Dataset

Learn how datasets are represented in TensorFlow's input pipeline.

Chapter Goals:

- Learn how to create a dataset in TensorFlow
- Implement a function that creates a dataset from NumPy data

A. Input pipeline

In TensorFlow, the input pipeline for executing a machine learning model is represented by the `Dataset` class (which we'll refer to as simply a dataset). A dataset can be created for a variety of input values, from NumPy arrays to protocol buffers. The most basic way to create a dataset is with the `tf.data.Dataset.from_tensor_slices` function.

```
import numpy as np
import tensorflow as tf

data = np.array([[  1. ,   2.1],
       [  2. ,   3. ],
       [  8.1, -10. ]])

d1 = tf.data.Dataset.from_tensor_slices(data)

print(d1)
```

Using from_tensor_slices to create a dataset from a NumPy array.

In the example, `d1` is a dataset containing the data from `data`. The dataset consists of three observations, with each observation being a row in `data`. Since each row of `data` has two columns, the observations in `d1` have shape `(2,)`.

We can also create datasets from tuple inputs. This is useful when we want to create a dataset from both feature data and labels for each data observation.

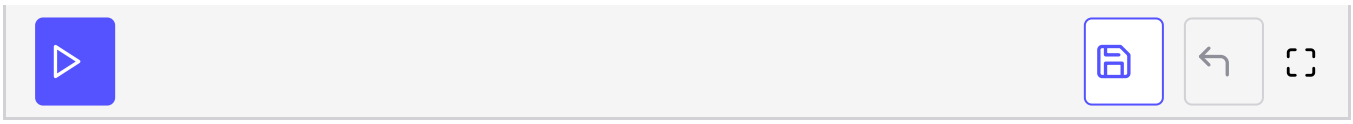
```
import numpy as np
import tensorflow as tf

data = np.array([[1. , 2. , 3. ],
       [1.1, 0. , 8. ]])

labels = np.array([1, 0])

d2 = tf.data.Dataset.from_tensor_slices((data, labels))

print(d2)
```



Using from_tensor_slices to create a dataset from a tuple of NumPy arrays.

In the example, `d2` is a dataset containing the data from `data` and the observation labels from `labels`. There are two total observations, and each observation has shape `(3,)`, since `data` has three columns.

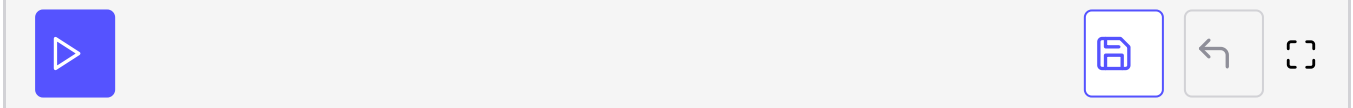
B. Image file dataset

The `from_tensor_slices` function is not limited to just taking NumPy arrays as input. For example, we can use it to create a dataset of file names. A popular application of this is creating a dataset for image files.

```
import numpy as np
import tensorflow as tf

filenames = ['img1.jpg', 'img2.jpg']
img_d1 = tf.data.Dataset.from_tensor_slices(filenames)
print(img_d1)

labels = np.array([1, 0])
img_d2 = tf.data.Dataset.from_tensor_slices((filenames, labels))
print(img_d2)
```



Using from_tensor_slices to create a dataset for image file names.

In the example, `img_d1` represents a dataset for the input file names, while `img_d2` also has a label for each image file. Note that each dataset observation is a filename, rather than the actual file contents. For more information on

processing image files to retrieve the byte data, see the **Image Recognition** course on Educative.

C. Specialized datasets

Apart from the `from_tensor_slices` function, we can also use `TFRecordDataset` and `TextLineDataset` to create specialized datasets for protocol buffers and text data, respectively.

```
import numpy as np
import tensorflow as tf

records_files = ['one.tfrecords', 'two.tfrecords']
d1 = tf.data.TFRecordDataset(records_files)
print(d1)

txt_files = ['lines.txt']
d2 = tf.data.TextLineDataset(txt_files)
print(d2)
```

Using TFRecordDataset and TextLineDataset to create specialized datasets.

The `TFRecordDataset` takes in a list of TFRecords files and creates a dataset where each observation is an individual serialized protocol buffer. In the example, `d1` contains the serialized protocol buffers from `'one.tfrecords'` and `'two.tfrecords'`.

The `TextLineDataset` takes in a list of text files and creates a dataset where each observation is a separate line from the text files. In the example, `d2` contains the lines from `'lines.txt'`.