DCGAN - Issue in understanding code

This a part of the code for a Deconvolutional-Convoltional Generativ

```
discriminator.trainable = False
ganInput = Input(shape=(100,))
# getting the output of the generator
# and then feeding it to the discriminator
# new model = D(G(input))
x = generator(ganInput)
ganOutput = discriminator(x)
gan = Model(input=ganInput, output=ganOutput)
gan.compile(loss='binary_crossentropy', optimizer=Adam())
```

Issue 1 - I do not understand what the line ganInput = Input(shape=(
function then what will ganInput contain ?

Issue 2 - What is the role of the `Model` API ? I read about in the kera

Please ask for any further clarification / details you need.

Keras with TensorFlow backend COMPLETE SOURCE CODE : http

python-3.x tensorflow keras conv-neural-network

The model will include all layers required in the computation of outpu
input or multi-output models, you can use lists as well:

```
model = Model(inputs=[ganInput1, ganInput2], outputs=[ganOutput1, ga
ganOutput3])
```

Which means to compute ganOutput1, ganOutput2, ganOutput3 the
ganInput1, ganInput2

This is necessary for backtracking so that way the Model api has wh

this line loads the mnist data : `(X_train, Y_train), (X_test, Y_test)`
and `Y_train` has training data and its corresponding target values ..
data and its corresponding target values

```
# ========================================================
# Here the data is being loaded
# X_train = training data, Y_train = training targets
# X_test = testing data , Y_test = testing targets
# ========================================================
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()

# ================================================
# Reshaping the training and testing data
# He has added one extra dimension which is always one
# ================================================
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1)
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)


X_train = X_train.astype('float32')

# ================================================
# Initially pixel values are in range of 0-255
# he makes the pixel values to be between -1 to 1
# 4] Convolution layer with activation LeakyRELU
# 5] Applying BatchNormalization
# 6] UpSampling2D layer
# 7] Convolution layer with activation LeakyRELU
# ====================================================================
generator = Sequential([
        Dense(128*7*7, input_dim=100, activation=LeakyReLU(0.2)),
        BatchNormalization(),
        Reshape((7,7,128)),
        UpSampling2D(),
        Convolution2D(64, 5, 5, border_mode='same', activation=Leaky
        BatchNormalization(),
        UpSampling2D(),
        Convolution2D(1, 5, 5, border_mode='same', activation='tanh'
    ])

generator.summary()

# ====================================================================
# He builds the discriminator model over here
# 1] Convolution layer which takes as input an image of shape (28, 2
# 2] Dropout layer
# 3] Convolution layer for down-sampling with LeakyReLU as activatio
# 4] Dropout layer
# 5] Flatten layer to flatten the output
# 6] 1 output node with sigmoid activation
# ========================================================       :===
discriminator = Sequential([
        Convolution2D(64, 5, 5, subsample=(2,2), input_shape=(28,28,
border_mode='same', activation=LeakyReLU(0.2)),
        Dropout(0.3),
        Convolution2D(128, 5, 5, subsample=(2,2), border_mode='same'
activation=LeakyReLU(0.2)),
        Dropout(0.3),
        Flatten(),
        Dense(1, activation='sigmoid')
    ])
```

```
# Input(shape=(100,)) returns a tensor of this shape (100,)
# ==================================================================
ganInput = Input(shape=(100,))
# getting the output of the generator
# and then feeding it to the discriminator
# new model = D(G(input))

# ===========================================================
# giving the input tensor of shape (100,) to generator model
# ===========================================================
x = generator(ganInput)

# ===========================================================
# the output of generator will be of shape (batch_size, 28, 28, 1)
# this output of generator will go to discriminator as input
# Remember we have defined discriminator input as shape (28, 28, 1)
# ===========================================================
ganOutput = discriminator(x)

# ==================================================================
# Now it is clear that generators output is needed as input to discr
# You have to tell this to Model api for backpropogation
# Your Model api is the whole model you have built
# it tells you that your model is a combination of generator and dis
where that data flow is from generator to discriminator
# YOUR_Model = generator -> discriminator
# This is something like you want to train generator and discriminat
model and not as two different models
# but at the same time they are actually being trained individually
sense)
# ==================================================================
gan = Model(input=ganInput, output=ganOutput)
gan.compile(loss='binary_crossentropy', optimizer=Adam())
gan.summary()

def train(epoch=10, batch_size=128):
    batch_count = X_train.shape[0] // batch_size
```

```
# these are the predicted images from the generator
predictions = generator.predict(noise_input, batch_size=

# the discriminator takes in the real images and the gen
X = np.concatenate([predictions, image_batch])

# labels for the discriminator
y_discriminator = [0]*batch_size + [1]*batch_size

# Let's train the discriminator
discriminator.trainable = True
discriminator.train_on_batch(X, y_discriminator)

# Let's train the generator
noise_input = np.random.rand(batch_size, 100)
y_generator = [1]*batch_size
discriminator.trainable = False
gan.train_on_batch(noise_input, y_generator)
```

edited Dec 26 '17 at 22:37

Could you please elaborate a bit more as to what `model` does taking into c
code and this use case of `model` ' – Tanmay Bhatnagar Dec 26 '17 at 15:

1 I have edited my answer ... Check out my explanation – Jai Dec 27 '17 at 4:

Just one more thing, when i call the `gan` model is the call going to the `gan`
call going to first the `discriminator` and then the `generator` . What I mea
the code namely `gan` , `discriminator` , `generator` or just 2 namely ` dis
`gan` is just a combination of them both. – Tanmay Bhatnagar Dec 27 '17

There are three models `gan` which is combination of `generator` and `dis`
`generator` and `discriminator` .. I think you should check out online how
a clear understanding – Jai Dec 27 '17 at 20:54