```
from numpy import expand_dims
from numpy import zeros
from numpy import ones
from numpy import vstack
from numpy.random import randn
from numpy.random import randint
from keras.datasets.mnist import load_data
from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Reshape
from keras.layers import Flatten
from keras.layers import Conv2D
from keras.layers import Conv2DTranspose
from keras.layers import LeakyReLU
from keras.layers import Dropout
from matplotlib import pyplot
```

```
# define the standalone discriminator model
def define_discriminator(in_shape=(28,28,1)):
  model = Sequential()
  model.add(Conv2D(64, (3,3), strides=(2, 2), padding='same', input_shape=in_shape))
  model.add(LeakyReLU(alpha=0.2))
  model.add(Dropout(0.4))
  model.add(Conv2D(64, (3,3), strides=(2, 2), padding='same'))
  model.add(LeakyReLU(alpha=0.2))
  model.add(Dropout(0.4))
  model.add(Flatten())
  model.add(Dense(1, activation='sigmoid'))
  opt = Adam(lr=0.0002, beta_1=0.5)
  model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])
  return model
```

```
# define the standalone generator model
def define_generator(latent_dim):
  model = Sequential()
```

```
  n_nodes = 128 * 7 * 7
  model.add(Dense(n_nodes, input_dim=latent_dim))
  model.add(LeakyReLU(alpha=0.2))
  model.add(Reshape((7, 7, 128)))
  model.add(Conv2DTranspose(128, (4,4), strides=(2,2), padding='same'))
  model.add(LeakyReLU(alpha=0.2))
  model.add(Conv2DTranspose(128, (4,4), strides=(2,2), padding='same'))
  model.add(LeakyReLU(alpha=0.2))
  model.add(Conv2D(1, (7,7), activation='sigmoid', padding='same'))
  return model
```

```
def define_gan(g_model, d_model):
  d_model.trainable = False
  model = Sequential()
  model.add(g_model)
  model.add(d_model)
  opt = Adam(lr=0.0002, beta_1=0.5)
  model.compile(loss='binary_crossentropy', optimizer=opt)
  return model
```

```
# load and prepare mnist training images
def load_real_samples():
  # load mnist dataset
  (trainX, _), (_, _) = load_data()
  X = expand_dims(trainX, axis=-1)
  X = X.astype('float32')
  X = X / 255.0
  return X
```

```
def generate_real_samples(dataset, n_samples):
  ix = randint(0, dataset.shape[0], n_samples)
  X = dataset[ix]
  y = ones((n_samples, 1))
  return X, y
```

```
def generate_latent_points(latent_dim, n_samples):
```

```
def generate_latent_points(latent_dim, n_samples):
  x_input = randn(latent_dim * n_samples)
  x_input = x_input.reshape(n_samples, latent_dim)
  return x_input
```

```
def generate_fake_samples(g_model, latent_dim, n_samples):
  x_input = generate_latent_points(latent_dim, n_samples)
  X = g_model.predict(x_input)
  y = zeros((n_samples, 1))
  return X, y
```

```
def save_plot(examples, epoch, n=10):
  for i in range(n * n):
    pyplot.subplot(n, n, 1 + i)
    pyplot.axis('off')
    pyplot.imshow(examples[i, :, :, 0], cmap='gray_r')
  filename = 'generated_plot_e%03d.png' % (epoch+1)
  pyplot.savefig(filename)
  pyplot.close()
```

```
def summarize_performance(epoch, g_model, d_model, dataset, latent_dim, n_samples=100):
  X_real, y_real = generate_real_samples(dataset, n_samples)
  _, acc_real = d_model.evaluate(X_real, y_real, verbose=0)
  x_fake, y_fake = generate_fake_samples(g_model, latent_dim, n_samples)
  _, acc_fake = d_model.evaluate(x_fake, y_fake, verbose=0)
  print('>Accuracy real: %.0f%%, fake: %.0f%%' % (acc_real*100, acc_fake*100))
  save_plot(x_fake, epoch)
  filename = 'generator_model_%03d.h5' % (epoch + 1)
  g_model.save(filename)
```

```
def train(g_model, d_model, gan_model, dataset, latent_dim, n_epochs=100, n_batch=256):
  bat_per_epo = int(dataset.shape[0] / n_batch)
  half_batch = int(n_batch / 2)
  for i in range(n_epochs):
    for j in range(bat_per_epo):
      X_real, y_real = generate_real_samples(dataset, half_batch)
      X_fake, y_fake = generate_fake_samples(g_model, latent_dim, half_batch)
```

```
      X_fake, y_fake = generate_fake_samples(g_model, latent_dim, half_batch)
      X, y = vstack((X_real, X_fake)), vstack((y_real, y_fake))
      d_loss, _ = d_model.train_on_batch(X, y)
      X_gan = generate_latent_points(latent_dim, n_batch)
      y_gan = ones((n_batch, 1))
      g_loss = gan_model.train_on_batch(X_gan, y_gan)
      print('>%d, %d/%d, d=%.3f, g=%.3f' % (i+1, j+1, bat_per_epo, d_loss, g_loss))
    if (i+1) % 10 == 0:
      summarize_performance(i, g_model, d_model, dataset, latent_dim)
```

```
latent_dim = 100
d_model = define_discriminator()
g_model = define_generator(latent_dim)
gan_model = define_gan(g_model, d_model)
dataset = load_real_samples()
train(g_model, d_model, gan_model, dataset, latent_dim)
```

```
>100, 117/234, d=0.697, g=0.683
>100, 118/234, d=0.691, g=0.696
>100, 119/234, d=0.695, g=0.714
>100, 120/234, d=0.689, g=0.718
>100, 121/234, d=0.693, g=0.707
>100, 122/234, d=0.698, g=0.699
>100, 123/234, d=0.691, g=0.700
>100, 124/234, d=0.689, g=0.701
>100, 125/234, d=0.691, g=0.694
>100, 126/234, d=0.688, g=0.700
>100, 127/234, d=0.694, g=0.692
>100, 128/234, d=0.688, g=0.682
>100, 129/234, d=0.692, g=0.694
>100, 130/234, d=0.692, g=0.694
>100, 131/234, d=0.690, g=0.695
>100, 132/234, d=0.693, g=0.688
>100, 133/234, d=0.694, g=0.696
>100, 134/234, d=0.692, g=0.708
>100, 135/234, d=0.696, g=0.705
>100, 136/234, d=0.691, g=0.700
>100, 137/234, d=0.694, g=0.697
>100, 138/234, d=0.692, g=0.696
>100, 139/234, d=0.692, g=0.683
```

```
>100, 140/234, d=0.691, g=0.689
>100, 141/234, d=0.691, g=0.708
>100, 142/234, d=0.694, g=0.704
>100, 143/234, d=0.688, g=0.702
>100, 144/234, d=0.688, g=0.686
>100, 145/234, d=0.692, g=0.687
>100, 146/234, d=0.693, g=0.698
>100, 147/234, d=0.687, g=0.707
>100, 148/234, d=0.695, g=0.703
>100, 149/234, d=0.699, g=0.684
>100, 150/234, d=0.694, g=0.692
>100, 151/234, d=0.691, g=0.705
>100, 152/234, d=0.685, g=0.702
>100, 153/234, d=0.692, g=0.693
>100, 154/234, d=0.690, g=0.702
>100, 155/234, d=0.696, g=0.697
>100, 156/234, d=0.692, g=0.697
>100, 157/234, d=0.690, g=0.704
>100, 158/234, d=0.689, g=0.705
>100, 159/234, d=0.687, g=0.705
>100, 160/234, d=0.691, g=0.698
>100, 161/234, d=0.694, g=0.705
>100, 162/234, d=0.693, g=0.694
>100, 163/234, d=0.695, g=0.691
>100, 164/234, d=0.688, g=0.684
>100, 165/234, d=0.693, g=0.696
>100, 166/234, d=0.692, g=0.700
>100, 167/234, d=0.689, g=0.702
>100, 168/234, d=0.690, g=0.702
>100, 169/234, d=0.693, g=0.704
>100, 170/234, d=0.695, g=0.709
>100, 171/234, d=0.692, g=0.689
>100, 172/234, d=0.688, g=0.691
>100, 173/234, d=0.698, g=0.693
>100, 174/234, d=0.693, g=0.691
>100, 175/234, d=0.691, g=0.683
```

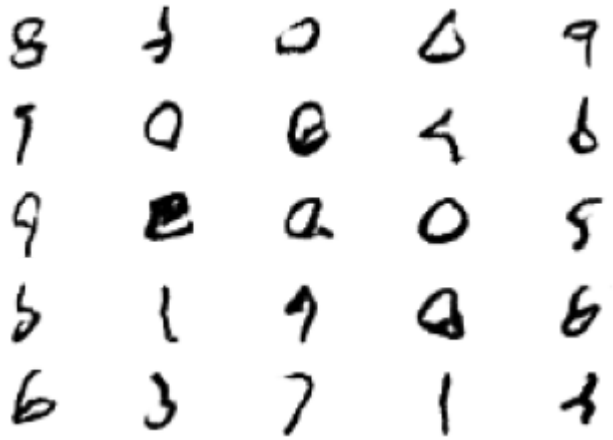
```
# example of loading the generator model and generating images
from keras.models import load_model
from numpy.random import randn
from matplotlib import pyplot
```

```
def generate_latent_points(latent_dim, n_samples):
  x_input = randn(latent_dim * n_samples)
  x_input = x_input.reshape(n_samples, latent_dim)
  return x_input


def save_plot(examples, n):
  for i in range(n * n):
    pyplot.subplot(n, n, 1 + i)
    pyplot.axis('off')
    pyplot.imshow(examples[i, :, :, 0], cmap='gray_r')
  pyplot.show()

model = load_model('generator_model_100.h5')
latent_points = generate_latent_points(100, 25)
X = model.predict(latent_points)
save_plot(X, 5)
```

```
WARNING:tensorflow:No training configuration found in the save file, so the model was *not* compiled. Compile it manua
```

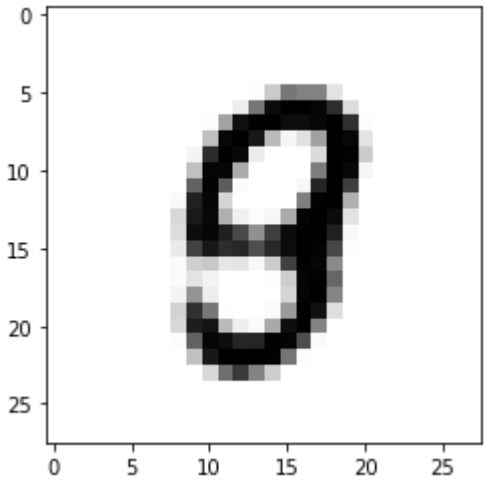


```
from keras.models import load_model
from numpy import asarray
from matplotlib import pyplot

model = load_model('generator_model_100.h5')
vector = asarray([[0.0 for   in range(100)]])
```

```
X = model.predict(vector)
pyplot.imshow(X[0, :, :, 0], cmap='gray_r')
pyplot.show()
```

```
WARNING:tensorflow:No training configuration found in the save file, so the model was *not* compiled. Compile it manua
```



✓ 0s completed at 10:07 PM