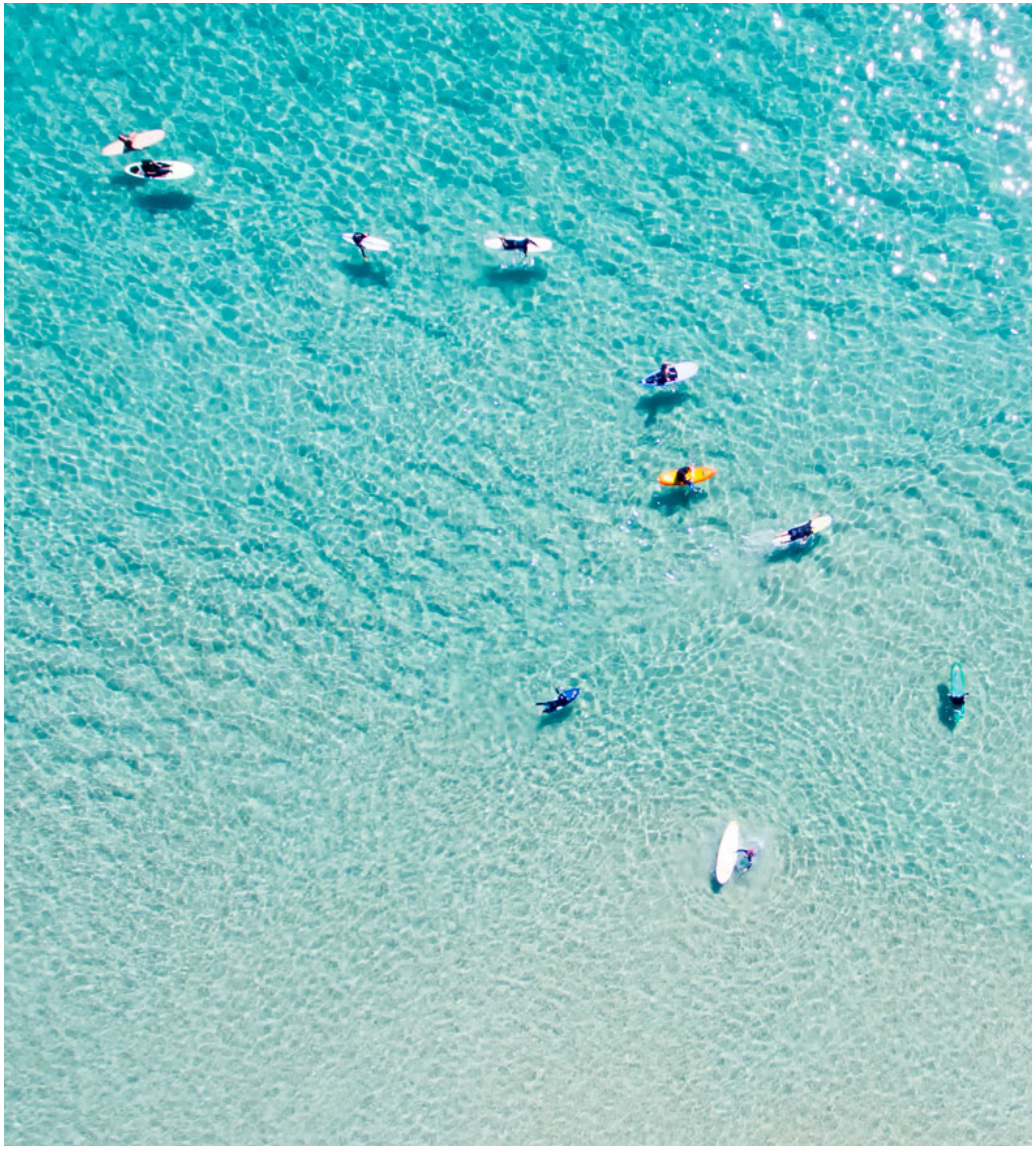
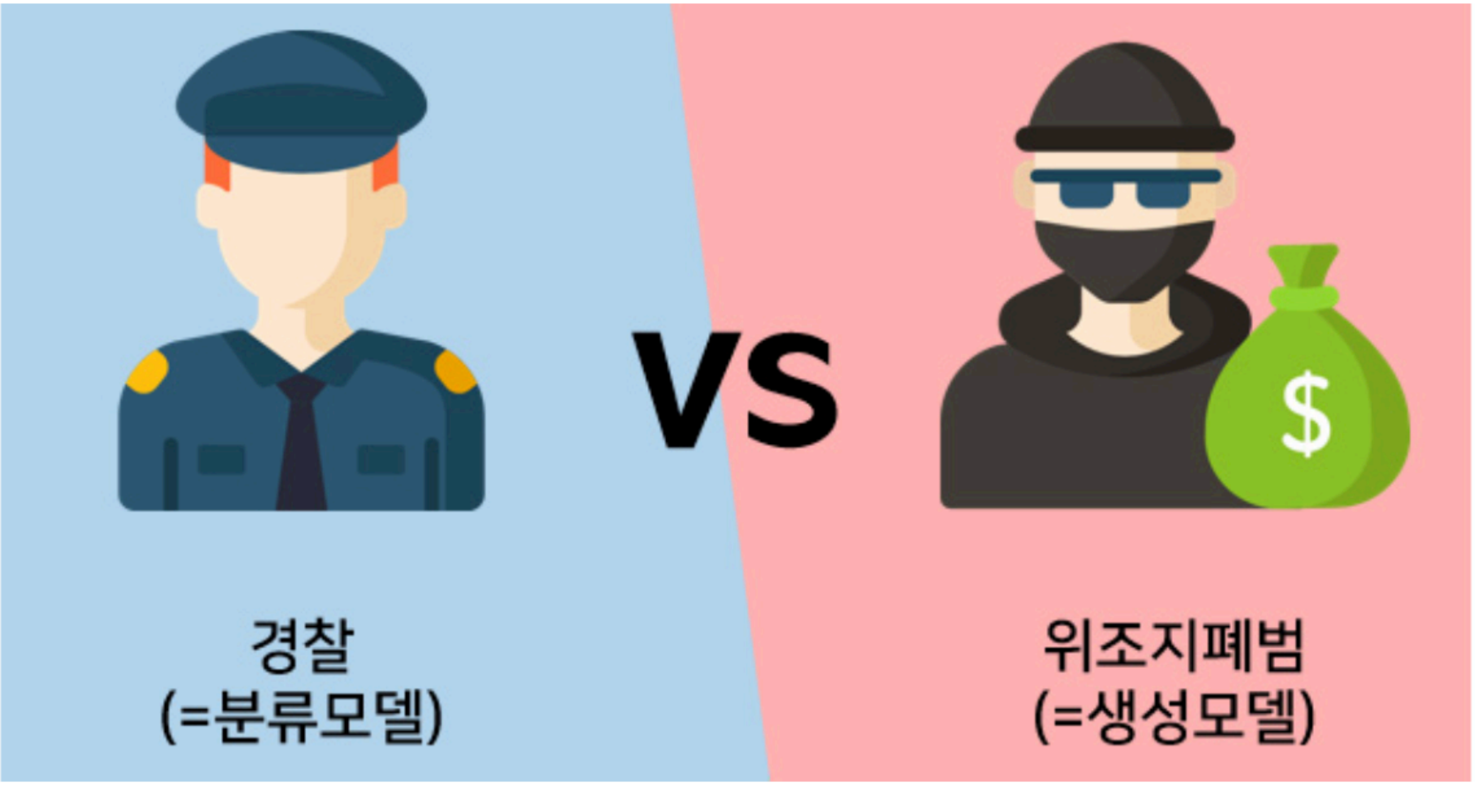
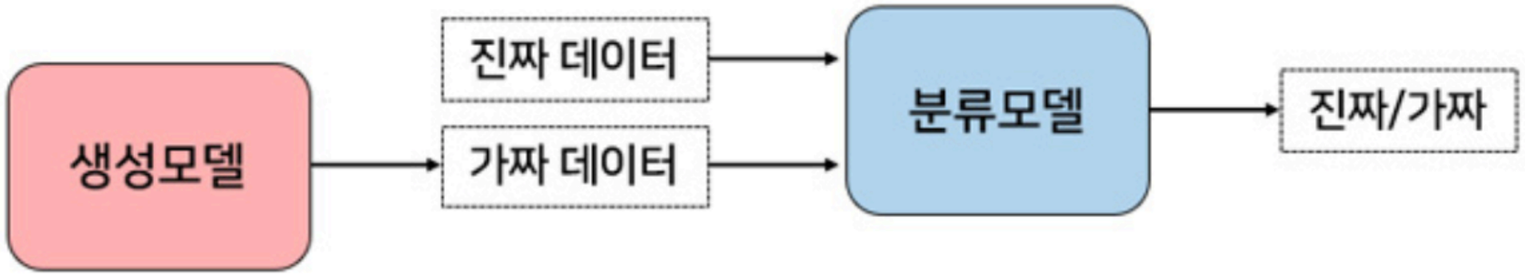
우승진

DCGAN

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS





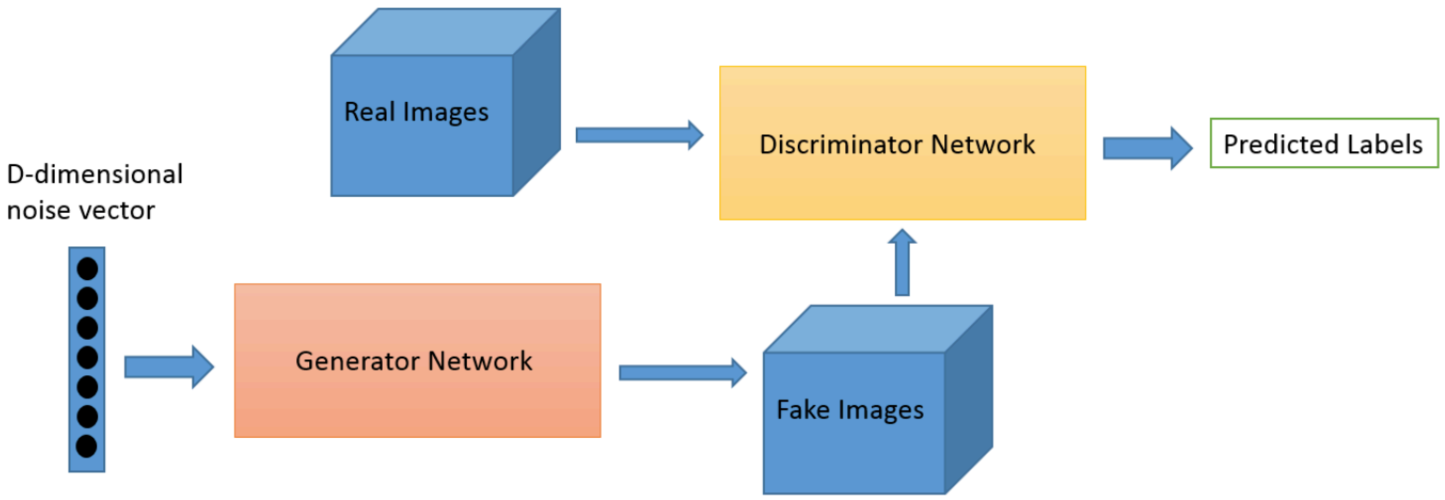

생성모델

진짜 데이터

가짜 데이터

분류모델

진짜/가짜



D-dimensional noise vector

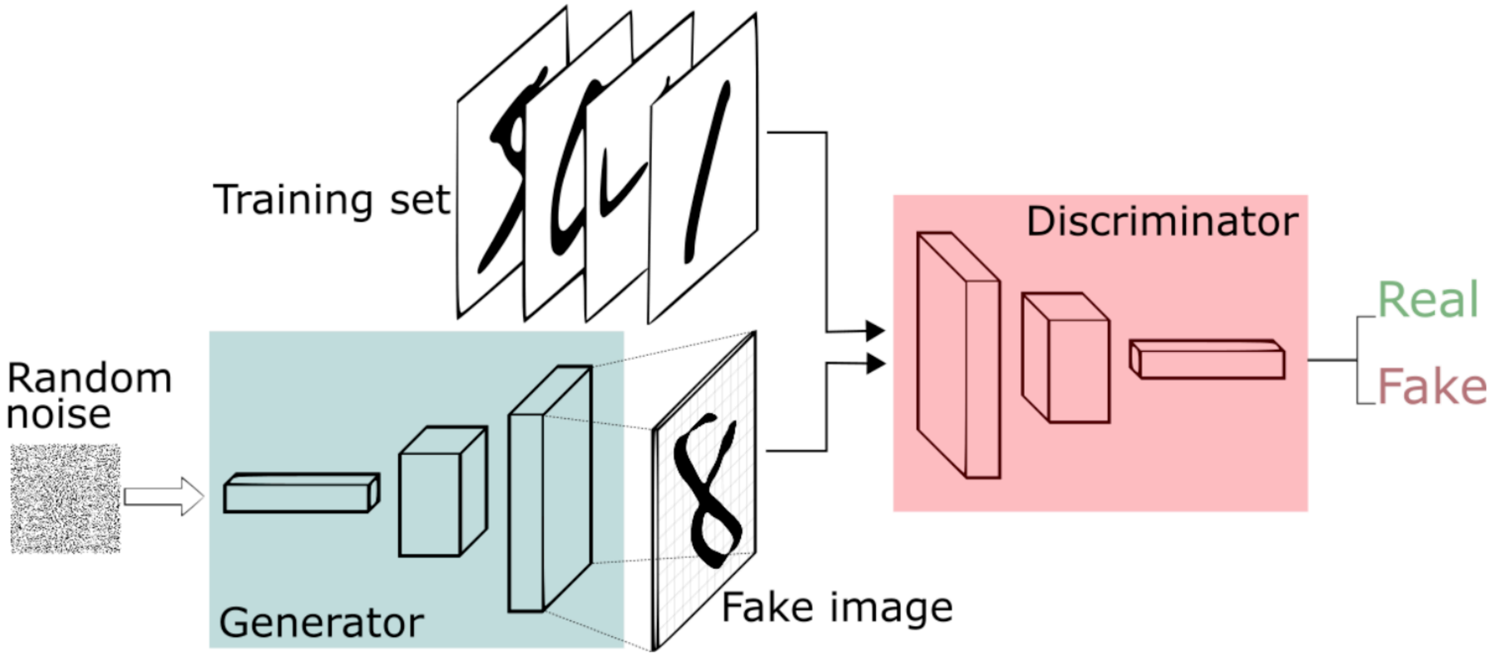
Generator Network

Real Images

Fake Images

Discriminator Network

Predicted Labels



Training set

Random noise

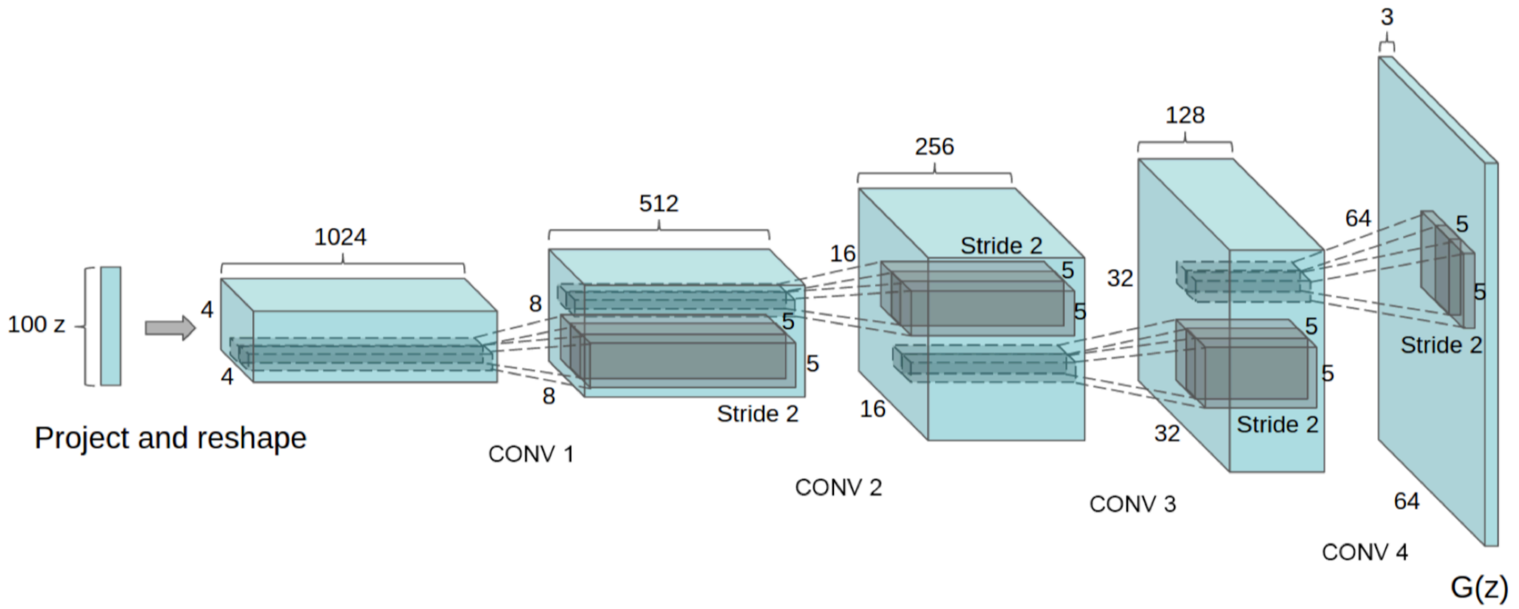
Generator

Fake image

Discriminator

Real

Fake



Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

No pre-processing was applied to training images besides scaling to the range of the tanh activation function [-1, 1].

All models were trained with mini-batch stochastic gradient descent (SGD) with a mini-batch size of 128.

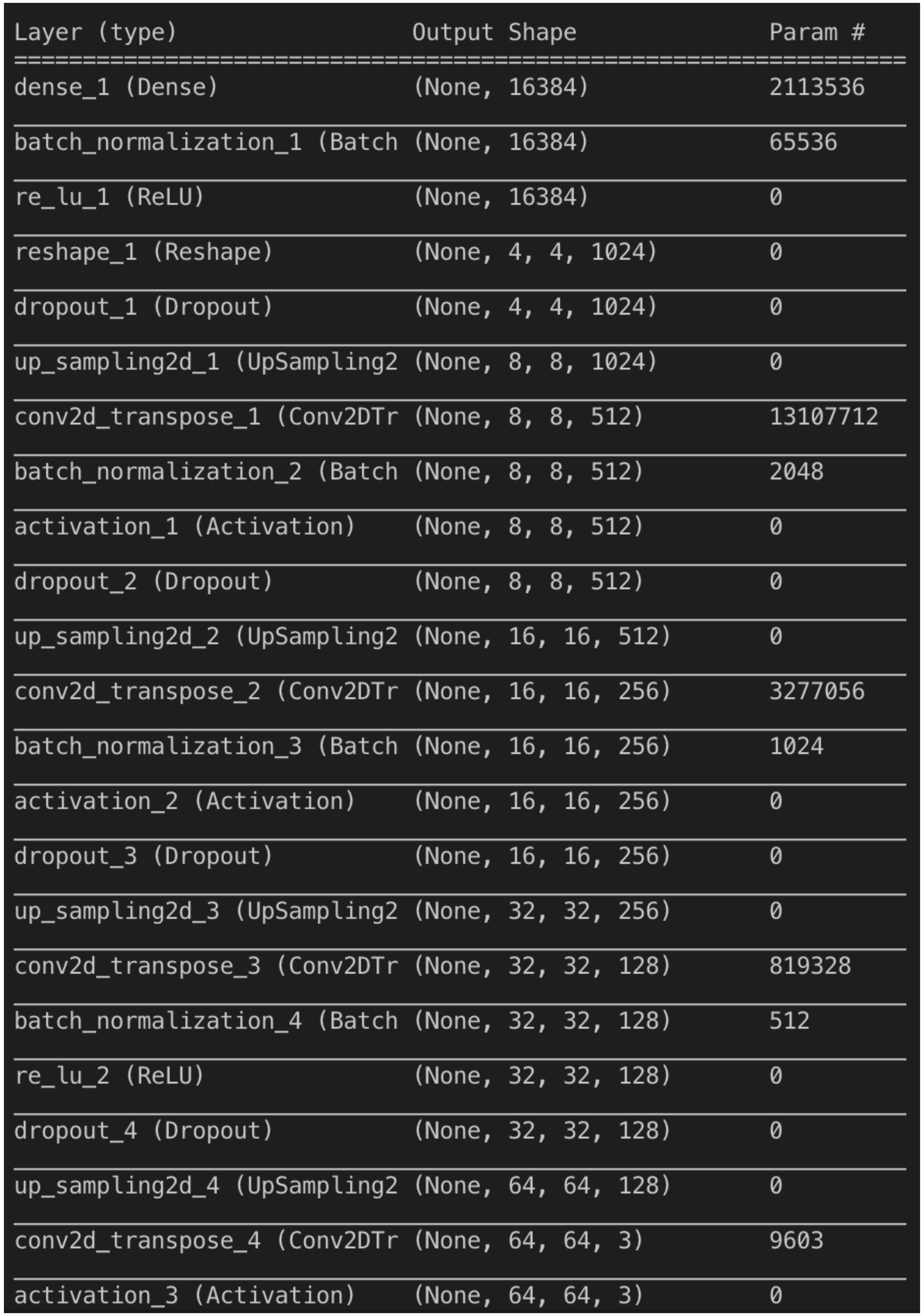
All weights were initialized from a zero-centered Normal distribution with standard deviation 0.02.

In the LeakyReLU, the slope of the leak was set to 0.2 in all models.

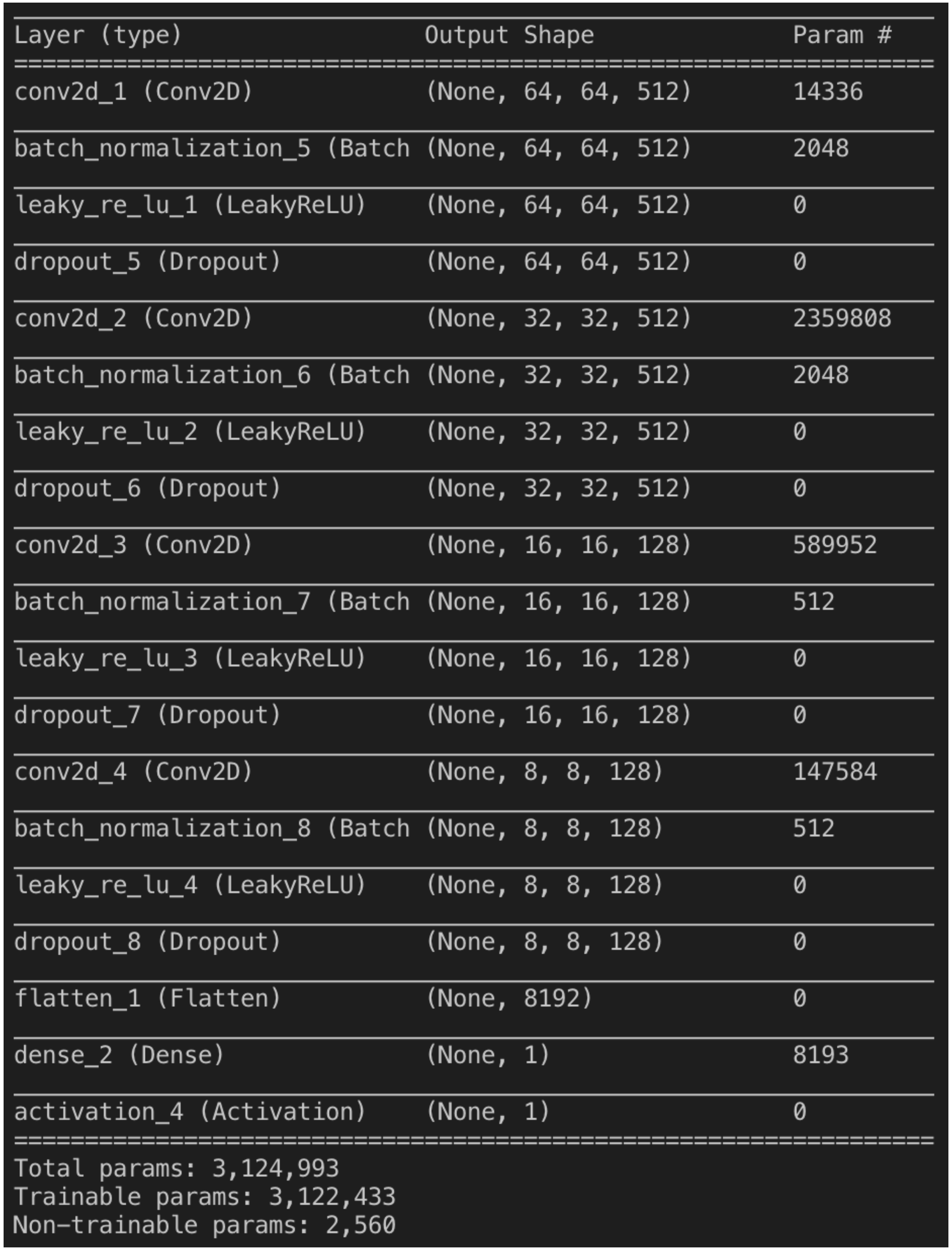
While previous GAN work has used momentum to accelerate training, we used the Adam optimizer (Kingma & Ba, 2014) with tuned hyperparameters.

We found the suggested learning rate of 0.001, to be too high, using 0.0002 instead.

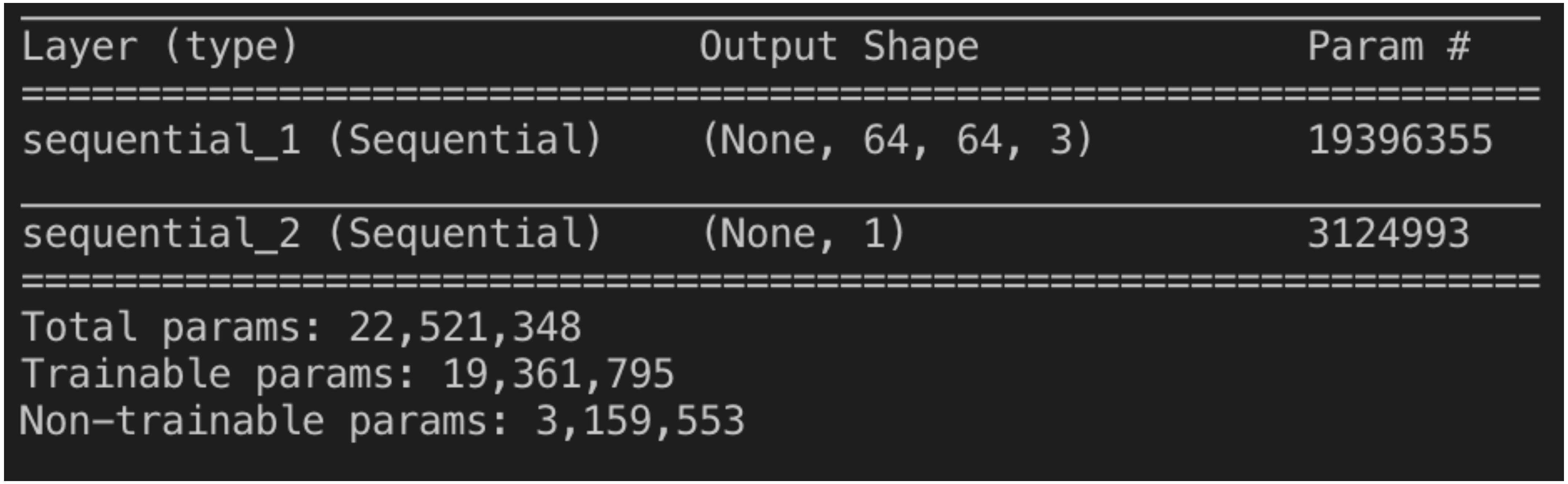
Additionally, we found leaving the momentum term $\beta1$ at the suggested value of 0.9 resulted in training oscillation and instability while reducing it to 0.5 helped stabilize training.



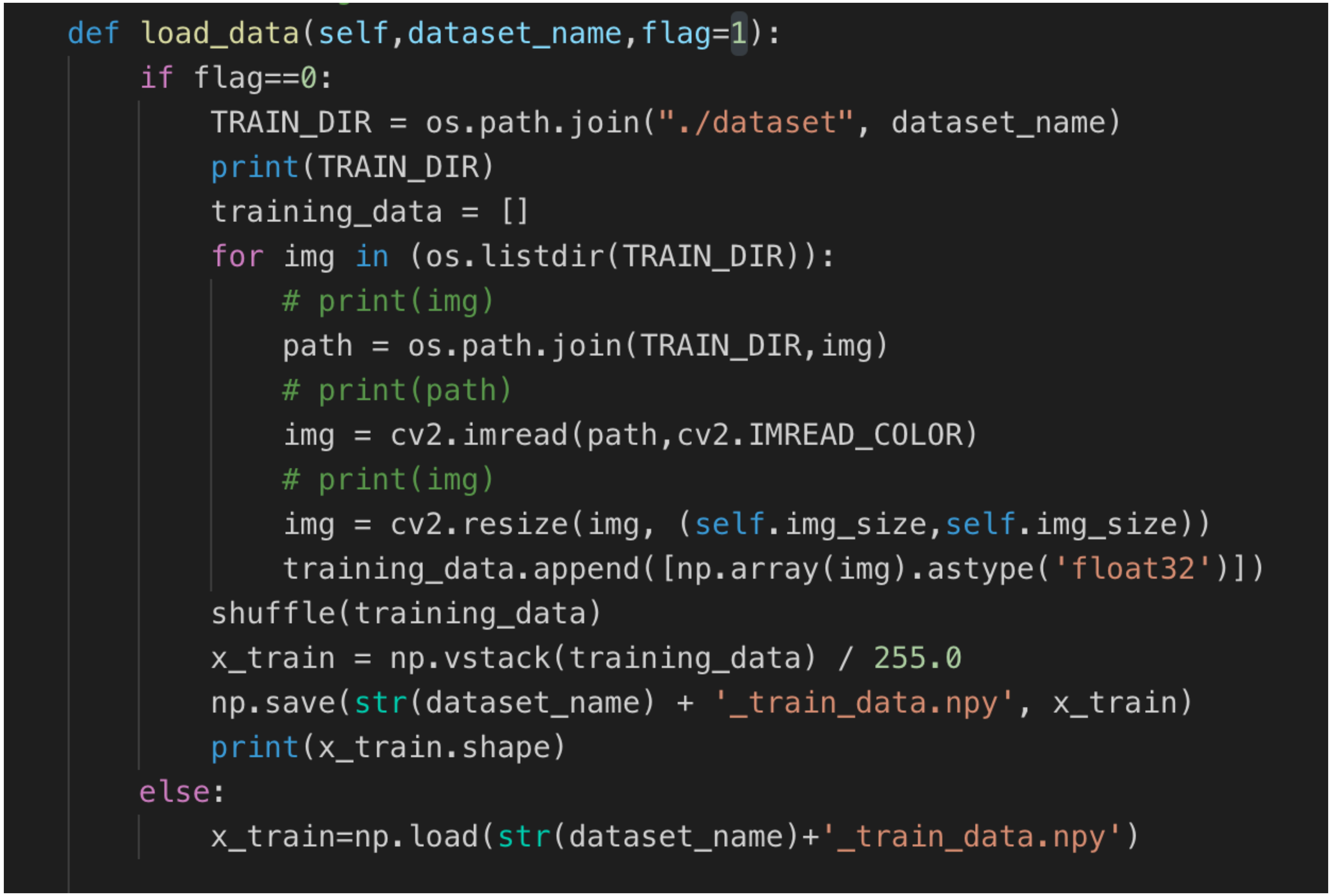
```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 16384)             2113536
_________________________________________________________________
batch_normalization_1 (Batch (None, 16384)             65536
_________________________________________________________________
re_lu_1 (ReLU)               (None, 16384)             0
_________________________________________________________________
reshape_1 (Reshape)          (None, 4, 4, 1024)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 1024)        0
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 8, 8, 1024)        0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 8, 8, 512)         13107712
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 512)         2048
_________________________________________________________________
activation_1 (Activation)    (None, 8, 8, 512)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 8, 8, 512)         0
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 16, 16, 512)       0
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 16, 16, 256)       3277056
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 256)       1024
_________________________________________________________________
activation_2 (Activation)    (None, 16, 16, 256)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 16, 16, 256)       0
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 32, 32, 256)       0
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr (None, 32, 32, 128)       819328
_________________________________________________________________
batch_normalization_4 (Batch (None, 32, 32, 128)       512
_________________________________________________________________
re_lu_2 (ReLU)               (None, 32, 32, 128)       0
_________________________________________________________________
dropout_4 (Dropout)          (None, 32, 32, 128)       0
_________________________________________________________________
up_sampling2d_4 (UpSampling2 (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_transpose_4 (Conv2DTr (None, 64, 64, 3)         9603
_________________________________________________________________
activation_3 (Activation)    (None, 64, 64, 3)         0
```



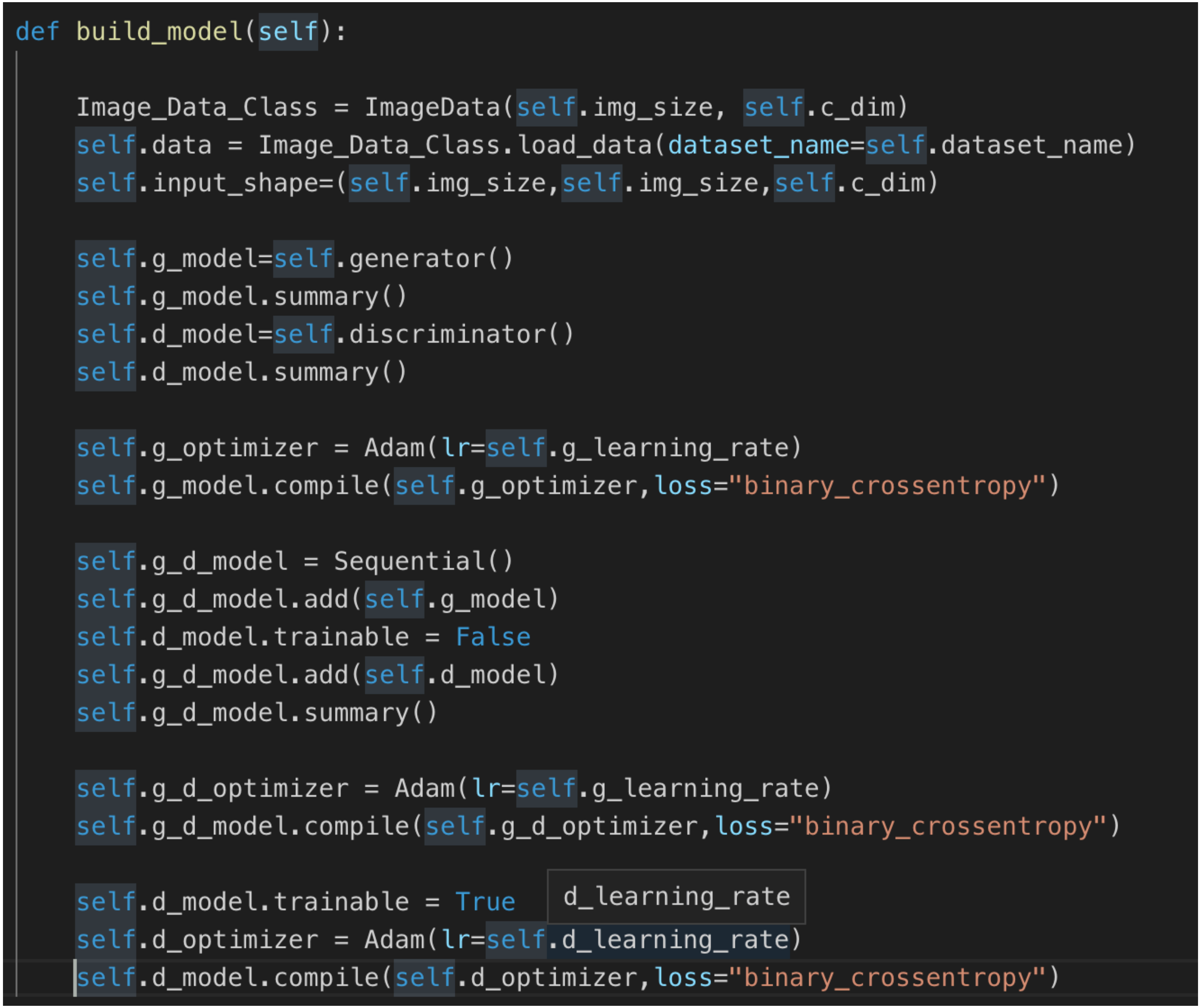
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 64, 64, 512)       14336
_________________________________________________________________
batch_normalization_5 (Batch (None, 64, 64, 512)       2048
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 64, 64, 512)       0
_________________________________________________________________
dropout_5 (Dropout)          (None, 64, 64, 512)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 512)       2359808
_________________________________________________________________
batch_normalization_6 (Batch (None, 32, 32, 512)       2048
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 32, 32, 512)       0
_________________________________________________________________
dropout_6 (Dropout)          (None, 32, 32, 512)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 128)       589952
_________________________________________________________________
batch_normalization_7 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 16, 16, 128)       0
_________________________________________________________________
dropout_7 (Dropout)          (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
batch_normalization_8 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 8, 8, 128)         0
_________________________________________________________________
dropout_8 (Dropout)          (None, 8, 8, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 8193
_________________________________________________________________
activation_4 (Activation)    (None, 1)                 0
=================================================================
Total params: 3,124,993
Trainable params: 3,122,433
Non-trainable params: 2,560
```



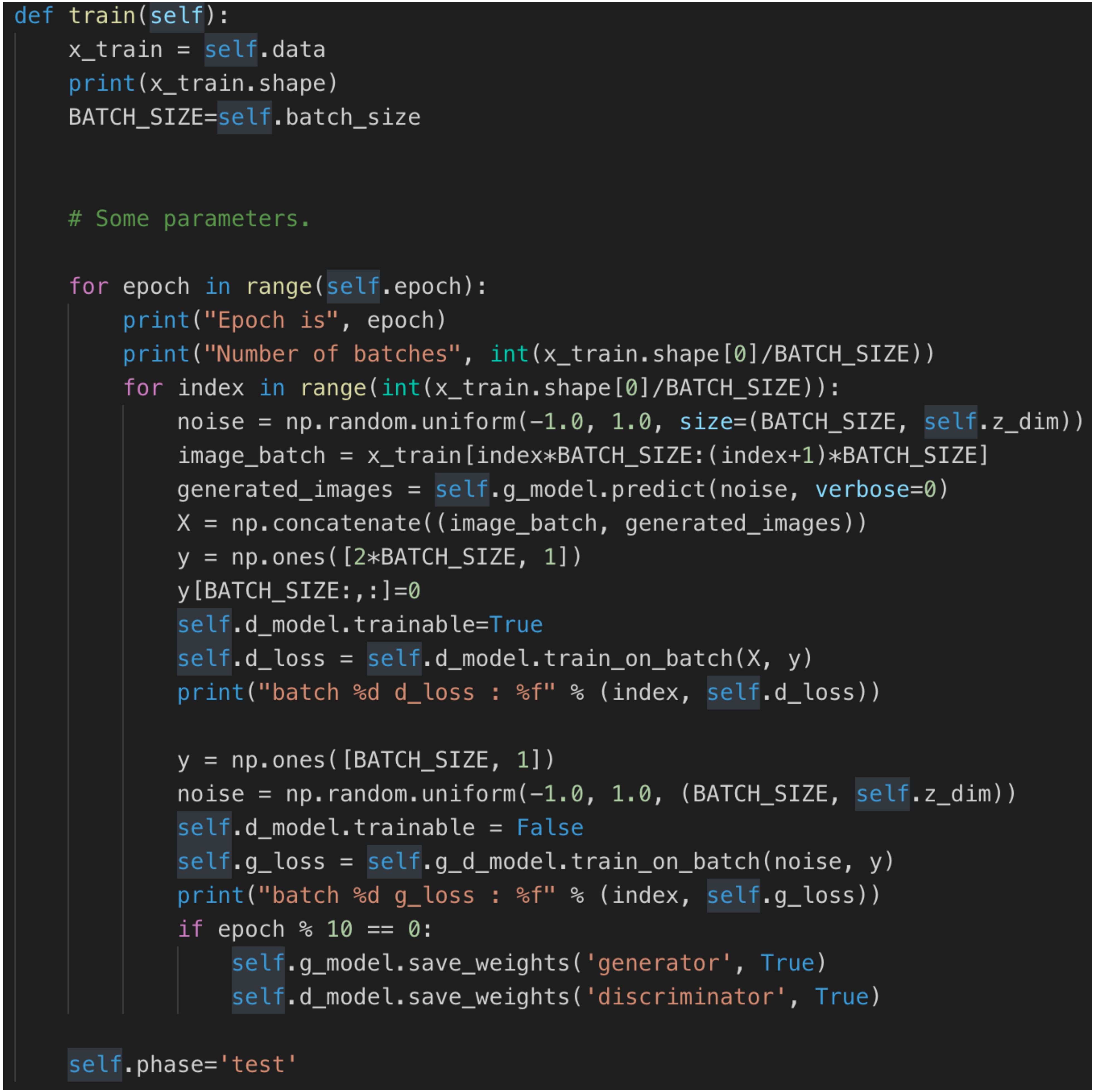
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential_1 (Sequential)    (None, 64, 64, 3)         19396355

_________________________________________________________________
sequential_2 (Sequential)    (None, 1)                 3124993
=================================================================
Total params: 22,521,348
Trainable params: 19,361,795
Non-trainable params: 3,159,553
```



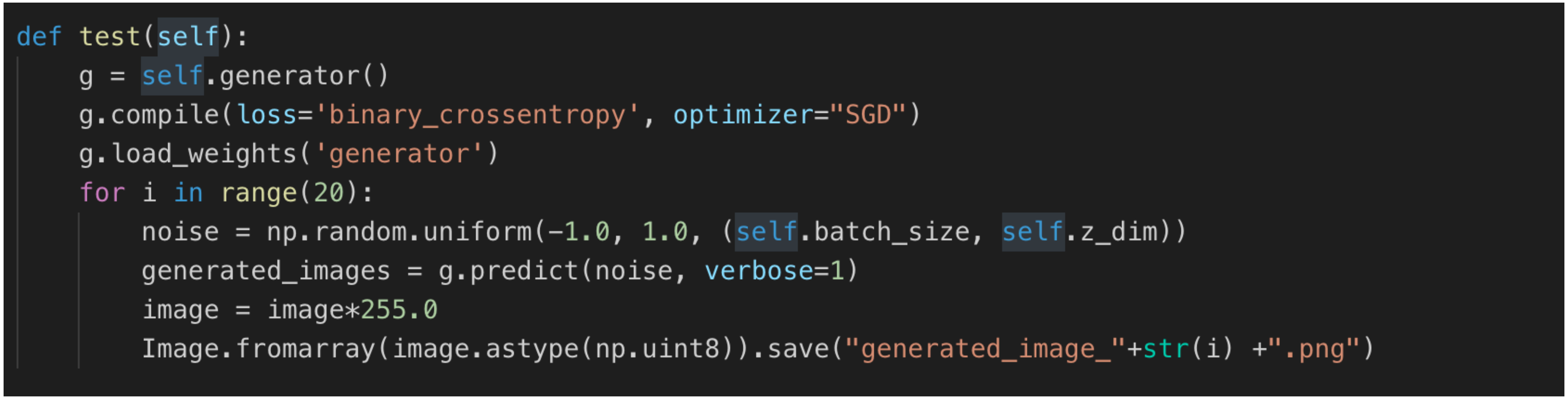
```
    def load_data(self,dataset_name,flag=1):
        if flag==0:
            TRAIN_DIR = os.path.join("./dataset", dataset_name)
            print(TRAIN_DIR)
            training_data = []
            for img in (os.listdir(TRAIN_DIR)):
                # print(img)
                path = os.path.join(TRAIN_DIR,img)
                # print(path)
                img = cv2.imread(path,cv2.IMREAD_COLOR)
                # print(img)
                img = cv2.resize(img, (self.img_size,self.img_size))
                training_data.append([np.array(img).astype('float32')])
            shuffle(training_data)
            x_train = np.vstack(training_data) / 255.0
            np.save(str(dataset_name) + '_train_data.npy', x_train)
            print(x_train.shape)
        else:
            x_train=np.load(str(dataset_name)+'_train_data.npy')
```



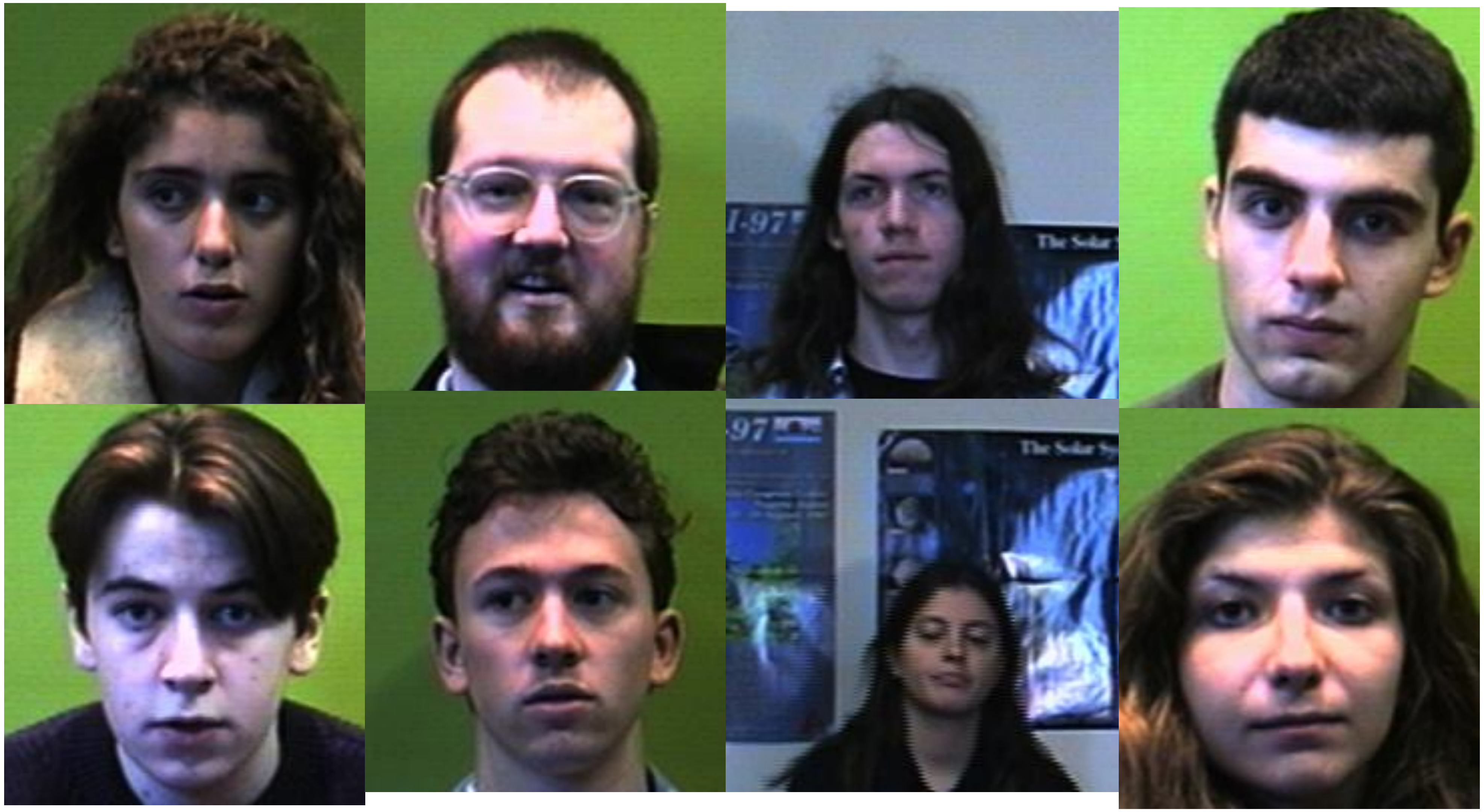
```
def build_model(self):

    Image_Data_Class = ImageData(self.img_size, self.c_dim)
    self.data = Image_Data_Class.load_data(dataset_name=self.dataset_name)
    self.input_shape=(self.img_size,self.img_size,self.c_dim)

    self.g_model=self.generator()
    self.g_model.summary()
    self.d_model=self.discriminator()
    self.d_model.summary()

    self.g_optimizer = Adam(lr=self.g_learning_rate)
    self.g_model.compile(self.g_optimizer,loss="binary_crossentropy")

    self.g_d_model = Sequential()
    self.g_d_model.add(self.g_model)
    self.d_model.trainable = False
    self.g_d_model.add(self.d_model)
    self.g_d_model.summary()

    self.g_d_optimizer = Adam(lr=self.g_learning_rate)
    self.g_d_model.compile(self.g_d_optimizer,loss="binary_crossentropy")

    self.d_model.trainable = True
    self.d_optimizer = Adam(lr=self.d_learning_rate)
    self.d_model.compile(self.d_optimizer,loss="binary_crossentropy")
```

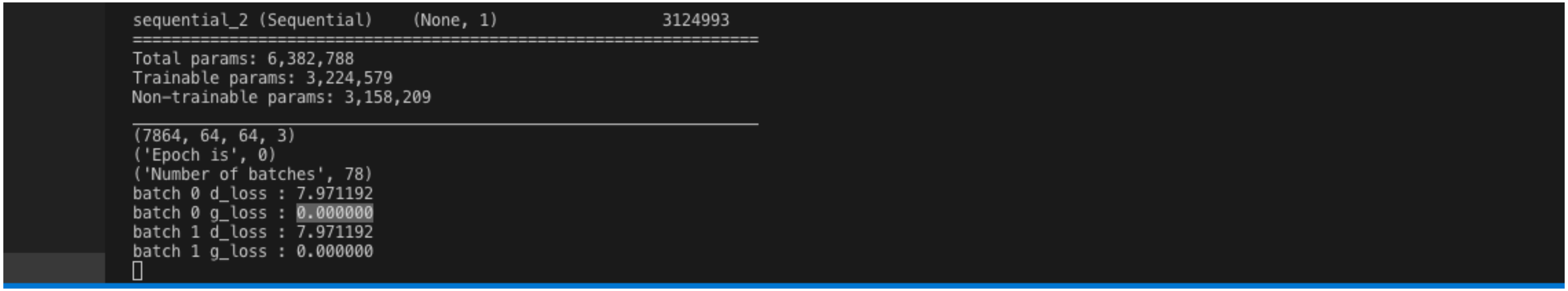


```
def train(self):
    x_train = self.data
    print(x_train.shape)
    BATCH_SIZE=self.batch_size


    # Some parameters.

    for epoch in range(self.epoch):
        print("Epoch is", epoch)
        print("Number of batches", int(x_train.shape[0]/BATCH_SIZE))
        for index in range(int(x_train.shape[0]/BATCH_SIZE)):
            noise = np.random.uniform(-1.0, 1.0, size=(BATCH_SIZE, self.z_dim))
            image_batch = x_train[index*BATCH_SIZE:(index+1)*BATCH_SIZE]
            generated_images = self.g_model.predict(noise, verbose=0)
            X = np.concatenate((image_batch, generated_images))
            y = np.ones([2*BATCH_SIZE, 1])
            y[BATCH_SIZE:,:]=0
            self.d_model.trainable=True
            self.d_loss = self.d_model.train_on_batch(X, y)
            print("batch %d d_loss : %f" % (index, self.d_loss))

            y = np.ones([BATCH_SIZE, 1])
            noise = np.random.uniform(-1.0, 1.0, (BATCH_SIZE, self.z_dim))
            self.d_model.trainable = False
            self.g_loss = self.g_d_model.train_on_batch(noise, y)
            print("batch %d g_loss : %f" % (index, self.g_loss))
            if epoch % 10 == 0:
                self.g_model.save_weights('generator', True)
                self.d_model.save_weights('discriminator', True)

    self.phase='test'
```



```
def test(self):
    g = self.generator()
    g.compile(loss='binary_crossentropy', optimizer="SGD")
    g.load_weights('generator')
    for i in range(20):
        noise = np.random.uniform(-1.0, 1.0, (self.batch_size, self.z_dim))
        generated_images = g.predict(noise, verbose=1)
        image = image*255.0
        Image.fromarray(image.astype(np.uint8)).save("generated_image_"+str(i) +".png")
```





```
sequential_2 (Sequential)    (None, 1)                 3124993
=================================================================
Total params: 6,382,788
Trainable params: 3,224,579
Non-trainable params: 3,158,209
_________________________________________________________________
(7864, 64, 64, 3)
('Epoch is', 0)
('Number of batches', 78)
batch 0 d_loss : 7.971192
batch 0 g_loss : 0.000000
batch 1 d_loss : 7.971192
batch 1 g_loss : 0.000000
```