# Execution

Learn how to execute data extraction in TensorFlow.

Chapter Goals:

- Iterate through a dataset and extract pixel data using `tf.Session`

## A. Data iteration

In the previous chapter we set up a next-element tensor using the `get_next` function. The way we actually execute the data extraction is by using the `run` function of `tf.Session`.

Each time we use `sess.run(next_image)`, we are iterating a single step through our dataset. So the first time we use `sess.run(next_image)`, it'll return the first pixel array in `dataset` (as a NumPy array), and the $i^{th}$ time it'll return the $i^{th}$ pixel array in `dataset`.

```python
it = dataset.make_one_shot_ite
next_image = it.get_next()
sess = tf.Session()
for i in range(3):
    sess.run(next_image)
```

Iteration through a dataset. Each time sess.run(next_image) is called, it returns the dataset value at the current iteration.

We can only call `sess.run(next_image)` for the number of images there are in the dataset before we get an `OutOfRangeError`.

## B. Using data

Rather than going through the decoding process every time we use a dataset of image files, it is usually better to just decode once and save the NumPy pixel data in a file. This can be done with the `numpy.save` function.

The code below shows examples of using `numpy.save`. The file itself is not human-readable, but can be loaded back into a NumPy array with `numpy.load`

```
import numpy as np

arr = np.array([1, 2, 3])
# Saves to 'arr.npy'
np.save('arr.npy', arr)
# Also saves to 'arr.npy'
np.save('arr', arr)
# Loading from that file
arr_copy = np.load('arr.npy')
print(repr(arr_copy))
```

There are other more efficient ways to store the decoded data. However, for smaller projects that don't use too large of an image dataset, the way of saving NumPy data in a file works fine.

## Time to Code!

In this chapter we'll be completing the `get_image_data` function from the previous chapter.

Since we're going to return a list of all our image pixel data, we want to first initialize this list.

**Set `image_data_list` equal to an empty list.**

We'll put our iteration and execution code within the scope of a `tf.Session`. For more information on `tf.Session` scopes, see the Machine Learning for Software Engineers course on Educative.

**Create a `with tf.Session() as sess` code block.**

**Inside the scope of `sess`, create a `for` loop that iterates `i` through `range(len(image_paths))`.**

Inside the `for` loop, we'll execute a single data extraction and append the extracted pixel array to the list.

**Set `image_data` equal to `sess.run(next_image)`.**

**Append the resultant `image_data` to `image_data_list`.**

After iterating through the entire dataset, we can return `image_data_list`.

### Outside the scope of `sess`, return `image_data_list`.

```
import tensorflow as tf

# Get the decoded image data from the input image file paths
def get_image_data(image_paths, image_type=None, resize_shape=None, channels=0):
    dataset = get_dataset(image_paths, image_type, resize_shape, channels)
    iterator = dataset.make_one_shot_iterator()
    next_image = iterator.get_next()
    # CODE HERE
```