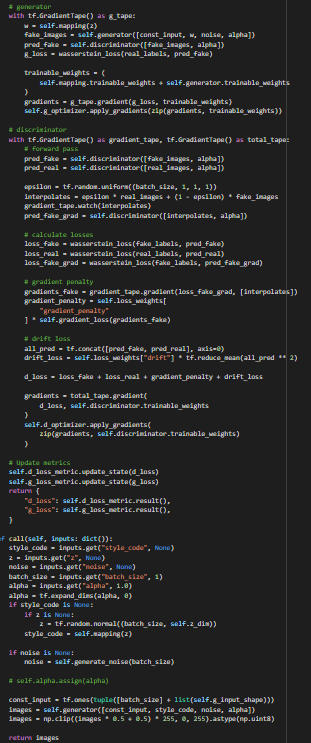
Next, the generator network is defined. This will act as the creator of the real/image which will undergo classifying by the discriminator



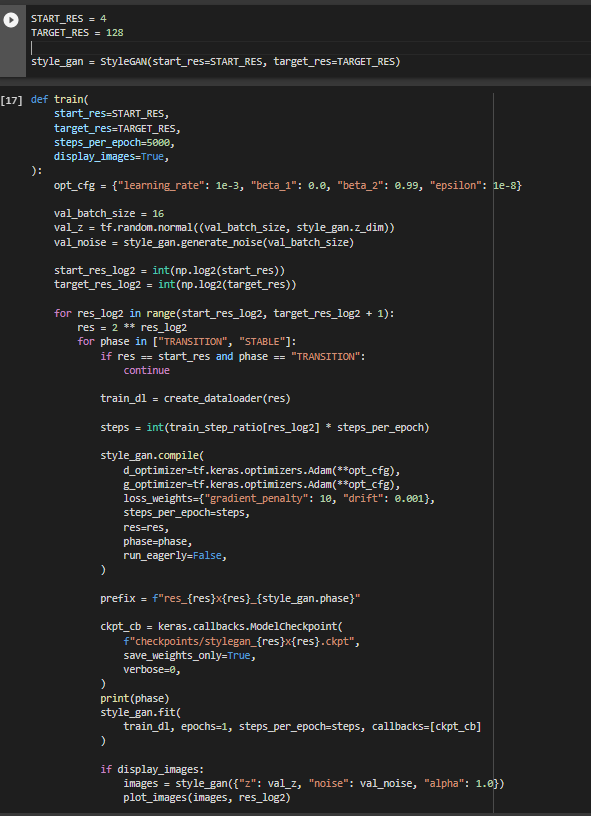
The Discriminator model was also defined incorporating the parameters defined earlier.



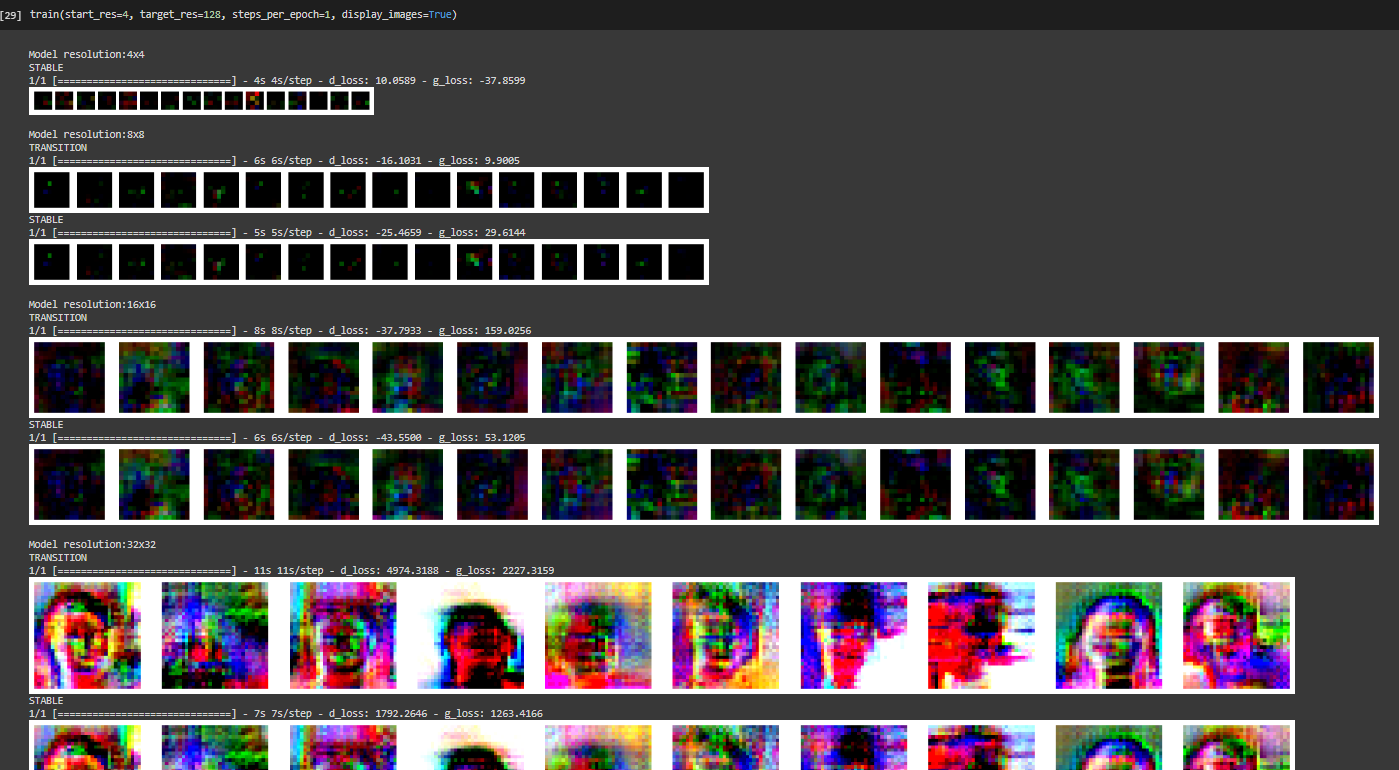
The StyleGAN model with custom train\_step metrics. Metrics to be used were also defined to gauge the progress of the model throughout the training epochs.

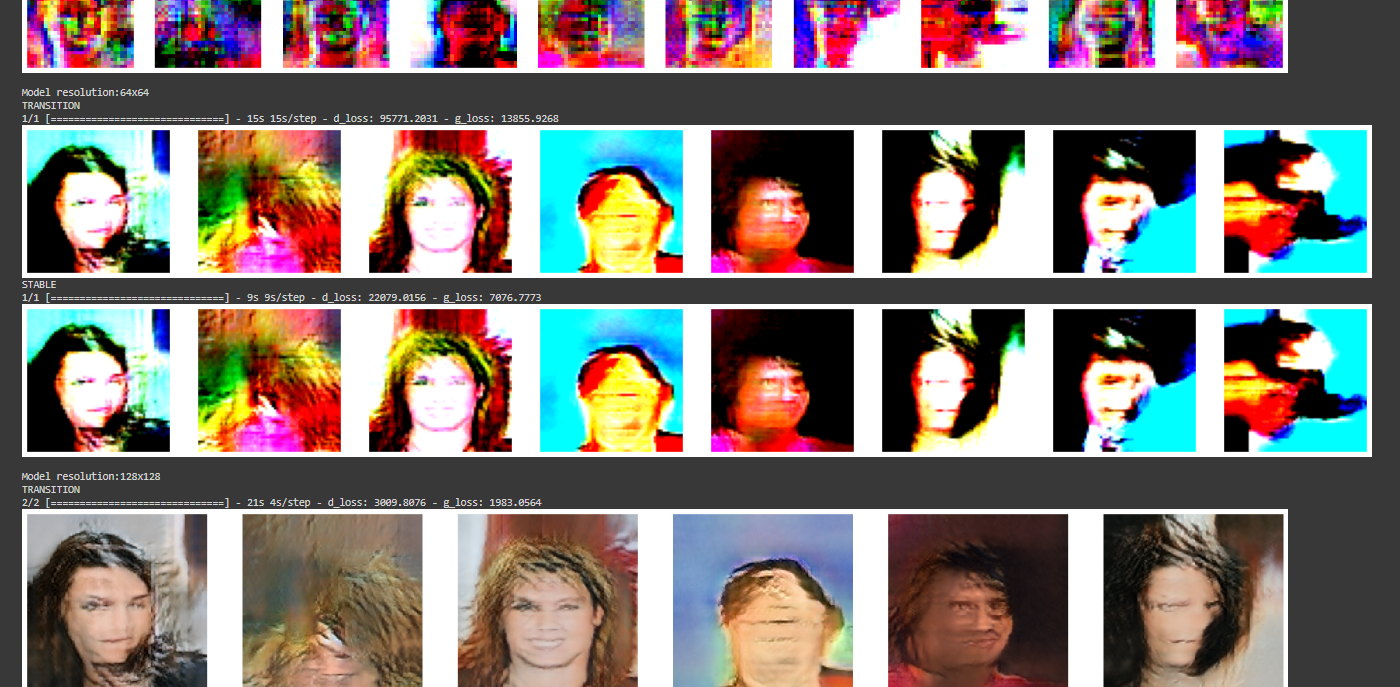
 

The section for training the model is defined. The model is first build at the smallest resolution then progressively grow the model to a higher resolution by combining the generator and discriminator blocks. The training happens in two phases, “transition” and “stable”. In the transition is where the previous features are incorporated with the current resolution which allows a smoother transition when scaling the resolution up. In fitting the model, each epoch is used as a phase.

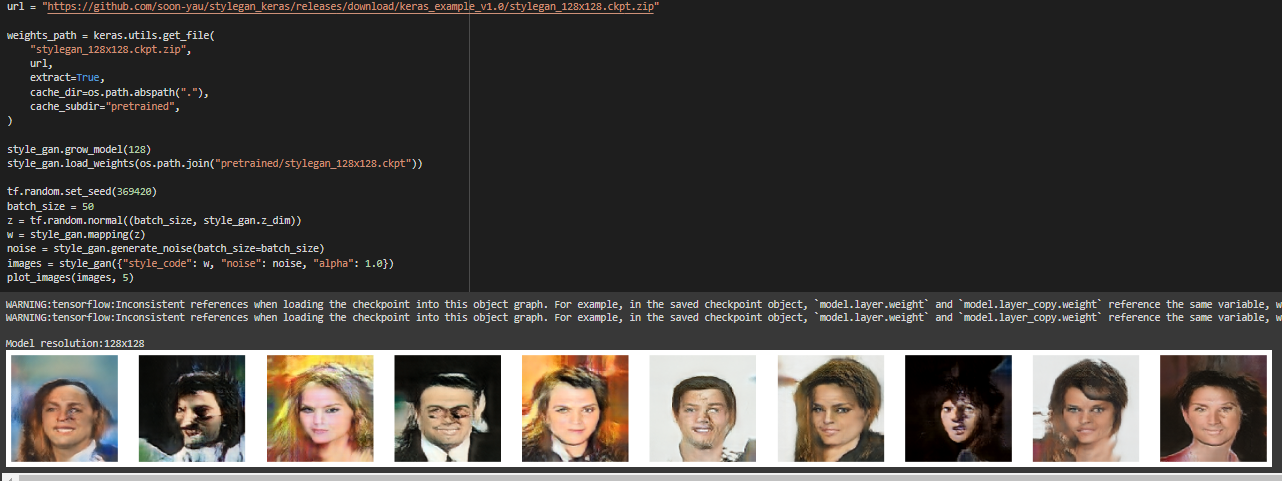


This is the sample progression every epoch, where 2 phases occur in each. This is just the initial sample, to improve the results would require a higher epoch.





Given the time constraints of training the images, results can be shown using pretrained 64x64 checkpoints.



Images produced can also be mixed to create a new image containing the style of one of the input images.



In conclusion, StyleGAN utilizes an alternative generator network which it borrows from style transfer literature, in particular, the use of instance normalization. It proposes changes to the generator model, incorporating mapping networks to map points throughout the latent space. It also uses a form of regularization called mixing regularization which mixes the 2 style variables during training. This exercise aims to generate computerized celebrity face images from scratch using the celeb\_a dataset as the input dataset. Images can also be mixed to create new images containing the style and features of the 2 inputs.

References:

https://keras.io/examples/generative/stylegan/