Image Files

Discover how image data is stored in files and how pixels affect image interpretation.

Chapter Goals:

- Learn about image data and pixels
- Read byte data from an image file

A. Image data

Before we do any image processing, we need to understand how image files work. Specifically, we'll discuss how these files use byte data and pixels to represent images.

If you've ever looked at an image file's properties before, it'll show the dimensions of the image, i.e. the height and width of the image. The height and width are based on number of pixels. For example, if the dimensions of an image are 400x300 (width x height), then the total number of pixels in the image is 120000.

The basketball image has dimensions 200x200 (200px width, 200px height), while the tennis ball image is 72x66 (72px width, 66px height).

The function `tf.read_file` takes the file name as its required argument and returns the contents of the file as a tensor with type `tf.string`. When the input file is an image, the output of `tf.read_file` will be the raw byte data of the image file. Although the raw byte output represents the image's pixel data, it cannot be used directly. In the next chapter we'll see how we can convert

the raw data to usable pixel data.

B. Pixels

So what exactly is a pixel? A pixel is essentially just a point on an image, with a specific shade, color, and/or opacity. We normally represent a pixel as a single integer or multiple integers. Pixels take a specific form based on the interpretation of the image, which is usually one of the following:

- ***Grayscale***: Viewing the image as shades of black and white. Each pixel is an integer between 0-255, where 0 is completely black and 255 is completely white.
- ***RGB***: The default interpretation for color images. Each pixel is made up of 3 integers between 0-255, where the integers represent the intensity of red, green, and blue, respectively, for the pixel.
- ***RGBA***: An extension of RGB with an added *alpha* field. The alpha field represents the opacity of an image, and in this Lab we'll represent a pixel's alpha value as an integer from 0-255 with 0 being fully transparent and 255 being fully opaque.

We can choose to interpret an image however we want, but there is usually one interpretation that is optimal. For example, we could interpret a black and white image with RGB pixel values, but it is more efficient to view it as a grayscale image (3x fewer integers used). On the other hand, it would be unwise to interpret a colored image using grayscale pixels, since the pixels won't be able to capture any of the actual colors.

Time to Code!

In the next three chapters you'll be working on the `decode_image` function, which decodes image data from a file.

In this chapter, we'll use `tf.read_file` to read the image binary data from the file.

Set `value` equal to the output of `tf.read_file` with argument `filename`.

```
import tensorflow as tf

# Decode image data from a file in Tensorflow
def decode_image(filename, image_type, resize_shape, channels=0):
    # CODE HERE
```

```
    pass
```