Building TensorFlow dataset in a tensor

In [28]:
```
a = [1.2, 3.4, 7.5, 4.1, 5.0, 1.0]

ds = tf.data.Dataset.from_tensor_slices(a)

print(ds)
```

```
<TensorSliceDataset shapes: (), types: tf.float32>
```

In [29]:
```
for item in ds:
    print(item)
```

```
tf.Tensor(1.2, shape=(), dtype=float32)
tf.Tensor(3.4, shape=(), dtype=float32)
tf.Tensor(7.5, shape=(), dtype=float32)
tf.Tensor(4.1, shape=(), dtype=float32)
tf.Tensor(5.0, shape=(), dtype=float32)
tf.Tensor(1.0, shape=(), dtype=float32)
```

In [30]:
```
ds_batch = ds.batch(3)

for i, elem in enumerate(ds_batch, 100):
    print('batch {}:'.format(i), elem.numpy())
```

```
batch 100: [1.2 3.4 7.5]
batch 101: [4.1 5.  1. ]
```

Connecting two tensors in one dataset (1/2)

Approach 1

In [31]:
```
tf.random.set_seed(1)

t_x = tf.random.uniform([4, 3], dtype=tf.float32)
t_y = tf.range(4)
```

In [32]:
```
ds_x = tf.data.Dataset.from_tensor_slices(t_x)
ds_y = tf.data.Dataset.from_tensor_slices(t_y)

ds_joint = tf.data.Dataset.zip((ds_x, ds_y))

for example in ds_joint:
    print('  x: ', example[0].numpy(),
          '  y: ', example[1].numpy())
```

```
  x:  [0.165 0.901 0.631]   y:  0
  x:  [0.435 0.292 0.643]   y:  1
  x:  [0.976 0.435 0.66 ]   y:  2
  x:  [0.605 0.637 0.614]   y:  3
```

Connecting two tensors in one dataset (2/2)

Approach 2

```
In [33]: ds_joint = tf.data.Dataset.from_tensor_slices((t_x, t_y))

         for example in ds_joint:
             print('  x: ', example[0].numpy(),
                   '  y: ', example[1].numpy())

           x:  [0.165 0.901 0.631]   y:  0
           x:  [0.435 0.292 0.643]   y:  1
           x:  [0.976 0.435 0.66 ]   y:  2
           x:  [0.605 0.637 0.614]   y:  3

In [34]: ds_trans = ds_joint.map(lambda x, y: (x*2-1.0, y))

         for example in ds_trans:
             print('  x: ', example[0].numpy(),
                   '  y: ', example[1].numpy())

           x:  [-0.67   0.803  0.262]   y:  0
           x:  [-0.131 -0.416  0.285]   y:  1
           x:  [ 0.952 -0.13   0.32 ]   y:  2
           x:  [0.21  0.273 0.229]   y:  3
```

shuffle(), batch(), repeat() method (1/6)

In [35]:
```
tf.random.set_seed(1)
ds = ds_joint.shuffle(buffer_size=len(t_x))

for example in ds:
    print('  x: ', example[0].numpy(),
          '  y: ', example[1].numpy())
```

```
  x:  [0.976 0.435 0.66 ]   y:  2
  x:  [0.435 0.292 0.643]   y:  1
  x:  [0.165 0.901 0.631]   y:  0
  x:  [0.605 0.637 0.614]   y:  3
```

In [36]:
```
ds = ds_joint.batch(batch_size=3,
                    drop_remainder=False)

batch_x, batch_y = next(iter(ds))

print('batch x: \n', batch_x.numpy())

print('batch y:   ', batch_y.numpy())
```

```
batch x:
 [[0.165 0.901 0.631]
 [0.435 0.292 0.643]
 [0.976 0.435 0.66 ]]
batch y:    [0 1 2]
```

shuffle(), batch(), repeat() method (2/6)

In [37]:
```
ds = ds_joint.batch(3).repeat(count=2)

for i,(batch_x, batch_y) in enumerate(ds):
    print(i, batch_x.shape, batch_y.numpy())
```

```
0 (3, 3) [0 1 2]
1 (1, 3) [3]
2 (3, 3) [0 1 2]
3 (1, 3) [3]
```

In [38]:
```
ds = ds_joint.repeat(count=2).batch(3)

for i,(batch_x, batch_y) in enumerate(ds):
    print(i, batch_x.shape, batch_y.numpy())
```

```
0 (3, 3) [0 1 2]
1 (3, 3) [3 0 1]
2 (2, 3) [2 3]
```

shuffle(), batch(), repeat() method (3/6)

In [39]:
```
tf.random.set_seed(1)

## Step 1 : shuffle -> batch -> repeat
ds = ds_joint.shuffle(4).batch(2).repeat(3)

for i,(batch_x, batch_y) in enumerate(ds):
    print(i, batch_x.shape, batch_y.numpy())
```

```
0 (2, 3) [2 1]
1 (2, 3) [0 3]
2 (2, 3) [0 3]
3 (2, 3) [1 2]
4 (2, 3) [3 0]
5 (2, 3) [1 2]
```

shuffle(), batch(), repeat() method (4/6)

```
In [40]: tf.random.set_seed(1)

         ## Step 1': shuffle -> batch -> repeat
         ds = ds_joint.shuffle(4).batch(2).repeat(20)

         for i,(batch_x, batch_y) in enumerate(ds):
             print(i, batch_x.shape, batch_y.numpy())
```

```
0 (2, 3) [2 1]
1 (2, 3) [0 3]
2 (2, 3) [0 3]
3 (2, 3) [1 2]
4 (2, 3) [3 0]
5 (2, 3) [1 2]
6 (2, 3) [1 3]
7 (2, 3) [2 0]
8 (2, 3) [1 2]
9 (2, 3) [3 0]
10 (2, 3) [3 0]
11 (2, 3) [2 1]
12 (2, 3) [3 0]
13 (2, 3) [1 2]
14 (2, 3) [3 0]
15 (2, 3) [2 1]
16 (2, 3) [2 3]
```

shuffle(), batch(), repeat() method (4/6)

In [41]:
```
tf.random.set_seed(1)

## Step 2: batch -> shuffle -> repeat
ds = ds_joint.batch(2).shuffle(4).repeat(3)

for i,(batch_x, batch_y) in enumerate(ds):
    print(i, batch_x.shape, batch_y.numpy())
```

```
0 (2, 3) [0 1]
1 (2, 3) [2 3]
2 (2, 3) [0 1]
3 (2, 3) [2 3]
4 (2, 3) [2 3]
5 (2, 3) [0 1]
```

shuffle(), batch(), repeat() method (5/6)

```
In [42]: tf.random.set_seed(1)

         ## Step 2: batch -> shuffle -> repeat
         ds = ds_joint.batch(2).shuffle(4).repeat(20)

         for i,(batch_x, batch_y) in enumerate(ds):
             print(i, batch_x.shape, batch_y.numpy())
```

```
0 (2, 3) [0 1]
1 (2, 3) [2 3]
2 (2, 3) [0 1]
3 (2, 3) [2 3]
4 (2, 3) [2 3]
5 (2, 3) [0 1]
6 (2, 3) [2 3]
7 (2, 3) [0 1]
8 (2, 3) [2 3]
9 (2, 3) [0 1]
10 (2, 3) [2 3]
11 (2, 3) [0 1]
12 (2, 3) [2 3]
13 (2, 3) [0 1]
14 (2, 3) [2 3]
15 (2, 3) [0 1]
16 (2, 3) [0 1]
```

shuffle(), batch(), repeat() method (6/6)

In [43]:
```
tf.random.set_seed(1)

## Step 3 : batch -> repeat -> shuffle
ds = ds_joint.batch(2).repeat(3).shuffle(4)

for i,(batch_x, batch_y) in enumerate(ds):
    print(i, batch_x.shape, batch_y.numpy())
```

```
0 (2, 3) [0 1]
1 (2, 3) [0 1]
2 (2, 3) [2 3]
3 (2, 3) [2 3]
4 (2, 3) [0 1]
5 (2, 3) [2 3]
```

Building dataset from a file stored in a local drive (1/6)

In [44]:
```
import pathlib

imgdir_path = pathlib.Path('cat_dog_images')

file_list = sorted([str(path) for path in imgdir_path.glob('*.jpg')])

print(file_list)
```

```
['cat_dog_images\\cat-01.jpg', 'cat_dog_images\\cat-02.jpg', 'cat_dog_images\\cat-03.jpg', 'cat_dog_images\\dog-01.jpg',
'cat_dog_images\\dog-02.jpg', 'cat_dog_images\\dog-03.jpg']
```

Building dataset from a file stored in a local drive (2/6)

In [45]:
```
import matplotlib.pyplot as plt
import os


fig = plt.figure(figsize=(10, 5))
for i,file in enumerate(file_list):
    img_raw = tf.io.read_file(file)
    img = tf.image.decode_image(img_raw)
    print( 'Image Size:  ', img.shape)
    ax = fig.add_subplot(2, 3, i+1)
    ax.set_xticks([]); ax.set_yticks([])
    ax.imshow(img)
    ax.set_title(os.path.basename(file), size=15)

# plt.savefig('images/13_1.png', dpi=300)
plt.tight_layout()
plt.show()
```

Building dataset from a file stored in a local drive (3/6)

```
Image size:  (900, 1200, 3)
Image size:  (900, 1200, 3)
Image size:  (900, 742, 3)
Image size:  (800, 1200, 3)
Image size:  (800, 1200, 3)
Image size:  (900, 1200, 3)
```

cat-01.jpg

cat-02.jpg

cat-03.jpg

dog-01.jpg

dog-02.jpg

dog-03.jpg

Building dataset from a file stored in a local drive (4/6)

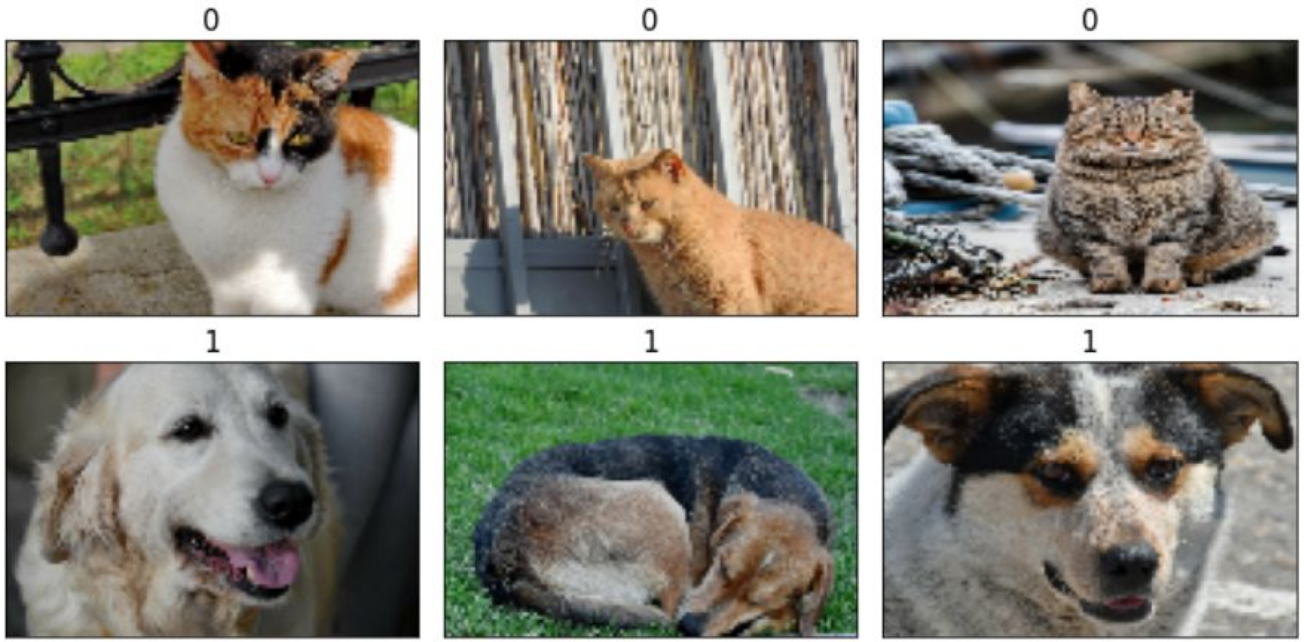
```
In [46]: labels = [1 if 'dog' in os.path.basename(file) else 0
                   for file in file_list]
         print(labels)

         [0, 0, 0, 1, 1, 1]

In [47]: ds_files_labels = tf.data.Dataset.from_tensor_slices(
             (file_list, labels))

         for item in ds_files_labels:
             print(item[0].numpy(), item[1].numpy())

         b'cat_dog_images\\cat-01.jpg' 0
         b'cat_dog_images\\cat-02.jpg' 0
         b'cat_dog_images\\cat-03.jpg' 0
         b'cat_dog_images\\dog-01.jpg' 1
         b'cat_dog_images\\dog-02.jpg' 1
         b'cat_dog_images\\dog-03.jpg' 1
```

Building dataset from a file stored in a local drive (5/6)

```
In [48]: def load_and_preprocess(path, label):
             image = tf.io.read_file(path)
             image = tf.image.decode_jpeg(image, channels=3)
             image = tf.image.resize(image, [img_height, img_width])
             image /= 255.0

             return image, label

         img_width, img_height = 120, 80

         ds_images_labels = ds_files_labels.map(load_and_preprocess)

         fig = plt.figure(figsize=(10, 5))
         for i,example in enumerate(ds_images_labels):
             print(example[0].shape, example[1].numpy())
             ax = fig.add_subplot(2, 3, i+1)
             ax.set_xticks([]); ax.set_yticks([])
             ax.imshow(example[0])
             ax.set_title('{}'.format(example[1].numpy()),
                          size=15)

         plt.tight_layout()

         plt.show()
```

Building dataset from a file stored in a local drive (6/6)

```
(80, 120, 3) 0
(80, 120, 3) 0
(80, 120, 3) 0
(80, 120, 3) 1
(80, 120, 3) 1
(80, 120, 3) 1
```



Loading datasets from tensorflow.datasets library (1/10)

```
In [49]: !pip install --upgrade tensorflow-datasets

Collecting tensorflow-datasets
  Downloading tensorflow_datasets-4.4.0-py3-none-any.whl (4.0 MB)
Requirement already satisfied: tqdm in c:\users\emcast\anaconda3\lib\site-packages (from tensorflow-datasets) (4.59.0)
Collecting dill
  Downloading dill-0.3.4-py2.py3-none-any.whl (86 kB)
Requirement already satisfied: six in c:\users\emcast\anaconda3\lib\site-packages (from tensorflow-datasets) (1.15.0)
Requirement already satisfied: absl-py in c:\users\emcast\anaconda3\lib\site-packages (from tensorflow-datasets) (0.13.0)
Collecting promise
  Downloading promise-2.3.tar.gz (19 kB)
```

Loading datasets from tensorflow.datasets library (2/10)

```
In [50]: import tensorflow_datasets as tfds

         print(len(tfds.list_builders()))
         print(tfds.list_builders()[:5])

         278
         ['abstract_reasoning', 'accentdb', 'aeslc', 'aflw2k3d', 'ag_news_subset']
```

Loading datasets from tensorflow.datasets library (3/10)

- The following command generates the entire list.

```
In [51]: tfds.list_builders()

Out[51]: ['abstract_reasoning',
          'accentdb',
          'aeslc',
          'aflw2k3d',
          'ag_news_subset',
          'ai2_arc',
          'ai2_arc_with_ir',
          'amazon_us_reviews',
          'anli',
          'arc',
          'bair_robot_pushing_small',
          'bccd',
          'beans',
          'big_patent',
          'bigearthnet',
          'billsum',
          'binarized_mnist',
          'binary_alpha_digits',
          'blimp',
          'bool_q',
          'c4',
          'caltech101',
          'caltech_birds2010',
          'caltech_birds2011',
          'cars196',
          'cassava',
```

Loading datasets from tensorflow.datasets library (4/10)

- Downloading CelebA dataset.

```
In [52]: celeba_bldr = tfds.builder('celeb_a')

         print(celeba_bldr.info.features)
         print('\n', 30*"=", '\n')
         print(celeba_bldr.info.features.keys())
         print('\n', 30*"=", '\n')
         print(celeba_bldr.info.features['image'])
         print('\n', 30*"=", '\n')
         print(celeba_bldr.info.features['attributes'].keys())
         print('\n', 30*"=", '\n')
         print(celeba_bldr.info.citation)
```

```
FeaturesDict({
    'attributes': FeaturesDict({
        '5_o_Clock_Shadow': tf.bool,
        'Arched_Eyebrows': tf.bool,
        'Attractive': tf.bool,
        'Bags_Under_Eyes': tf.bool,
        'Bald': tf.bool,
        'Bangs': tf.bool,
        'Big_Lips': tf.bool,
        'Big_Nose': tf.bool,
        'Black_Hair': tf.bool,
        'Blond_Hair': tf.bool,
```

Loading datasets from tensorflow.datasets library (5/10)

- Download and save data in the local drive.

```
In [53]: celeba_bldr.download_and_prepare()

Downloading and preparing dataset Unknown size (download: Unknown size, generated: Unknown size, total: Unknown size) to
C:\Users\emcast\tensorflow_datasets\celeb_a\2.0.1...

Dl Completed...: 100% 4/4 [08:34<00:00, 197.62s/ url]

Dl Size...: 1414/0 [08:34<00:00, 2.65 MiB/s]

Dataset celeb_a downloaded and prepared to C:\Users\emcast\tensorflow_datasets\celeb_a\2.0.1. Subsequent calls will reuse
this data.
```

Loading datasets from tensorflow.datasets library (6/10)

- Load data from the drive by tf.data.Datasets

```
In [54]: datasets = celeba_bldr.as_dataset(shuffle_files=False)

         datasets.keys()

Out[54]: dict_keys([Split('train'), Split('validation'), Split('test')])
```

Loading datasets from tensorflow.datasets library (7/10)

In [55]:
```
ds_train = datasets['train']
assert isinstance(ds_train, tf.data.Dataset)

example = next(iter(ds_train))
print(type(example))
print(example.keys())
```

```
<class 'dict'>
dict_keys(['attributes', 'image', 'landmarks'])
```

Loading datasets from tensorflow.datasets library (8/10)

```
In [56]: ds_train = ds_train.map(lambda item:
             (item['image'], tf.cast(item['attributes']['Male'], tf.int32)))
```

```
In [57]: ds_train = ds_train.batch(18)
         images, labels = next(iter(ds_train))

         print(images.shape, labels)

         (18, 218, 178, 3) tf.Tensor([0 1 0 0 1 1 1 1 1 0 0 0 1 0 0 1 1 1], shape=(18,), dtype=int32)
```

Loading datasets from tensorflow.datasets library (9/10)

In [58]:
```
fig = plt.figure(figsize=(12, 8))
for i,(image,label) in enumerate(zip(images, labels)):
    ax = fig.add_subplot(3, 6, i+1)
    ax.set_xticks([]); ax.set_yticks([])
    ax.imshow(image)
    ax.set_title('{}'.format(label), size=15)

# plt.savefig('images/13_3.png', dpi=300)
plt.show()
```

Loading datasets from tensorflow.datasets library (10/10)

Another approach to loading datasets (1/3)

```
In [59]: mnist, mnist_info = tfds.load('mnist', with_info=True,
                                       shuffle_files=False)

         print(mnist_info)

         print(mnist.keys())
```

```
Downloading and preparing dataset Unknown size (download: Unknown size, generated: Unknown size, total: Unknown size) to
C:\Users\emcast\tensorflow_datasets\mnist\3.0.1...

Dl Completed...: 100% 4/4 [00:09<00:00, 2.57s/ url]

Dl Size...: 100% 10/10 [00:09<00:00, 1.47 MiB/s]

Extraction completed...: 100% 4/4 [00:09<00:00, 2.77s/ file]
```

Another approach to loading datasets (2/3)

```
In [60]: ds_train = mnist['train']

         assert isinstance(ds_train, tf.data.Dataset)

         ds_train = ds_train.map(lambda item:
              (item['image'], item['label']))

         ds_train = ds_train.batch(10)
         batch = next(iter(ds_train))
         print(batch[0].shape, batch[1])

         fig = plt.figure(figsize=(15, 6))
         for i,(image,label) in enumerate(zip(batch[0], batch[1])):
             ax = fig.add_subplot(2, 5, i+1)
             ax.set_xticks([]); ax.set_yticks([])
             ax.imshow(image[:, :, 0], cmap='gray_r')
             ax.set_title('{}'.format(label), size=15)


         plt.show()
```

Another approach to loading datasets (3/3)

```
(10, 28, 28, 1) tf.Tensor([4 1 0 7 8 1 2 7 1 6], shape=(10,), dtype=int64)
```