```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from keras.datasets import fashion_mnist
from keras.layers import Dense, Flatten, Reshape
from keras.layers import LeakyReLU
from keras.optimizers import Adam
from keras.models import Sequential
```

```
rows = 28
cols = 28
channels = 1
noise = 100
```

```
def generator(noise, rows, cols, channels):
    shape = (rows, cols, channels)
    model = Sequential()
    model.add(Dense(128, input_dim = noise))
    model.add(LeakyReLU(alpha = 0.01))
    model.add(Dense((rows*cols*channels), activation = 'tanh'))
    model.add(Reshape(shape))
    return model
```

```
def discriminator(rows, cols, channels):
    shape = (rows, cols, channels)
    model = Sequential()
    model.add(Flatten(input_shape = shape))
    model.add(Dense(128))
    model.add(LeakyReLU(alpha = 0.01))
    model.add(Dense(1, activation = 'sigmoid'))
    return model
```

```
def gan(generator, discriminator):
    model = Sequential()
    model.add(generator)
    model.add(discriminator)
    return model
```

```
generator = generator(noise, rows, cols, channels)
discriminator = discriminator(rows, cols, channels)
discriminator.compile(loss = 'binary_crossentropy',
                      optimizer = Adam(),
                      metrics = ['accuracy'])
discriminator.trainable = False
gan = gan(generator, discriminator)
gan.compile(loss = 'binary_crossentropy', optimizer = Adam())
```

```
check_points = []
loss = []
accuracy = []
```

```
def train(iterations, sample_interval, batch_size):

    (X_train, _), (_, _) = fashion_mnist.load_data()
    X_train = X_train / 127.5 - 1.0
    X_train = np.expand_dims(X_train, axis=3)

    true = np.ones((batch_size, 1))
    fake = np.zeros((batch_size, 1))

    for i in range(iterations):

        idt = np.random.randint(0, X_train.shape[0], batch_size)
        true_images = X_train[idt]

        idf = np.random.normal(0, 1, (batch_size, 100))
        fake_images = generator.predict(idf)
        disc_loss_true = discriminator.train_on_batch(true_images, true)
        disc_loss_fake = discriminator.train_on_batch(fake_images, fake)
        disc_loss, tot_accuracy = 0.5 * np.add(disc_loss_true, disc_loss_fake)

```

```
        idf = np.random.normal(0, 1, (batch_size, 100))
        fake_images = generator.predict(idf)

        gen_loss = gan.train_on_batch(idf, true)
        loss=[]
        if (i + 1) % sample_interval == 0:
            print(type(disc_loss))
            print(type(gen_loss))


            loss.append((disc_loss, gen_loss))

            accuracy.append(100.0 * tot_accuracy)
            check_points.append(i + 1)

            print("%d [Discriminator loss: %f, accuracy.: %.2f%%] [Generator loss: %f]" %
                  (i + 1, disc_loss, 100.0 * tot_accuracy, gen_loss))

            display_images(generator, noise)
```

```
def display_images(generator, noise, grid_rows=4, grid_cols=4):

    n = np.random.normal(0, 1, (grid_rows * grid_cols, noise))
    images = generator.predict(n)
    images = 0.5 * images + 0.5

    fig, axs = plt.subplots(grid_rows,
                            grid_cols,
                            figsize=(4, 4),
                            sharey=True,
                            sharex=True)

    count = 0
    for i in range(grid_rows):
        for j in range(grid_cols):
            axs[i, j].imshow(images[count, :, :, 0], cmap='gray')
            axs[i, j].axis('off')
            count += 1

```

```
iterations = 200
sample_interval = 100
batch_size = 128
train(iterations, sample_interval, batch_size)
```