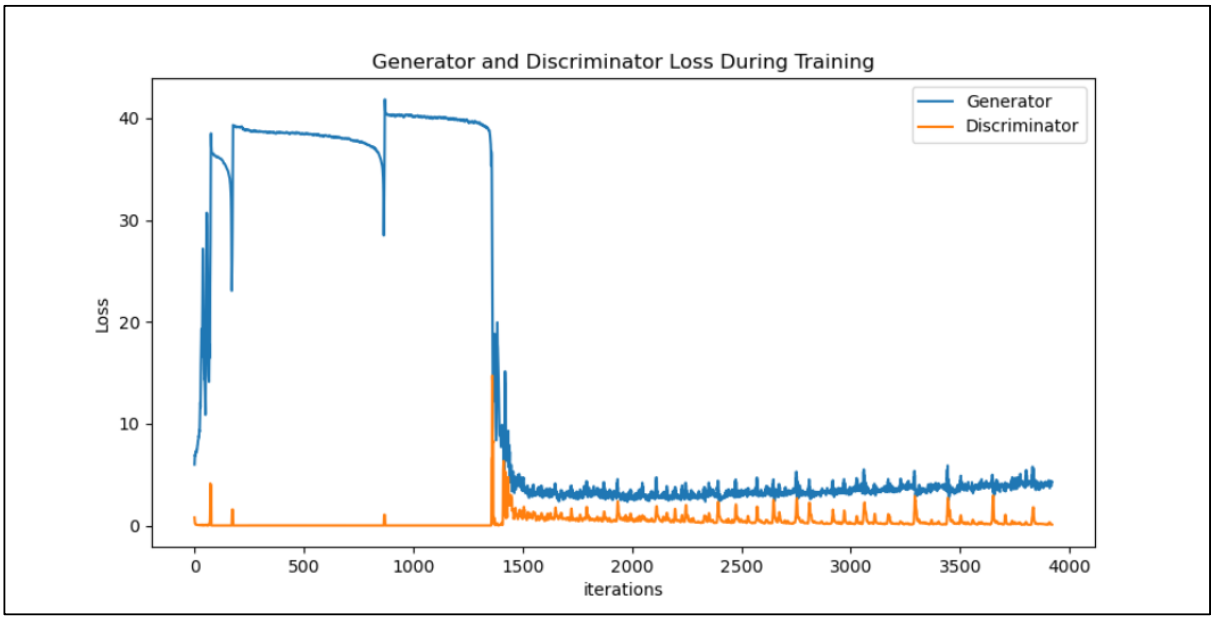
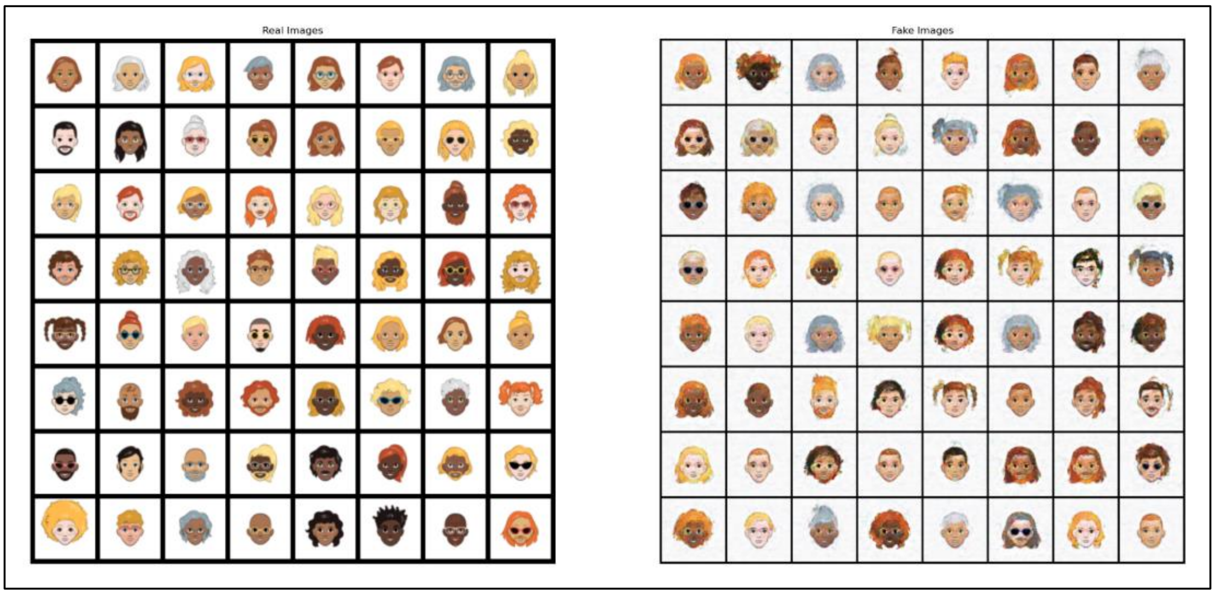
MLDS GENERATIVE AI – Assignment 5

Loss Profiles:



Generated Images:



Model Summary:

The generator architecture is designed to transform a noise vector into a synthetic image. It uses a sequence of convolutional transpose layers to upscale the input, increasing spatial dimensions while reducing channel depth. Batch normalization is applied after each convolutional transpose layer to stabilize training and improve gradient flow. ReLU activation functions are used to introduce non-linearity, allowing the network to learn complex representations. The final output is passed through a Tanh activation function to ensure pixel values are within the range [-1, 1].

The discriminator architecture is structured to evaluate the realism of input images. It employs a series of convolutional layers to downsample the input while increasing channel depth, effectively extracting features. LeakyReLU activation functions are used to allow a small gradient when the input is negative, helping prevent dying neurons. Batch normalization is applied after most convolutional layers to stabilize training. The final layer outputs a probability that the input image is real, using a sigmoid activation function. This architecture helps the discriminator distinguish between real and fake images effectively.

Config parameters:

A batch size of 64 is used as it provides a good balance between memory usage and gradient stability. Image of size 64x64 allows for efficient training while still capturing meaningful features of the images. Num channels is set to 3 for RGB images. Random_dim 100 is the dimension of the noise vector used as input to the generator. This allows the generator to produce diverse images without becoming too difficult to train. gen_dim 64 and disc_dim 64 control the number of filters in the first layers of the generator and discriminator. Increasing the number of epochs from the initial 5 to 25 allows the model more time to converge and learn from the data. Learning rate 0.0002 is relatively low. A low learning rate helps prevent overshooting and allows for more stable convergence, especially when combined with Adam optimizer's default settings. Setting beta1 to 0.5 can help by giving more weight to recent gradients.