## Congratulations! You passed!

Grade received 100% To pass 80% or higher

Go to next item

# Week 1 Quiz

**Total points** 8

**1.** Which of the following are correct about the Extract, Transform, Load (ETL) procedure? **1 / 1 point**

- [ ] Load phase involves loading the a pre-trained model into the workspace
- [x] Transform phase involves data normalization and scaling

**Correct**
Correct!

**Transform** is the process of converting the extracted data from its previous form into the form it needs to be in so that it can be used in our case for training.

- [x] Extract phase involves downloading a zip file from any external source containing the data

**Correct**
Correct!

**Extract** in general is the process of reading data from multiple sources/ a database

- [ ] Extract phase would involve splitting the data into training and test sets

**2.** What does the following code block achieve? **1 / 1 point**

```
tfds.load(name="mnist", split="train")
```

- ( ) Loads mnist labels and assign them to any training dataset
- ( ) Extracts the mnist training dataset from a zip file
- ( ) Splits the downloaded mnist data into train and test sets
- (•) Downloads and extracts training records from the mnist dataset

**Correct**
Correct!

It downloads the dataset (if not already stored in your local TensorFlow directory) and ***split="train"*** parameter tells it to return only training records

**3.** Can you explore 10 records from the dataset by loading them into an iterator like this? **1 / 1 point**

```
iterator = dataset.take(10)
```

- (•) Yes
- ( ) No

**Correct**
Correct!

take() method allows you to select the n examples from the dataset, where n is a passed as a parameter.

**4.** What is the role of the tfds.list_builders() function? **1 / 1 point**

- (•) To return string names of all available datasets in Tensorflow

- ○ To build a list of multiple datasets to load at a time
- ○ To return a list of files in the dataset
- ○ To create an empty dataset for creating a custom dataset

**Correct**
Correct!

tfds.list_builders() returns the string names of all tfds.core.DatasetBuilder which is the baseclass defined to handle all datasets present in Tensorflow APIs.

**5.** How would you inspect the metadata and the details of a TensorFlow dataset? **1 / 1 point**

- ○ Load the data using tfds.load() with the parameter with_info=true, and then inspect the showCoreData property.
- ○ There's no API for this, read the docs instead
- ○ Load the data using tfds.load() with the parameter as_supervised=False , and then inspect the DataSetInfo property
- ◉ Load the data using tfds.load() with the parameter with_info=true, and then inspect the DataSetInfo property

**Correct**
Correct!

DatasetInfo documents datasets, including its name, version, and features. ***with_info=true*** is the parameter to pass in tfds.load() to get the metadata.

**6.** Which of the following ways are used to load mnist dataset with major version 1, minor version 2 and any patch version ? **1 / 1 point**

- ☑ Specify the exact version for a patch version in the load parameter like this: ***tfds.load("mnist:1.2.1")***

**Correct**
This also works as it loads specifically the version 1.2.1. If it is not pre-cached, it gets downloaded, extracted and loaded even if there are different versions like 1.2.2 present in your TensorFlow installation folder.

- ☑ Specify the desired version with asterisk in patch version in the string in the load parameter like this:

***tfds.load("mnist").version("1.2.*")***

**Correct**
The asterisk helps in identifying any dataset with 1.2.x as a version meaning any patch version is identified

- ☐ You'll need to install the matching version of TFDS that installs that dataset, and then load it
- ☐ Specify the desired version as a split, like this:

***tfds.load(name="mnist", split="1.*.*")***

**7.** The fashion MNIST is a relatively simple example of a dataste used in computer vision modelling tasks used with or without TFDS. If you load the data using TensorFlow Keras datasets in TensorFlow 2.0 and above, what would the code look like? **1 / 1 point**

- ○

```
data = tf.keras.as_numpy(fashion_mnist)
(training_images,training_labels) , (test_images,test_labels) = data.load_data()
```

- ○

```
data = keras.dataset.fashion_mnist
(training_images,training_labels) , (test_images,test_labels) = data.load_data()
```

```
data = tfds.as_numpy(tfds.load('fashion_mnist',
split =['train','test'],
batch_size=-1,
as_supervised=True))

(training_images,training_labels) , (test_images,test_labels) = data
```

```
data = tf.keras.dataset.fashion_mnist
(training_images,training_labels),(test_images,test_labels) = data.load_data()
```

**Correct**
Correct!

The new Keras API integrated as part of TensorFlow in 2.0+ version makes it a seamless integration to access Dataset and other classes.

**8.** Which of the following code blocks would successfully create "Horses and Humans" test batches of 10 by shuffling 100 data samples? **1 / 1 point**

```
data = tfds.load('horses_or_humans',split = 'train', as_supervised=True)

batches = data.shuffle(100).batch(10)
```

```
data = tfds.load('horses_or_humans',split = 'test', as_supervised=True)

batches = data.shuffle(100).batch(10)
```

```
data = tfds.load('horses_or_humans',split = 'test', as_supervised=True)

batches = data.shuffle(batch(100),10)
```

```
data = tfds.load('horses_or_humans',split = 'test', as_supervised=False)

batches = data.shuffle(100).batch(10)
```

**Correct**
Correct!

You specify the split as "test" to fetch the test records and mention as_supervised="True" so that the returned tf.data.Dataset will have a 2-tuple structure (input, label) according to builder.info.supervised_keys. If False, the default, the returned tf.data.Dataset will have a dictionary with all the features and you will get an error when you call .shuffle() on it.