Congratulations! You passed!

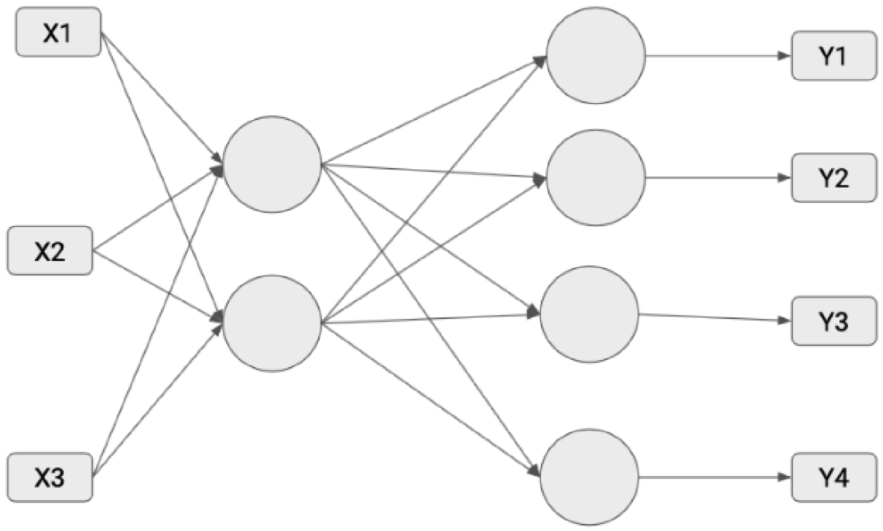
Grade received 95% **Latest Submission Grade** 95% **To pass** 80% or higher

Go to next item

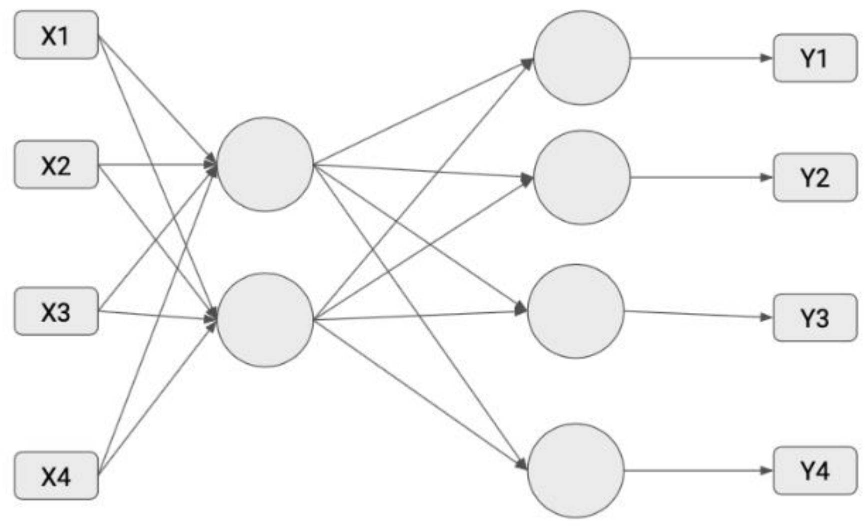
1. Which of the following is a valid architecture for an AutoEncoder? Check all that apply.

0.75 / 1 point

☐

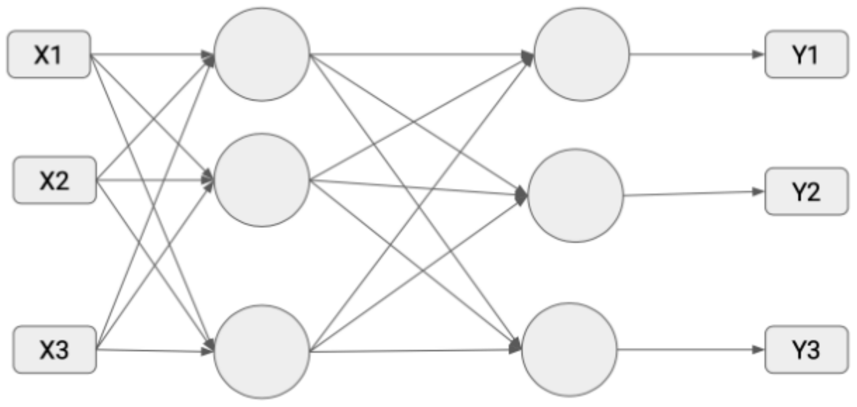


☑



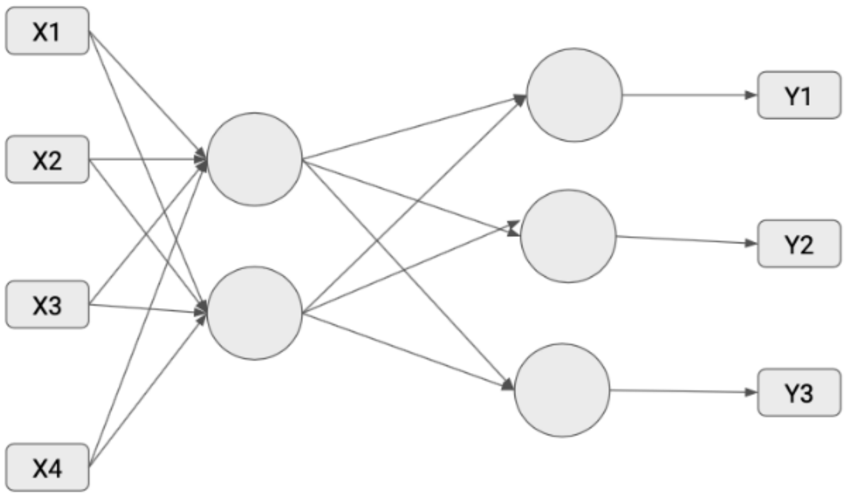
Correct
Correct!

☑



Correct
Correct! While the encoder layer and decoder layer have the same number of units which might take a straight pass through the layers, resulting in poor learning of latent representation, the architecture is still valid.

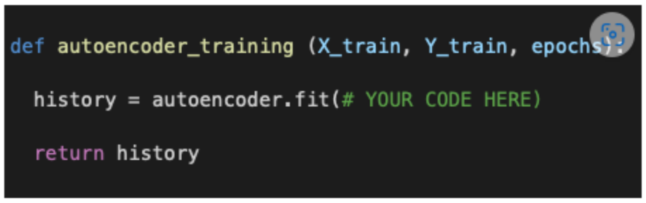
✓



This should not be selected
Incorrect! One of the constraints on an AutoEncoder is that it should have the same number of input and output units.

2. After initializing your AutoEncoder you are all set to train it. Which of the following pieces of code will you use ?

1 / 1 point



```
def autoencoder_training (X_train, Y_train, epochs):
  history = autoencoder.fit(# YOUR CODE HERE)
  return history
```

- ○ autoencoder.fit(X_train, Y_train, epochs=epochs)
- ○ autoencoder.fit(Y_train, Y_train, epochs=epochs)
- [✓ autoencoder.fit(X_train, X_train, epochs=epochs)]
- ○ autoencoder.fit(Y_train, X_train, epochs=epochs)

Correct
Correct! For data reconstruction purposes you fit input data values to input data values (as opposed to fitting them to output data values), this way the model learns best to replicate the data.

3. Consider the following code for a simple *AutoEncoder*, what is *model_1* outputting ?

1 / 1 point

```
inputs =  tf.keras.layers.Input(shape=(784,))

def simple_autoencoder():
    encoder = tf.keras.layers.Dense(units=32, activation='relu')(inputs)
    decoder = tf.keras.layers.Dense(units=784, activation='sigmoid')(encoder)
    return encoder, decoder

output_1, output_2 = simple_autoencoder()

model_1 = tf.keras.Model(inputs=inputs, outputs=output_1)

model_2 = tf.keras.Model(inputs=inputs, outputs=output_2)
```

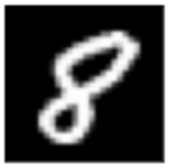
- ○ Displaying the label value which the model is trying to reconstruct.
- ○ Displaying the reconstruction of the original input which was fed to this architecture.
- [✓ Displaying the internal representation of the input the model is learning to replicate.]
- ○ Displaying the classification layer of the model, mapping input to the output label.

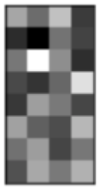
Correct
Correct! *model_1* is returning the encoded representation of your input values, which are being fed to the decoder as input.

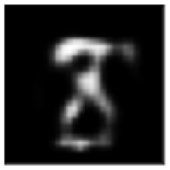
4. Consider the following code for a simple *AutoEncoder*, which of these is *model_1*'s output ?

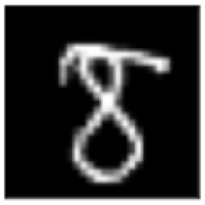
1 / 1 point

```python
inputs =  tf.keras.layers.Input(shape=(784,))

def simple_autoencoder():
    encoder = tf.keras.layers.Dense(units=32, activation='relu')(inputs)
    decoder = tf.keras.layers.Dense(units=784, activation='sigmoid')(encoder)
    return encoder, decoder

output_1, output_2 = simple_autoencoder()

model_1 = tf.keras.Model(inputs=inputs, outputs=output_1)

model_2 = tf.keras.Model(inputs=inputs, outputs=output_2)
```

○ 

● 

○ 

○ 

Correct

Correct!

5. Consider the following code for adding noise in an image. You use *tf.clip_by_value*to constrain the output image to values between 0 & 1.

1 / 1 point

```python
def map_image_with_noise(image, label):
  noise_factor = 0.5
  image = tf.cast(image, dtype=tf.float32)
  image = image / 255.0

  factor = noise_factor * tf.random.normal(shape=image.shape)
  image_noisy = image + factor
  image_noisy = tf.clip_by_value(image_noisy, 0.0, 1.0)

  return image_noisy, image
```

○ False

[✓ ● True]

Correct
Correct!