In [ ]:
```
# Install TensorFlow
# !pip install -q tensorflow-gpu==2.0.0-rc0

try:
  %tensorflow_version 2.x  # Colab only.
except Exception:
  pass

import tensorflow as tf
print(tf.__version__)
```

```
`%tensorflow_version` only switches the major version: `1.x` or `2.x`.
You set: `2.x  # Colab only.`. This will be interpreted as: `2.x`.


TensorFlow is already loaded. Please restart the runtime to change versions.
2.0.0-rc0
```

In [ ]:
```
# More imports
from tensorflow.keras.layers import Input, Dense, LeakyReLU, Dropout, \
  BatchNormalization
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import SGD, Adam

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sys, os
```

In [ ]:
```
# Load in the data
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()

# map inputs to (-1, +1) for better training
x_train, x_test = x_train / 255.0 * 2 - 1, x_test / 255.0 * 2 - 1
print("x_train.shape:", x_train.shape)
```

```
x_train.shape: (60000, 28, 28)
```

In [ ]:
```
# Flatten the data
N, H, W = x_train.shape
D = H * W
x_train = x_train.reshape(-1, D)
x_test = x_test.reshape(-1, D)
```

In [ ]:
```
# Dimensionality of the latent space
latent_dim = 100
```

In [ ]:
```
# Get the generator model
def build_generator(latent_dim):
  i = Input(shape=(latent_dim,))
  x = Dense(256, activation=LeakyReLU(alpha=0.2))(i)
```

```
  x = BatchNormalization(momentum=0.7)(x)
  x = Dense(512, activation=LeakyReLU(alpha=0.2))(x)
  x = BatchNormalization(momentum=0.7)(x)
  x = Dense(1024, activation=LeakyReLU(alpha=0.2))(x)
  x = BatchNormalization(momentum=0.7)(x)
  x = Dense(D, activation='tanh')(x)

  model = Model(i, x)
  return model
```

In [ ]:

```
# Get the discriminator model
def build_discriminator(img_size):
  i = Input(shape=(img_size,))
  x = Dense(512, activation=LeakyReLU(alpha=0.2))(i)
  x = Dense(256, activation=LeakyReLU(alpha=0.2))(x)
  x = Dense(1, activation='sigmoid')(x)
  model = Model(i, x)
  return model
```

In [ ]:

```
# Compile both models in preparation for training


# Build and compile the discriminator
discriminator = build_discriminator(D)
discriminator.compile(
    loss='binary_crossentropy',
    optimizer=Adam(0.0002, 0.5),
    metrics=['accuracy'])

# Build and compile the combined model
generator = build_generator(latent_dim)

# Create an input to represent noise sample from latent space
z = Input(shape=(latent_dim,))

# Pass noise through generator to get an image
img = generator(z)

# Make sure only the generator is trained
discriminator.trainable = False

# The true output is fake, but we label them real!
fake_pred = discriminator(img)

# Create the combined model object
combined_model = Model(z, fake_pred)

# Compile the combined model
combined_model.compile(loss='binary_crossentropy', optimizer=Adam(0.0002, 0.5))
```

In [ ]:

```
# Train the GAN


# Config
batch_size = 32
epochs = 30000
```

```
sample_period = 200 # every `sample_period` steps generate and save some data


# Create batch labels to use when calling train_on_batch
ones = np.ones(batch_size)
zeros = np.zeros(batch_size)

# Store the losses
d_losses = []
g_losses = []

# Create a folder to store generated images
if not os.path.exists('gan_images'):
  os.makedirs('gan_images')
```

In [ ]:

```
# A function to generate a grid of random samples from the generator
# and save them to a file
def sample_images(epoch):
  rows, cols = 5, 5
  noise = np.random.randn(rows * cols, latent_dim)
  imgs = generator.predict(noise)

  # Rescale images 0 - 1
  imgs = 0.5 * imgs + 0.5

  fig, axs = plt.subplots(rows, cols)
  idx = 0
  for i in range(rows):
    for j in range(cols):
      axs[i,j].imshow(imgs[idx].reshape(H, W), cmap='gray')
      axs[i,j].axis('off')
      idx += 1
  fig.savefig("gan_images/%d.png" % epoch)
  plt.close()
```

In [ ]:

```
# Main training loop
for epoch in range(epochs):
  ############################
  ### Train discriminator ###
  ############################

  # Select a random batch of images
  idx = np.random.randint(0, x_train.shape[0], batch_size)
  real_imgs = x_train[idx]

  # Generate fake images
  noise = np.random.randn(batch_size, latent_dim)
  fake_imgs = generator.predict(noise)

  # Train the discriminator
  # both loss and accuracy are returned
  d_loss_real, d_acc_real = discriminator.train_on_batch(real_imgs, ones)
  d_loss_fake, d_acc_fake = discriminator.train_on_batch(fake_imgs, zeros)
  d_loss = 0.5 * (d_loss_real + d_loss_fake)
  d_acc  = 0.5 * (d_acc_real + d_acc_fake)


  ########################
```

```
  ### Train generator ###
  ########################

  noise = np.random.randn(batch_size, latent_dim)
  g_loss = combined_model.train_on_batch(noise, ones)

  # do it again!
  noise = np.random.randn(batch_size, latent_dim)
  g_loss = combined_model.train_on_batch(noise, ones)

  # Save the losses
  d_losses.append(d_loss)
  g_losses.append(g_loss)

  if epoch % 100 == 0:
    print(f"epoch: {epoch+1}/{epochs}, d_loss: {d_loss:.2f}, \
      d_acc: {d_acc:.2f}, g_loss: {g_loss:.2f}")

  if epoch % sample_period == 0:
    sample_images(epoch)
```

```
epoch: 1/30000, d_loss: 0.74,       d_acc: 0.42, g_loss: 0.70
epoch: 101/30000, d_loss: 0.04,       d_acc: 1.00, g_loss: 3.91
epoch: 201/30000, d_loss: 0.92,       d_acc: 0.41, g_loss: 0.70
epoch: 301/30000, d_loss: 0.71,       d_acc: 0.45, g_loss: 0.58
epoch: 401/30000, d_loss: 0.69,       d_acc: 0.48, g_loss: 0.61
epoch: 501/30000, d_loss: 0.67,       d_acc: 0.52, g_loss: 0.63
epoch: 601/30000, d_loss: 0.69,       d_acc: 0.45, g_loss: 0.63
epoch: 701/30000, d_loss: 0.68,       d_acc: 0.44, g_loss: 0.68
epoch: 801/30000, d_loss: 0.67,       d_acc: 0.64, g_loss: 0.68
epoch: 901/30000, d_loss: 0.65,       d_acc: 0.61, g_loss: 0.71
epoch: 1001/30000, d_loss: 0.66,       d_acc: 0.58, g_loss: 0.71
epoch: 1101/30000, d_loss: 0.66,       d_acc: 0.64, g_loss: 0.74
epoch: 1201/30000, d_loss: 0.63,       d_acc: 0.72, g_loss: 0.76
epoch: 1301/30000, d_loss: 0.68,       d_acc: 0.56, g_loss: 0.75
epoch: 1401/30000, d_loss: 0.66,       d_acc: 0.69, g_loss: 0.72
epoch: 1501/30000, d_loss: 0.67,       d_acc: 0.56, g_loss: 0.80
epoch: 1601/30000, d_loss: 0.67,       d_acc: 0.59, g_loss: 0.80
epoch: 1701/30000, d_loss: 0.64,       d_acc: 0.62, g_loss: 0.80
epoch: 1801/30000, d_loss: 0.64,       d_acc: 0.78, g_loss: 0.75
epoch: 1901/30000, d_loss: 0.65,       d_acc: 0.62, g_loss: 0.76
epoch: 2001/30000, d_loss: 0.66,       d_acc: 0.55, g_loss: 0.76
epoch: 2101/30000, d_loss: 0.65,       d_acc: 0.59, g_loss: 0.78
epoch: 2201/30000, d_loss: 0.69,       d_acc: 0.48, g_loss: 0.75
epoch: 2301/30000, d_loss: 0.65,       d_acc: 0.69, g_loss: 0.78
epoch: 2401/30000, d_loss: 0.64,       d_acc: 0.67, g_loss: 0.78
epoch: 2501/30000, d_loss: 0.69,       d_acc: 0.50, g_loss: 0.80
epoch: 2601/30000, d_loss: 0.68,       d_acc: 0.62, g_loss: 0.78
epoch: 2701/30000, d_loss: 0.67,       d_acc: 0.66, g_loss: 0.76
epoch: 2801/30000, d_loss: 0.68,       d_acc: 0.61, g_loss: 0.79
epoch: 2901/30000, d_loss: 0.69,       d_acc: 0.56, g_loss: 0.80
epoch: 3001/30000, d_loss: 0.68,       d_acc: 0.52, g_loss: 0.76
epoch: 3101/30000, d_loss: 0.66,       d_acc: 0.67, g_loss: 0.77
epoch: 3201/30000, d_loss: 0.71,       d_acc: 0.52, g_loss: 0.77
epoch: 3301/30000, d_loss: 0.67,       d_acc: 0.62, g_loss: 0.77
epoch: 3401/30000, d_loss: 0.64,       d_acc: 0.64, g_loss: 0.78
epoch: 3501/30000, d_loss: 0.72,       d_acc: 0.48, g_loss: 0.77
epoch: 3601/30000, d_loss: 0.67,       d_acc: 0.59, g_loss: 0.82
epoch: 3701/30000, d_loss: 0.69,       d_acc: 0.58, g_loss: 0.80
epoch: 3801/30000, d_loss: 0.68,       d_acc: 0.52, g_loss: 0.73
epoch: 3901/30000, d_loss: 0.70,       d_acc: 0.45, g_loss: 0.74
epoch: 4001/30000, d_loss: 0.67,       d_acc: 0.58, g_loss: 0.79
epoch: 4101/30000, d_loss: 0.66,       d_acc: 0.59, g_loss: 0.76
```

```
epoch: 4201/30000, d_loss: 0.69,       d_acc: 0.52, g_loss: 0.74
epoch: 4301/30000, d_loss: 0.70,       d_acc: 0.50, g_loss: 0.76
epoch: 4401/30000, d_loss: 0.68,       d_acc: 0.56, g_loss: 0.76
epoch: 4501/30000, d_loss: 0.70,       d_acc: 0.55, g_loss: 0.78
epoch: 4601/30000, d_loss: 0.67,       d_acc: 0.58, g_loss: 0.78
epoch: 4701/30000, d_loss: 0.66,       d_acc: 0.59, g_loss: 0.72
epoch: 4801/30000, d_loss: 0.67,       d_acc: 0.58, g_loss: 0.76
epoch: 4901/30000, d_loss: 0.72,       d_acc: 0.52, g_loss: 0.73
epoch: 5001/30000, d_loss: 0.73,       d_acc: 0.42, g_loss: 0.73
epoch: 5101/30000, d_loss: 0.66,       d_acc: 0.67, g_loss: 0.78
epoch: 5201/30000, d_loss: 0.67,       d_acc: 0.62, g_loss: 0.75
epoch: 5301/30000, d_loss: 0.67,       d_acc: 0.53, g_loss: 0.77
epoch: 5401/30000, d_loss: 0.69,       d_acc: 0.56, g_loss: 0.78
epoch: 5501/30000, d_loss: 0.68,       d_acc: 0.61, g_loss: 0.75
epoch: 5601/30000, d_loss: 0.72,       d_acc: 0.50, g_loss: 0.75
epoch: 5701/30000, d_loss: 0.69,       d_acc: 0.55, g_loss: 0.76
epoch: 5801/30000, d_loss: 0.72,       d_acc: 0.47, g_loss: 0.76
epoch: 5901/30000, d_loss: 0.70,       d_acc: 0.47, g_loss: 0.75
epoch: 6001/30000, d_loss: 0.70,       d_acc: 0.56, g_loss: 0.77
epoch: 6101/30000, d_loss: 0.67,       d_acc: 0.59, g_loss: 0.80
epoch: 6201/30000, d_loss: 0.70,       d_acc: 0.50, g_loss: 0.76
epoch: 6301/30000, d_loss: 0.73,       d_acc: 0.48, g_loss: 0.75
epoch: 6401/30000, d_loss: 0.67,       d_acc: 0.61, g_loss: 0.75
epoch: 6501/30000, d_loss: 0.70,       d_acc: 0.55, g_loss: 0.79
epoch: 6601/30000, d_loss: 0.68,       d_acc: 0.59, g_loss: 0.77
epoch: 6701/30000, d_loss: 0.69,       d_acc: 0.53, g_loss: 0.76
epoch: 6801/30000, d_loss: 0.70,       d_acc: 0.53, g_loss: 0.76
epoch: 6901/30000, d_loss: 0.70,       d_acc: 0.53, g_loss: 0.77
epoch: 7001/30000, d_loss: 0.71,       d_acc: 0.52, g_loss: 0.76
epoch: 7101/30000, d_loss: 0.66,       d_acc: 0.59, g_loss: 0.77
epoch: 7201/30000, d_loss: 0.67,       d_acc: 0.61, g_loss: 0.80
epoch: 7301/30000, d_loss: 0.70,       d_acc: 0.52, g_loss: 0.77
epoch: 7401/30000, d_loss: 0.69,       d_acc: 0.53, g_loss: 0.76
epoch: 7501/30000, d_loss: 0.70,       d_acc: 0.52, g_loss: 0.76
epoch: 7601/30000, d_loss: 0.70,       d_acc: 0.48, g_loss: 0.77
epoch: 7701/30000, d_loss: 0.70,       d_acc: 0.45, g_loss: 0.76
epoch: 7801/30000, d_loss: 0.69,       d_acc: 0.56, g_loss: 0.75
epoch: 7901/30000, d_loss: 0.68,       d_acc: 0.58, g_loss: 0.76
epoch: 8001/30000, d_loss: 0.72,       d_acc: 0.44, g_loss: 0.77
epoch: 8101/30000, d_loss: 0.67,       d_acc: 0.55, g_loss: 0.78
epoch: 8201/30000, d_loss: 0.69,       d_acc: 0.59, g_loss: 0.76
epoch: 8301/30000, d_loss: 0.71,       d_acc: 0.47, g_loss: 0.77
epoch: 8401/30000, d_loss: 0.69,       d_acc: 0.52, g_loss: 0.78
epoch: 8501/30000, d_loss: 0.64,       d_acc: 0.69, g_loss: 0.78
epoch: 8601/30000, d_loss: 0.70,       d_acc: 0.53, g_loss: 0.79
epoch: 8701/30000, d_loss: 0.67,       d_acc: 0.53, g_loss: 0.79
epoch: 8801/30000, d_loss: 0.69,       d_acc: 0.52, g_loss: 0.75
epoch: 8901/30000, d_loss: 0.70,       d_acc: 0.52, g_loss: 0.76
epoch: 9001/30000, d_loss: 0.70,       d_acc: 0.48, g_loss: 0.76
epoch: 9101/30000, d_loss: 0.67,       d_acc: 0.55, g_loss: 0.74
epoch: 9201/30000, d_loss: 0.67,       d_acc: 0.56, g_loss: 0.77
epoch: 9301/30000, d_loss: 0.71,       d_acc: 0.53, g_loss: 0.77
epoch: 9401/30000, d_loss: 0.67,       d_acc: 0.61, g_loss: 0.78
epoch: 9501/30000, d_loss: 0.68,       d_acc: 0.61, g_loss: 0.77
epoch: 9601/30000, d_loss: 0.69,       d_acc: 0.56, g_loss: 0.79
epoch: 9701/30000, d_loss: 0.70,       d_acc: 0.52, g_loss: 0.71
epoch: 9801/30000, d_loss: 0.68,       d_acc: 0.55, g_loss: 0.78
epoch: 9901/30000, d_loss: 0.66,       d_acc: 0.64, g_loss: 0.76
epoch: 10001/30000, d_loss: 0.71,      d_acc: 0.47, g_loss: 0.75
epoch: 10101/30000, d_loss: 0.71,      d_acc: 0.48, g_loss: 0.79
epoch: 10201/30000, d_loss: 0.69,      d_acc: 0.55, g_loss: 0.78
epoch: 10301/30000, d_loss: 0.70,      d_acc: 0.58, g_loss: 0.80
epoch: 10401/30000, d_loss: 0.70,      d_acc: 0.50, g_loss: 0.80
epoch: 10501/30000, d_loss: 0.69,      d_acc: 0.59, g_loss: 0.76
epoch: 10601/30000, d_loss: 0.69,      d_acc: 0.52, g_loss: 0.73
```

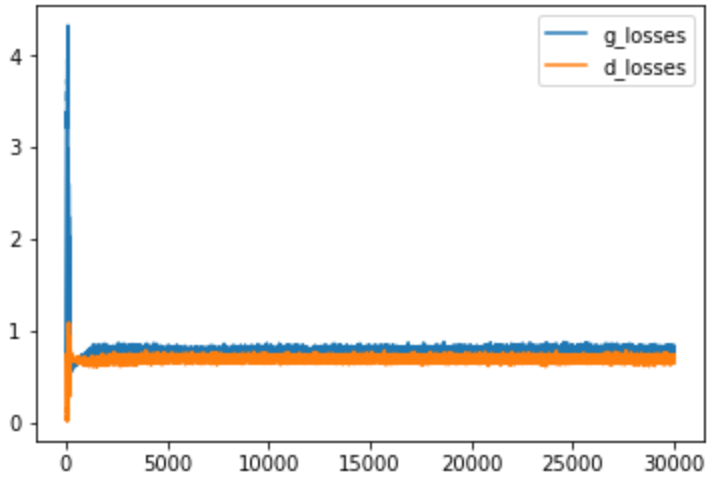
```
epoch: 10701/30000, d_loss: 0.70,       d_acc: 0.56, g_loss: 0.75
epoch: 10801/30000, d_loss: 0.69,       d_acc: 0.55, g_loss: 0.75
epoch: 10901/30000, d_loss: 0.67,       d_acc: 0.61, g_loss: 0.80
epoch: 11001/30000, d_loss: 0.67,       d_acc: 0.61, g_loss: 0.78
epoch: 11101/30000, d_loss: 0.69,       d_acc: 0.55, g_loss: 0.77
epoch: 11201/30000, d_loss: 0.69,       d_acc: 0.56, g_loss: 0.77
epoch: 11301/30000, d_loss: 0.72,       d_acc: 0.52, g_loss: 0.76
epoch: 11401/30000, d_loss: 0.68,       d_acc: 0.58, g_loss: 0.72
epoch: 11501/30000, d_loss: 0.70,       d_acc: 0.50, g_loss: 0.76
epoch: 11601/30000, d_loss: 0.69,       d_acc: 0.52, g_loss: 0.77
epoch: 11701/30000, d_loss: 0.69,       d_acc: 0.50, g_loss: 0.77
epoch: 11801/30000, d_loss: 0.67,       d_acc: 0.61, g_loss: 0.76
epoch: 11901/30000, d_loss: 0.71,       d_acc: 0.48, g_loss: 0.79
epoch: 12001/30000, d_loss: 0.71,       d_acc: 0.50, g_loss: 0.78
epoch: 12101/30000, d_loss: 0.72,       d_acc: 0.52, g_loss: 0.75
epoch: 12201/30000, d_loss: 0.69,       d_acc: 0.53, g_loss: 0.74
epoch: 12301/30000, d_loss: 0.68,       d_acc: 0.53, g_loss: 0.72
epoch: 12401/30000, d_loss: 0.65,       d_acc: 0.67, g_loss: 0.79
epoch: 12501/30000, d_loss: 0.66,       d_acc: 0.62, g_loss: 0.73
epoch: 12601/30000, d_loss: 0.69,       d_acc: 0.53, g_loss: 0.79
epoch: 12701/30000, d_loss: 0.65,       d_acc: 0.69, g_loss: 0.73
epoch: 12801/30000, d_loss: 0.69,       d_acc: 0.53, g_loss: 0.80
epoch: 12901/30000, d_loss: 0.70,       d_acc: 0.52, g_loss: 0.78
epoch: 13001/30000, d_loss: 0.68,       d_acc: 0.58, g_loss: 0.81
epoch: 13101/30000, d_loss: 0.67,       d_acc: 0.64, g_loss: 0.73
epoch: 13201/30000, d_loss: 0.71,       d_acc: 0.56, g_loss: 0.77
epoch: 13301/30000, d_loss: 0.70,       d_acc: 0.53, g_loss: 0.80
epoch: 13401/30000, d_loss: 0.69,       d_acc: 0.55, g_loss: 0.82
epoch: 13501/30000, d_loss: 0.70,       d_acc: 0.52, g_loss: 0.75
epoch: 13601/30000, d_loss: 0.72,       d_acc: 0.50, g_loss: 0.78
epoch: 13701/30000, d_loss: 0.67,       d_acc: 0.53, g_loss: 0.75
epoch: 13801/30000, d_loss: 0.71,       d_acc: 0.48, g_loss: 0.75
epoch: 13901/30000, d_loss: 0.72,       d_acc: 0.47, g_loss: 0.76
epoch: 14001/30000, d_loss: 0.67,       d_acc: 0.55, g_loss: 0.76
epoch: 14101/30000, d_loss: 0.67,       d_acc: 0.61, g_loss: 0.75
epoch: 14201/30000, d_loss: 0.70,       d_acc: 0.58, g_loss: 0.75
epoch: 14301/30000, d_loss: 0.71,       d_acc: 0.48, g_loss: 0.74
epoch: 14401/30000, d_loss: 0.68,       d_acc: 0.59, g_loss: 0.71
epoch: 14501/30000, d_loss: 0.68,       d_acc: 0.56, g_loss: 0.81
epoch: 14601/30000, d_loss: 0.71,       d_acc: 0.48, g_loss: 0.76
epoch: 14701/30000, d_loss: 0.72,       d_acc: 0.50, g_loss: 0.73
epoch: 14801/30000, d_loss: 0.69,       d_acc: 0.45, g_loss: 0.78
epoch: 14901/30000, d_loss: 0.70,       d_acc: 0.48, g_loss: 0.77
epoch: 15001/30000, d_loss: 0.64,       d_acc: 0.67, g_loss: 0.80
epoch: 15101/30000, d_loss: 0.70,       d_acc: 0.48, g_loss: 0.75
epoch: 15201/30000, d_loss: 0.65,       d_acc: 0.66, g_loss: 0.73
epoch: 15301/30000, d_loss: 0.73,       d_acc: 0.44, g_loss: 0.72
epoch: 15401/30000, d_loss: 0.71,       d_acc: 0.48, g_loss: 0.76
epoch: 15501/30000, d_loss: 0.69,       d_acc: 0.53, g_loss: 0.75
epoch: 15601/30000, d_loss: 0.65,       d_acc: 0.66, g_loss: 0.77
epoch: 15701/30000, d_loss: 0.68,       d_acc: 0.58, g_loss: 0.78
epoch: 15801/30000, d_loss: 0.70,       d_acc: 0.50, g_loss: 0.71
epoch: 15901/30000, d_loss: 0.68,       d_acc: 0.59, g_loss: 0.77
epoch: 16001/30000, d_loss: 0.69,       d_acc: 0.55, g_loss: 0.81
epoch: 16101/30000, d_loss: 0.66,       d_acc: 0.66, g_loss: 0.77
epoch: 16201/30000, d_loss: 0.69,       d_acc: 0.56, g_loss: 0.74
epoch: 16301/30000, d_loss: 0.68,       d_acc: 0.61, g_loss: 0.74
epoch: 16401/30000, d_loss: 0.69,       d_acc: 0.58, g_loss: 0.79
epoch: 16501/30000, d_loss: 0.69,       d_acc: 0.56, g_loss: 0.80
epoch: 16601/30000, d_loss: 0.68,       d_acc: 0.56, g_loss: 0.80
epoch: 16701/30000, d_loss: 0.71,       d_acc: 0.50, g_loss: 0.77
epoch: 16801/30000, d_loss: 0.68,       d_acc: 0.56, g_loss: 0.75
epoch: 16901/30000, d_loss: 0.67,       d_acc: 0.58, g_loss: 0.78
epoch: 17001/30000, d_loss: 0.65,       d_acc: 0.64, g_loss: 0.76
epoch: 17101/30000, d_loss: 0.72,       d_acc: 0.45, g_loss: 0.77
```

```
epoch: 17201/30000, d_loss: 0.68,       d_acc: 0.53, g_loss: 0.77
epoch: 17301/30000, d_loss: 0.67,       d_acc: 0.59, g_loss: 0.81
epoch: 17401/30000, d_loss: 0.70,       d_acc: 0.52, g_loss: 0.83
epoch: 17501/30000, d_loss: 0.70,       d_acc: 0.47, g_loss: 0.78
epoch: 17601/30000, d_loss: 0.67,       d_acc: 0.64, g_loss: 0.83
epoch: 17701/30000, d_loss: 0.68,       d_acc: 0.55, g_loss: 0.78
epoch: 17801/30000, d_loss: 0.72,       d_acc: 0.48, g_loss: 0.76
epoch: 17901/30000, d_loss: 0.69,       d_acc: 0.59, g_loss: 0.81
epoch: 18001/30000, d_loss: 0.70,       d_acc: 0.52, g_loss: 0.74
epoch: 18101/30000, d_loss: 0.67,       d_acc: 0.66, g_loss: 0.76
epoch: 18201/30000, d_loss: 0.71,       d_acc: 0.48, g_loss: 0.82
epoch: 18301/30000, d_loss: 0.72,       d_acc: 0.45, g_loss: 0.77
epoch: 18401/30000, d_loss: 0.70,       d_acc: 0.47, g_loss: 0.74
epoch: 18501/30000, d_loss: 0.69,       d_acc: 0.58, g_loss: 0.78
epoch: 18601/30000, d_loss: 0.68,       d_acc: 0.58, g_loss: 0.77
epoch: 18701/30000, d_loss: 0.69,       d_acc: 0.52, g_loss: 0.81
epoch: 18801/30000, d_loss: 0.69,       d_acc: 0.52, g_loss: 0.76
epoch: 18901/30000, d_loss: 0.69,       d_acc: 0.53, g_loss: 0.77
epoch: 19001/30000, d_loss: 0.68,       d_acc: 0.64, g_loss: 0.77
epoch: 19101/30000, d_loss: 0.70,       d_acc: 0.50, g_loss: 0.73
epoch: 19201/30000, d_loss: 0.70,       d_acc: 0.47, g_loss: 0.78
epoch: 19301/30000, d_loss: 0.69,       d_acc: 0.55, g_loss: 0.71
epoch: 19401/30000, d_loss: 0.69,       d_acc: 0.55, g_loss: 0.77
epoch: 19501/30000, d_loss: 0.67,       d_acc: 0.52, g_loss: 0.76
epoch: 19601/30000, d_loss: 0.69,       d_acc: 0.55, g_loss: 0.81
epoch: 19701/30000, d_loss: 0.66,       d_acc: 0.55, g_loss: 0.76
epoch: 19801/30000, d_loss: 0.69,       d_acc: 0.53, g_loss: 0.78
epoch: 19901/30000, d_loss: 0.68,       d_acc: 0.52, g_loss: 0.79
epoch: 20001/30000, d_loss: 0.71,       d_acc: 0.50, g_loss: 0.77
epoch: 20101/30000, d_loss: 0.70,       d_acc: 0.50, g_loss: 0.76
epoch: 20201/30000, d_loss: 0.64,       d_acc: 0.67, g_loss: 0.78
epoch: 20301/30000, d_loss: 0.68,       d_acc: 0.58, g_loss: 0.75
epoch: 20401/30000, d_loss: 0.66,       d_acc: 0.59, g_loss: 0.77
epoch: 20501/30000, d_loss: 0.67,       d_acc: 0.55, g_loss: 0.75
epoch: 20601/30000, d_loss: 0.68,       d_acc: 0.53, g_loss: 0.77
epoch: 20701/30000, d_loss: 0.67,       d_acc: 0.59, g_loss: 0.80
epoch: 20801/30000, d_loss: 0.68,       d_acc: 0.58, g_loss: 0.77
epoch: 20901/30000, d_loss: 0.70,       d_acc: 0.47, g_loss: 0.77
epoch: 21001/30000, d_loss: 0.75,       d_acc: 0.42, g_loss: 0.76
epoch: 21101/30000, d_loss: 0.69,       d_acc: 0.53, g_loss: 0.74
epoch: 21201/30000, d_loss: 0.67,       d_acc: 0.61, g_loss: 0.79
epoch: 21301/30000, d_loss: 0.67,       d_acc: 0.55, g_loss: 0.78
epoch: 21401/30000, d_loss: 0.67,       d_acc: 0.55, g_loss: 0.81
epoch: 21501/30000, d_loss: 0.69,       d_acc: 0.48, g_loss: 0.79
epoch: 21601/30000, d_loss: 0.70,       d_acc: 0.53, g_loss: 0.76
epoch: 21701/30000, d_loss: 0.66,       d_acc: 0.64, g_loss: 0.76
epoch: 21801/30000, d_loss: 0.66,       d_acc: 0.66, g_loss: 0.78
epoch: 21901/30000, d_loss: 0.68,       d_acc: 0.56, g_loss: 0.78
epoch: 22001/30000, d_loss: 0.68,       d_acc: 0.59, g_loss: 0.80
epoch: 22101/30000, d_loss: 0.71,       d_acc: 0.39, g_loss: 0.79
epoch: 22201/30000, d_loss: 0.69,       d_acc: 0.50, g_loss: 0.73
epoch: 22301/30000, d_loss: 0.69,       d_acc: 0.47, g_loss: 0.75
epoch: 22401/30000, d_loss: 0.67,       d_acc: 0.53, g_loss: 0.80
epoch: 22501/30000, d_loss: 0.67,       d_acc: 0.53, g_loss: 0.77
epoch: 22601/30000, d_loss: 0.70,       d_acc: 0.44, g_loss: 0.77
epoch: 22701/30000, d_loss: 0.66,       d_acc: 0.64, g_loss: 0.80
epoch: 22801/30000, d_loss: 0.69,       d_acc: 0.48, g_loss: 0.74
epoch: 22901/30000, d_loss: 0.69,       d_acc: 0.53, g_loss: 0.76
epoch: 23001/30000, d_loss: 0.69,       d_acc: 0.56, g_loss: 0.79
epoch: 23101/30000, d_loss: 0.72,       d_acc: 0.39, g_loss: 0.81
epoch: 23201/30000, d_loss: 0.68,       d_acc: 0.58, g_loss: 0.76
epoch: 23301/30000, d_loss: 0.71,       d_acc: 0.53, g_loss: 0.76
epoch: 23401/30000, d_loss: 0.69,       d_acc: 0.50, g_loss: 0.81
epoch: 23501/30000, d_loss: 0.69,       d_acc: 0.62, g_loss: 0.79
epoch: 23601/30000, d_loss: 0.69,       d_acc: 0.56, g_loss: 0.75
```

```
epoch: 23701/30000, d_loss: 0.73,       d_acc: 0.39, g_loss: 0.82
epoch: 23801/30000, d_loss: 0.68,       d_acc: 0.58, g_loss: 0.76
epoch: 23901/30000, d_loss: 0.68,       d_acc: 0.62, g_loss: 0.76
epoch: 24001/30000, d_loss: 0.69,       d_acc: 0.42, g_loss: 0.79
epoch: 24101/30000, d_loss: 0.66,       d_acc: 0.62, g_loss: 0.79
epoch: 24201/30000, d_loss: 0.66,       d_acc: 0.59, g_loss: 0.77
epoch: 24301/30000, d_loss: 0.66,       d_acc: 0.64, g_loss: 0.81
epoch: 24401/30000, d_loss: 0.67,       d_acc: 0.56, g_loss: 0.81
epoch: 24501/30000, d_loss: 0.71,       d_acc: 0.44, g_loss: 0.78
epoch: 24601/30000, d_loss: 0.65,       d_acc: 0.64, g_loss: 0.84
epoch: 24701/30000, d_loss: 0.68,       d_acc: 0.50, g_loss: 0.82
epoch: 24801/30000, d_loss: 0.68,       d_acc: 0.58, g_loss: 0.75
epoch: 24901/30000, d_loss: 0.72,       d_acc: 0.47, g_loss: 0.75
epoch: 25001/30000, d_loss: 0.67,       d_acc: 0.58, g_loss: 0.77
epoch: 25101/30000, d_loss: 0.67,       d_acc: 0.61, g_loss: 0.76
epoch: 25201/30000, d_loss: 0.67,       d_acc: 0.56, g_loss: 0.81
epoch: 25301/30000, d_loss: 0.69,       d_acc: 0.42, g_loss: 0.74
epoch: 25401/30000, d_loss: 0.69,       d_acc: 0.58, g_loss: 0.80
epoch: 25501/30000, d_loss: 0.69,       d_acc: 0.55, g_loss: 0.84
epoch: 25601/30000, d_loss: 0.71,       d_acc: 0.52, g_loss: 0.74
epoch: 25701/30000, d_loss: 0.69,       d_acc: 0.50, g_loss: 0.79
epoch: 25801/30000, d_loss: 0.68,       d_acc: 0.56, g_loss: 0.79
epoch: 25901/30000, d_loss: 0.67,       d_acc: 0.59, g_loss: 0.74
epoch: 26001/30000, d_loss: 0.67,       d_acc: 0.67, g_loss: 0.76
epoch: 26101/30000, d_loss: 0.70,       d_acc: 0.55, g_loss: 0.76
epoch: 26201/30000, d_loss: 0.68,       d_acc: 0.50, g_loss: 0.80
epoch: 26301/30000, d_loss: 0.68,       d_acc: 0.59, g_loss: 0.80
epoch: 26401/30000, d_loss: 0.70,       d_acc: 0.48, g_loss: 0.76
epoch: 26501/30000, d_loss: 0.70,       d_acc: 0.44, g_loss: 0.81
epoch: 26601/30000, d_loss: 0.67,       d_acc: 0.61, g_loss: 0.78
epoch: 26701/30000, d_loss: 0.68,       d_acc: 0.53, g_loss: 0.75
epoch: 26801/30000, d_loss: 0.69,       d_acc: 0.64, g_loss: 0.78
epoch: 26901/30000, d_loss: 0.67,       d_acc: 0.58, g_loss: 0.82
epoch: 27001/30000, d_loss: 0.70,       d_acc: 0.52, g_loss: 0.77
epoch: 27101/30000, d_loss: 0.72,       d_acc: 0.53, g_loss: 0.79
epoch: 27201/30000, d_loss: 0.67,       d_acc: 0.56, g_loss: 0.73
epoch: 27301/30000, d_loss: 0.69,       d_acc: 0.58, g_loss: 0.77
epoch: 27401/30000, d_loss: 0.70,       d_acc: 0.52, g_loss: 0.78
epoch: 27501/30000, d_loss: 0.70,       d_acc: 0.48, g_loss: 0.73
epoch: 27601/30000, d_loss: 0.65,       d_acc: 0.62, g_loss: 0.76
epoch: 27701/30000, d_loss: 0.70,       d_acc: 0.56, g_loss: 0.78
epoch: 27801/30000, d_loss: 0.70,       d_acc: 0.55, g_loss: 0.79
epoch: 27901/30000, d_loss: 0.68,       d_acc: 0.52, g_loss: 0.74
epoch: 28001/30000, d_loss: 0.66,       d_acc: 0.58, g_loss: 0.76
epoch: 28101/30000, d_loss: 0.69,       d_acc: 0.55, g_loss: 0.79
epoch: 28201/30000, d_loss: 0.72,       d_acc: 0.44, g_loss: 0.78
epoch: 28301/30000, d_loss: 0.68,       d_acc: 0.55, g_loss: 0.74
epoch: 28401/30000, d_loss: 0.66,       d_acc: 0.59, g_loss: 0.75
epoch: 28501/30000, d_loss: 0.70,       d_acc: 0.50, g_loss: 0.78
epoch: 28601/30000, d_loss: 0.68,       d_acc: 0.58, g_loss: 0.77
epoch: 28701/30000, d_loss: 0.68,       d_acc: 0.62, g_loss: 0.76
epoch: 28801/30000, d_loss: 0.71,       d_acc: 0.53, g_loss: 0.79
epoch: 28901/30000, d_loss: 0.70,       d_acc: 0.45, g_loss: 0.79
epoch: 29001/30000, d_loss: 0.70,       d_acc: 0.52, g_loss: 0.76
epoch: 29101/30000, d_loss: 0.69,       d_acc: 0.50, g_loss: 0.81
epoch: 29201/30000, d_loss: 0.70,       d_acc: 0.50, g_loss: 0.74
epoch: 29301/30000, d_loss: 0.69,       d_acc: 0.45, g_loss: 0.80
epoch: 29401/30000, d_loss: 0.68,       d_acc: 0.59, g_loss: 0.76
epoch: 29501/30000, d_loss: 0.69,       d_acc: 0.61, g_loss: 0.74
epoch: 29601/30000, d_loss: 0.68,       d_acc: 0.55, g_loss: 0.75
epoch: 29701/30000, d_loss: 0.65,       d_acc: 0.61, g_loss: 0.78
epoch: 29801/30000, d_loss: 0.66,       d_acc: 0.61, g_loss: 0.75
epoch: 29901/30000, d_loss: 0.66,       d_acc: 0.64, g_loss: 0.78
```

In [ ]:
```
plt.plot(g_losses, label='g_losses')
plt.plot(d_losses, label='d_losses')
plt.legend()
```

Out[ ]: <matplotlib.legend.Legend at 0x7f46a05ca748>

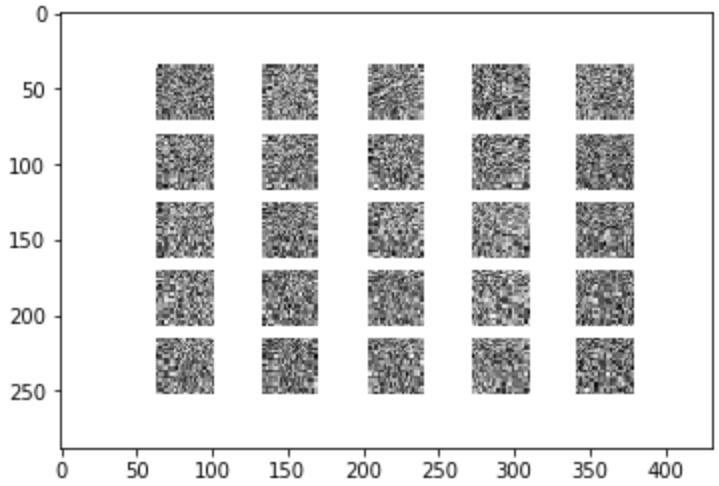


In [ ]:
```
!ls gan_images
```

```
0.png      13800.png  17800.png  21600.png  25600.png  29600.png  6600.png
10000.png  14000.png  18000.png  21800.png  25800.png  29800.png  6800.png
1000.png   1400.png   1800.png   22000.png  26000.png  3000.png   7000.png
10200.png  14200.png  18200.png  2200.png   2600.png   3200.png   7200.png
10400.png  14400.png  18400.png  22200.png  26200.png  3400.png   7400.png
10600.png  14600.png  18600.png  22400.png  26400.png  3600.png   7600.png
10800.png  14800.png  18800.png  22600.png  26600.png  3800.png   7800.png
11000.png  15000.png  19000.png  22800.png  26800.png  4000.png   8000.png
11200.png  15200.png  19200.png  23000.png  27000.png  400.png    800.png
11400.png  15400.png  19400.png  23200.png  27200.png  4200.png   8200.png
11600.png  15600.png  19600.png  23400.png  27400.png  4400.png   8400.png
11800.png  15800.png  19800.png  23600.png  27600.png  4600.png   8600.png
12000.png  16000.png  20000.png  23800.png  27800.png  4800.png   8800.png
1200.png   1600.png   2000.png   24000.png  28000.png  5000.png   9000.png
12200.png  16200.png  200.png    2400.png   2800.png   5200.png   9200.png
12400.png  16400.png  20200.png  24200.png  28200.png  5400.png   9400.png
12600.png  16600.png  20400.png  24400.png  28400.png  5600.png   9600.png
12800.png  16800.png  20600.png  24600.png  28600.png  5800.png   9800.png
13000.png  17000.png  20800.png  24800.png  28800.png  6000.png
13200.png  17200.png  21000.png  25000.png  29000.png  600.png
13400.png  17400.png  21200.png  25200.png  29200.png  6200.png
13600.png  17600.png  21400.png  25400.png  29400.png  6400.png
```

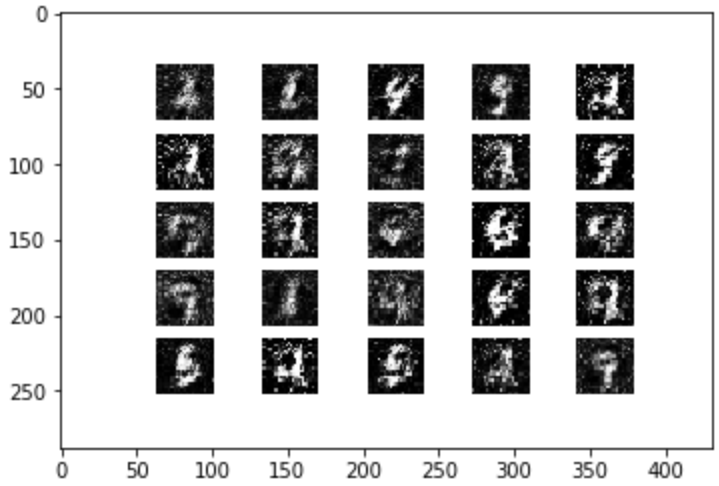
In [ ]:
```
from skimage.io import imread
a = imread('gan_images/0.png')
plt.imshow(a)
```

Out[ ]: <matplotlib.image.AxesImage at 0x7f46a05a69b0>



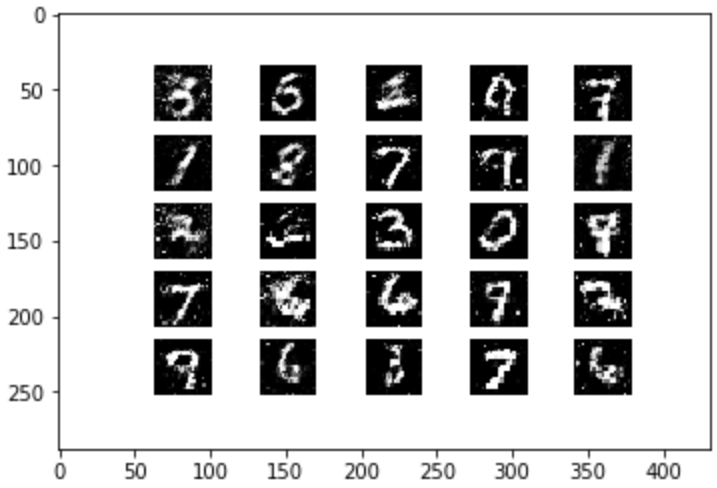
In [ ]:
```
a = imread('gan_images/1000.png')
plt.imshow(a)
```

Out[ ]: <matplotlib.image.AxesImage at 0x7f46a0484eb8>



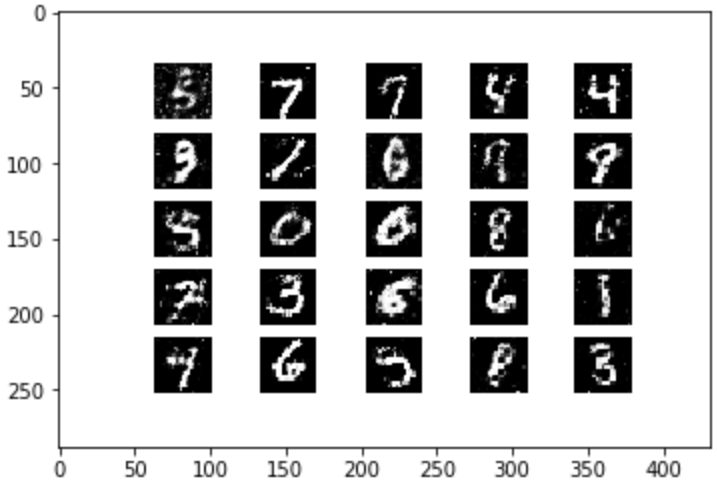
In [ ]:
```
a = imread('gan_images/5000.png')
plt.imshow(a)
```

Out[ ]: <matplotlib.image.AxesImage at 0x7f46a0467a58>



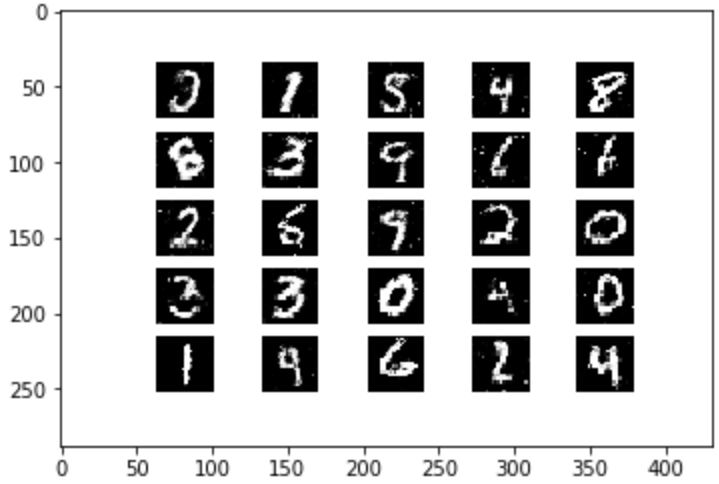
In [ ]:
```
a = imread('gan_images/10000.png')
plt.imshow(a)
```

Out[ ]: <matplotlib.image.AxesImage at 0x7f46a03cd588>



In [ ]:
```
a = imread('gan_images/20000.png')
plt.imshow(a)
```

Out[ ]: <matplotlib.image.AxesImage at 0x7f46a03720f0>



In [ ]:
```
a = imread('gan_images/29800.png')
plt.imshow(a)
```

Out[ ]: <matplotlib.image.AxesImage at 0x7f46a024ec18>