

## GANs

LATEST SUBMISSION GRADE

100%

1. In GANs, the network learns to improve on creating data by the way of knowledge flowing back from the *discriminator* to the *generator*.

1 / 1 point

- ☒ True  
☐ False

✓ Correct

Correct! The feedback sent from the discriminator helps the generator in better generation of the new data.

2. In the process of training a GAN, the *generator* is trained by getting it to produce a batch of fake images, and also labelling them as real images despite them being fake. *While this happens the evaluation performed by the discriminator helps in updating the parameters for the discriminator.*

1 / 1 point

- ☐ True  
☒ False

✓ Correct

Correct! The parameters of the *discriminator* are frozen during this step.

3. Consider the following piece of code for a generator, what is the purpose of using the *selu* activation function instead of ReLU?

1 / 1 point

```
generator = keras.models.Sequential([
    keras.layers.Dense(64, activation="selu",
        input_shape=[random_normal_dimensions]),
    keras.layers.Dense(128, activation="selu"),
    keras.layers.Dense(28 * 28, activation="sigmoid"),
    keras.layers.Reshape([28, 28])
])
```

- ☐ You want to remove the negative values which cancel out the positive values.  
☒ ReLU removes the noise within your data, but your intention is to keep it which is why selu is used.

✓ Correct

Correct!

4. Consider the following code for training the generator and check all that are true.

1 / 1 point

```
# Train the generator - PHASE 2
noise = tf.random.normal(shape=[batch_size, random_normal_dimensions])
generator_labels = tf.constant([[1.]]) * batch_size
discriminator.trainable = False
gan.train_on_batch(noise, generator_labels)
```

- ☒ You set the trainable parameters of the discriminator to *false* because updating the discriminator weights will corrupt the training process.

✓ Correct

Correct! You set them to false because the discriminator weights will get corrupted because of feeding it fake labels against both, *fake* and *original* images.

- ☐ You set the trainable parameters of the discriminator to *false* because updating the discriminator weights after every epoch is costly in the *phase 2* of the training.

- ☒ You set *all* of the *generator\_labels=1* while passing in both the real images and fake images in *phase 2* of the training.

✓ Correct

Correct! You pass both, *fake* and *original* images and set the label of all of them to 1 so you could try to trick the discriminator.

- ☐ You set *all* of the *generator\_labels=1* and pass in only the real images in *phase 2* of the training.

5. With regards to GANs, what does the term *mode collapse* mean?

1 / 1 point

- ☐ When the generator is no longer able to fool the discriminator with the generated data.  
☐ When the discriminator is no longer able to distinguish between real and fake data.  
☐ When the quality of the generated data stops to improve as the number of epochs increase.  
☒ When the model starts to generate more and more of the same data with which it was able to fool the discriminator.

✓ Correct

Correct!

6. Which of the following are some of the *best practices* when building GANs (**DCGANs**) which help us avoid the problem of *mode collapse*? Check all that apply.

1 / 1 point

- ☒ Batch normalization should be used in the generator except in the output layer.

✓ Correct

Correct!

- ☒ Avoid the use of *Dense* layer in both the discriminator and the generator.

✓ Correct

Correct!

- ☐ In the generator's architecture you should use pooling layers or Conv2D instead of Conv2DTranspose layers.

- ☐ All activation layers in the generator's architecture should be *selu* and in the discriminator's all activation layers should be *ReLU*.

7. You can apply a 3x3 stride filter of 1 on a 3x3 image using Conv2DTranspose (Process of deconvolution).

1 / 1 point

- ☒ True  
☐ False

✓ Correct

Correct! While it may not sound possible, Conv2DTranspose makes it possible by filling more data in the 3x3 image, making it a 9x9 image.

8. Following is the code of a *discriminator*. According to *best practices*, which activation function should be used ?

1 / 1 point

```
x = inputs = tf.keras.Input(shape=input_shape)
x = layers.Conv2D(64, 4, strides=2, padding='same')(x)
x = # your code here

x = layers.Conv2D(128, 4, strides=2, padding='same', use_bias=False)(x)
x = layers.BatchNormalization()(x)
x = # your code here

x = layers.Conv2D(256, 4, strides=2, padding='same', use_bias=False)(x)
x = layers.BatchNormalization()(x)
x = # your code here

x = layers.Conv2D(512, 4, strides=2, padding='same', use_bias=False)(x)
x = layers.BatchNormalization()(x)
x = # your code here

outputs = layers.Conv2D(1, 4, strides=1, padding='valid')(x)
```

- ☐ ReLU  
☐ tanh  
☐ selu  
☒ LeakyReLU

✓ Correct

Correct! You want to maintain some values when learning, instead of zeroing them out, which is what ReLU does.