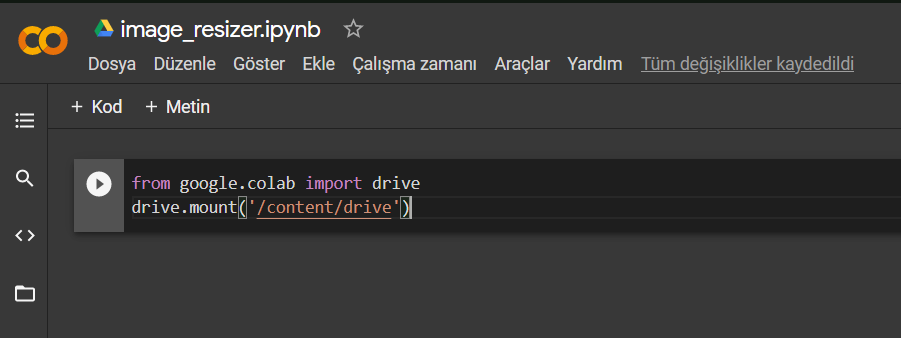
We downloaded our dataset from kaggle. This dataset is called WikiArt dataset[10] and it is for generating art in multiple styles and genres. Our dataset will only use landscape paintings data. For that we only downloaded landscape folder and we will only use that. Our dataset consists 15000 images. First we will write a small python script to select all the images from the folder and resize them to 64x64 and save them in a path landscapeResized/resized so the ‘resized’ folder will contain 15000 64x64 landscape painting images.

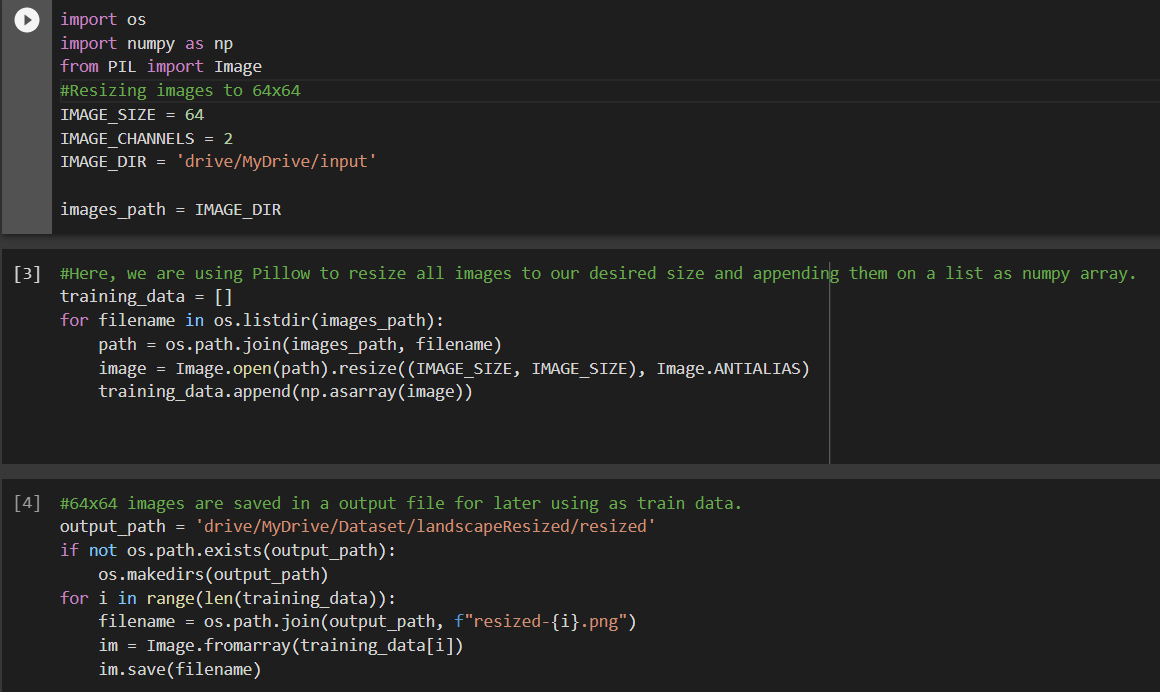
First of all we need to download our dataset and upload it to Google drive. We will create a folder called “input” and we will put the images that we uploaded to the input file.

Let’s open up Google colab website and create a new notebook.

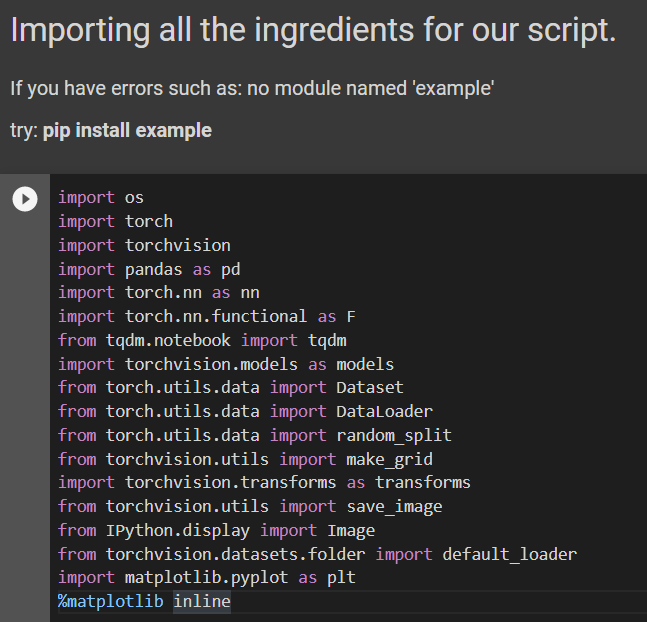
Now let’s synchronize our notebook with our Google Drive. Go ahead and run the following code then to the authorization.



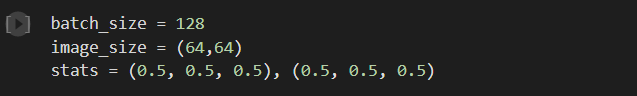
We are choosing our input file to get the images then we resize them by using Pillow. At the end we save them in the path landscapeResized/resized for later use in training.



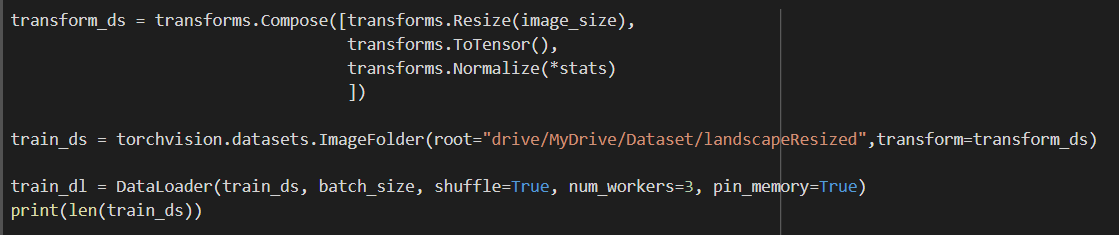
Now we create a new notebook and call it art\_gan.ipynb. First we will import the necessary libraries.



We need to define some things before going ahead. We are doing transformation on our images before it’s begin devoured by our model. Here we are using an image size of 64 x 64. And batch size means: number of images processed in an iteration. Our batch size equals to 128.

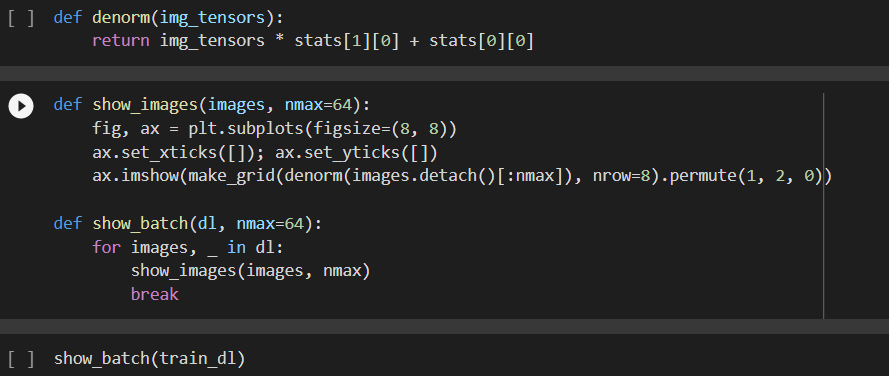


Now we will load our train data, print the size and display some of them.

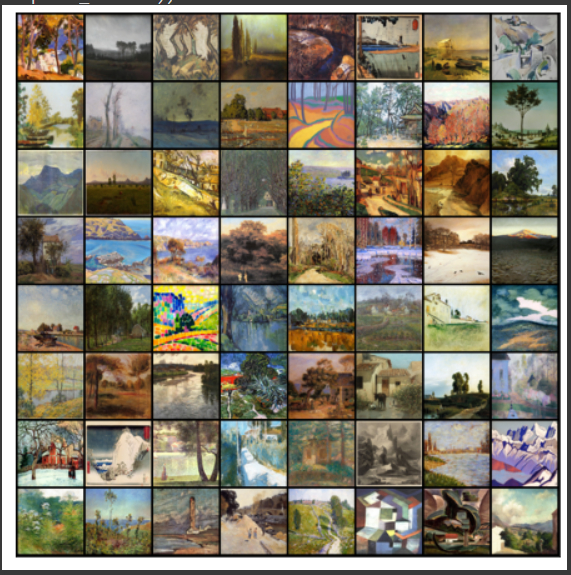


We will get output: 15000.

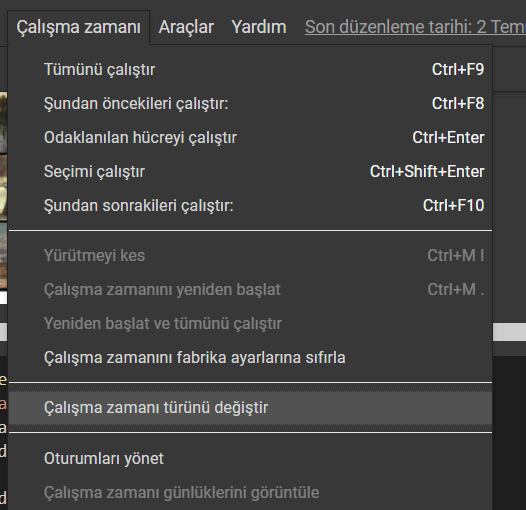
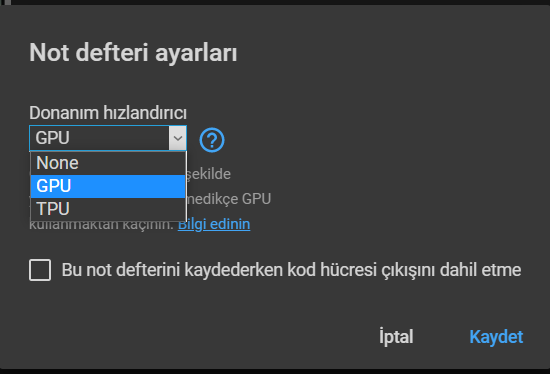
Our functions to display images.



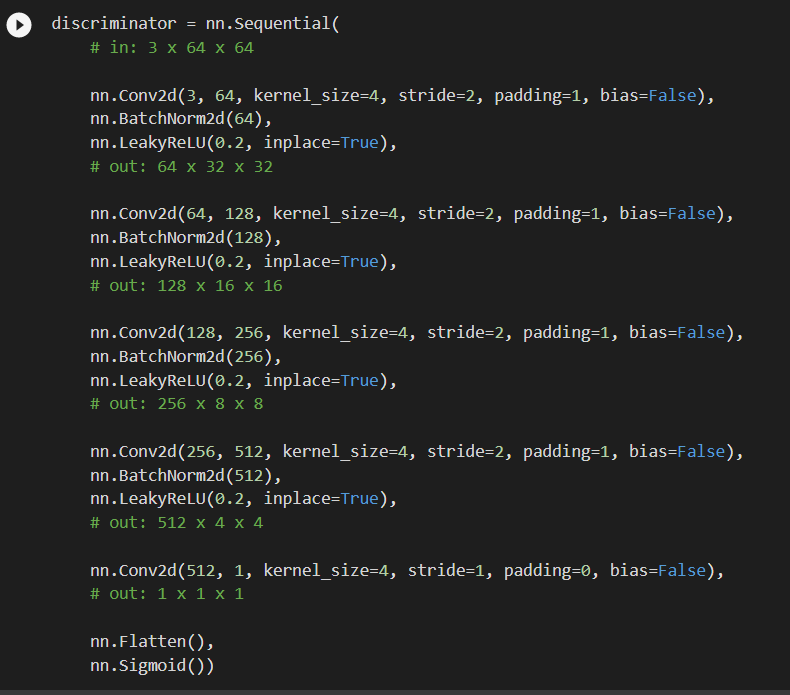
Output:



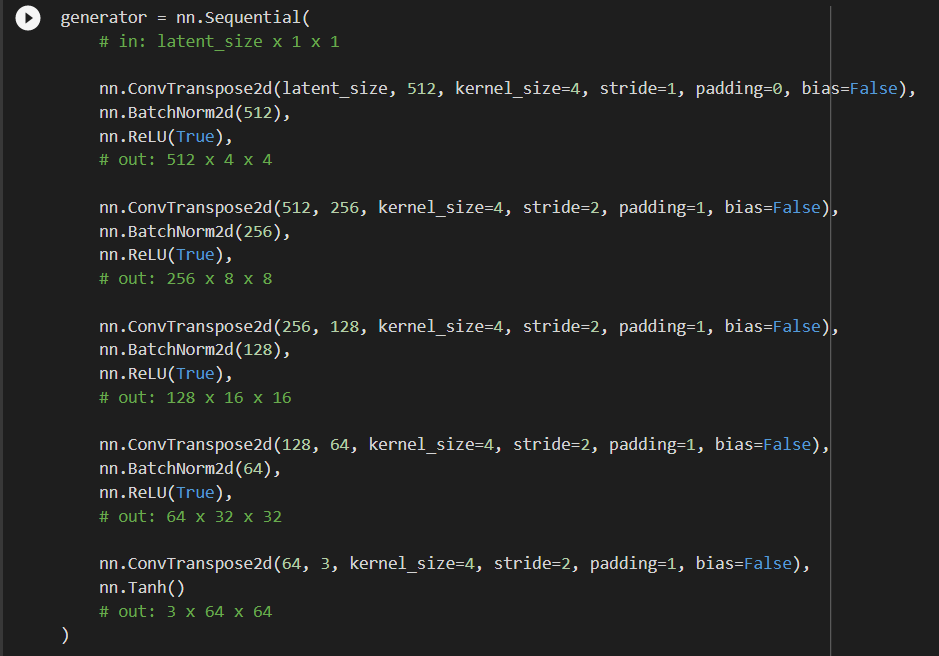
We will connect to the GPU.

The discriminator takes two sets of input; real images and fake images. Its job is to classify them properly whether the given image is fake or real. Since we are dealing with images we will use Convolution Neural Network(CNNs) for our discriminator’s neural architecture.

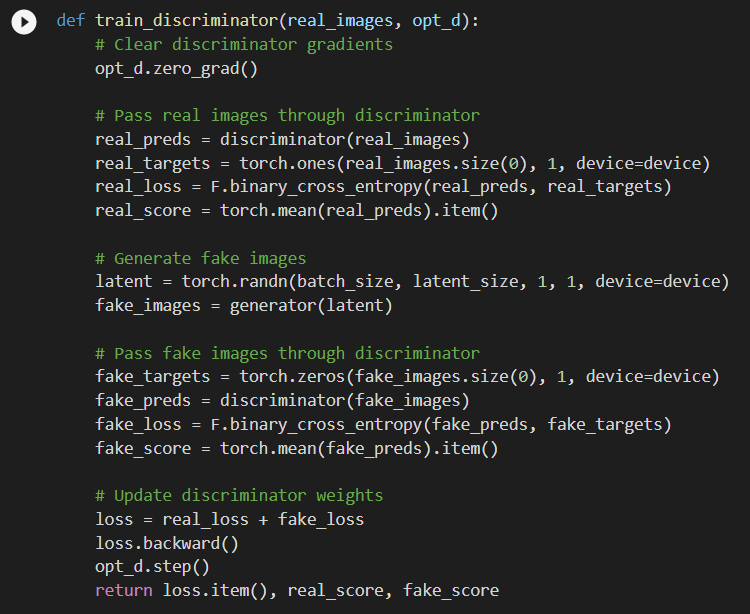


The input to the generator is typically a vector or a matrix of random numbers (referred to as a latent tensor) which is used as a seed for generating an image. The generator will convert a latent tensor of shape (128, 1, 1) into an image tensor of shape 3 x 28 x 28. To achieve this, we'll use the ConvTranspose2d layer from PyTorch, which is performed to as a transposed convolution (also referred to as a deconvolution)



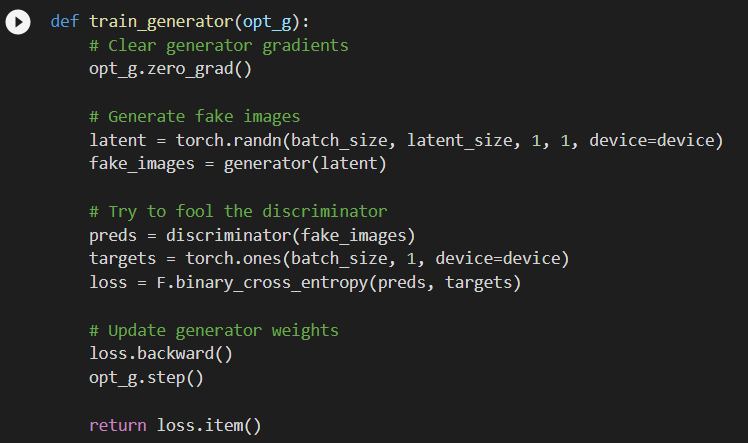
**Discriminator Training**

Here the discriminator is a binary classification model, so we can use the binary cross entropy loss function to quantify how well it is able to differentiate between real and generated images.

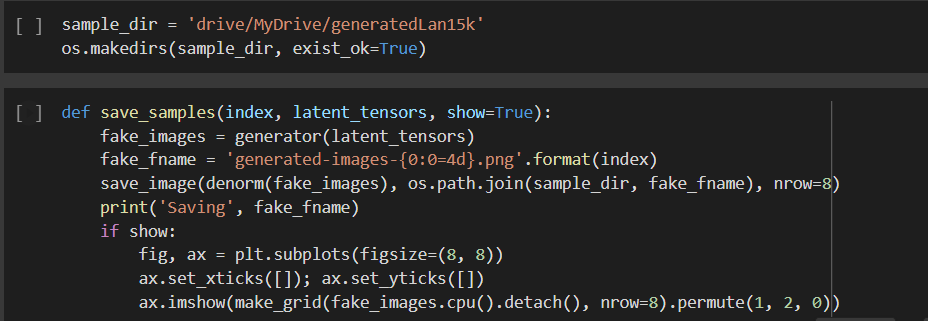


**Generator Training**

The generator’s job is to keep generating real like artwork images so as to fool the discriminator. Now here’s come the interesting part, we will be using discriminator as a loss function with which we can generate more real like images.



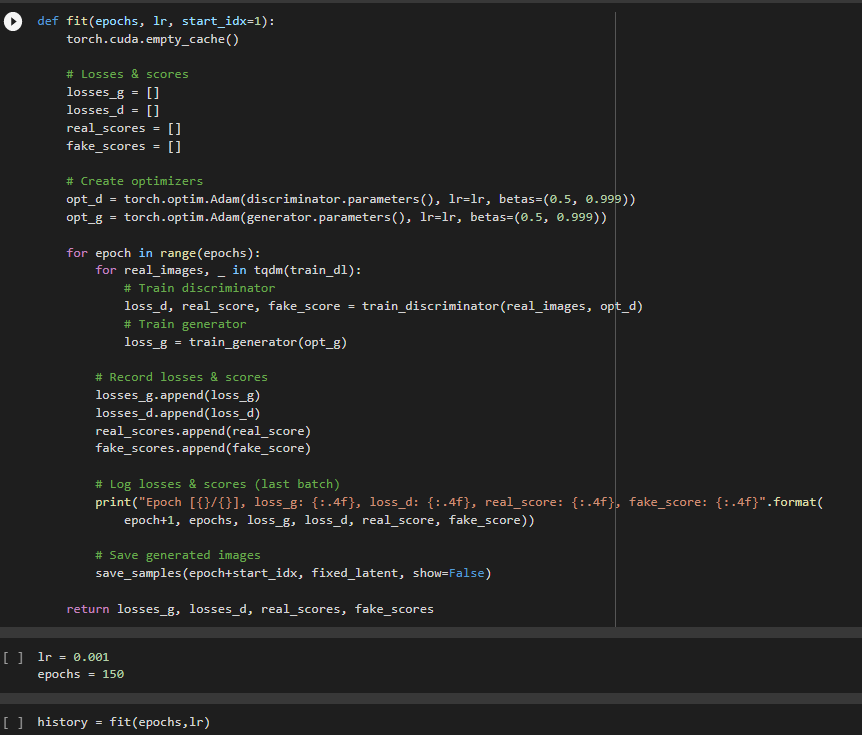
Let’s create a directory where we can save intermediate outputs from the generator to visually inspect the progress of the model. We’ll also create a helper function to export the generated images.



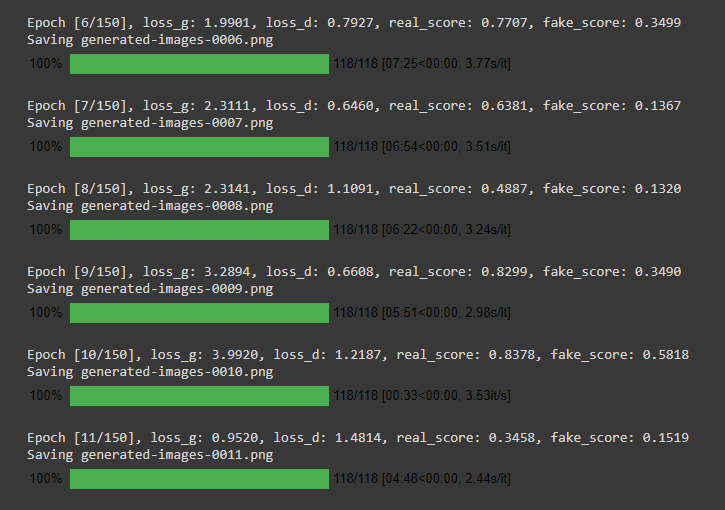
**Full Training Loop**

Now we will define a fit function wherein we will be training both our models in tandem. We'll use Adam optimizer with some custom parameters (betas) that are known to work well for GANs. We will also save some sample generated images at regular intervals for inspection.

Learning rate will be 0.001 and we will have 150 epochs.

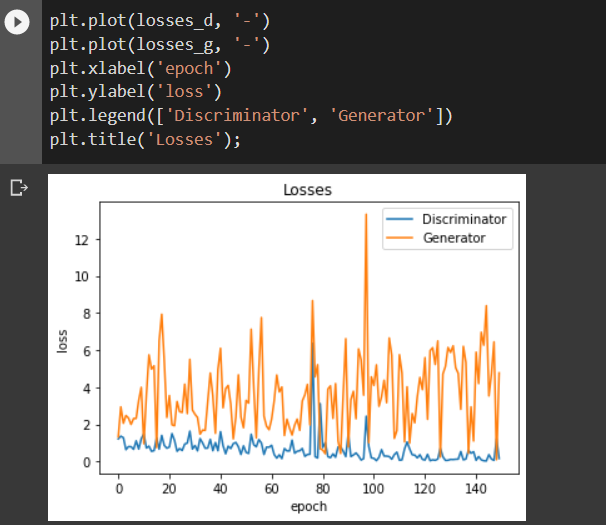


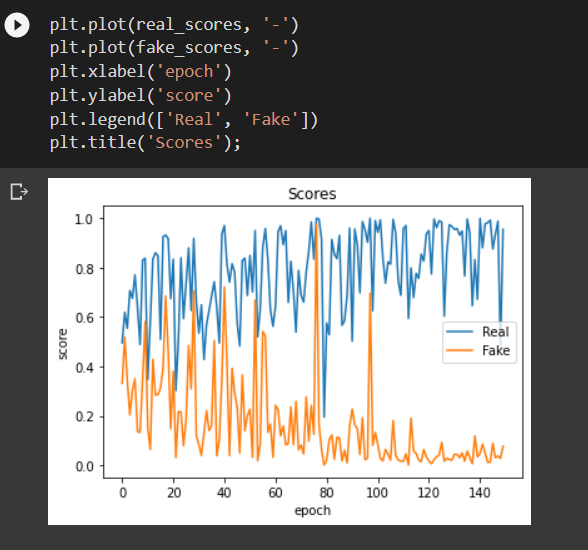
Output:



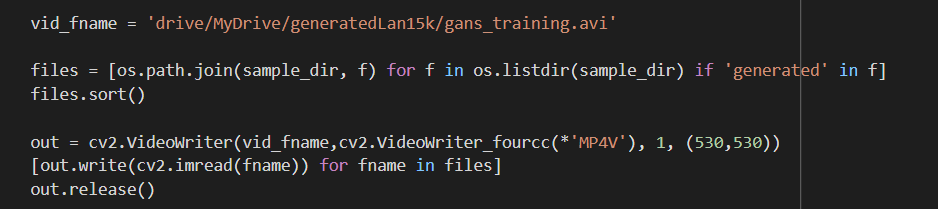
When all the epochs completed our generated images will be saved in the output path.

We will plot our metrics through iterations.





In the end we will create a gif to show models improvement through epochs and save it.



Generated images in epoch 150:

