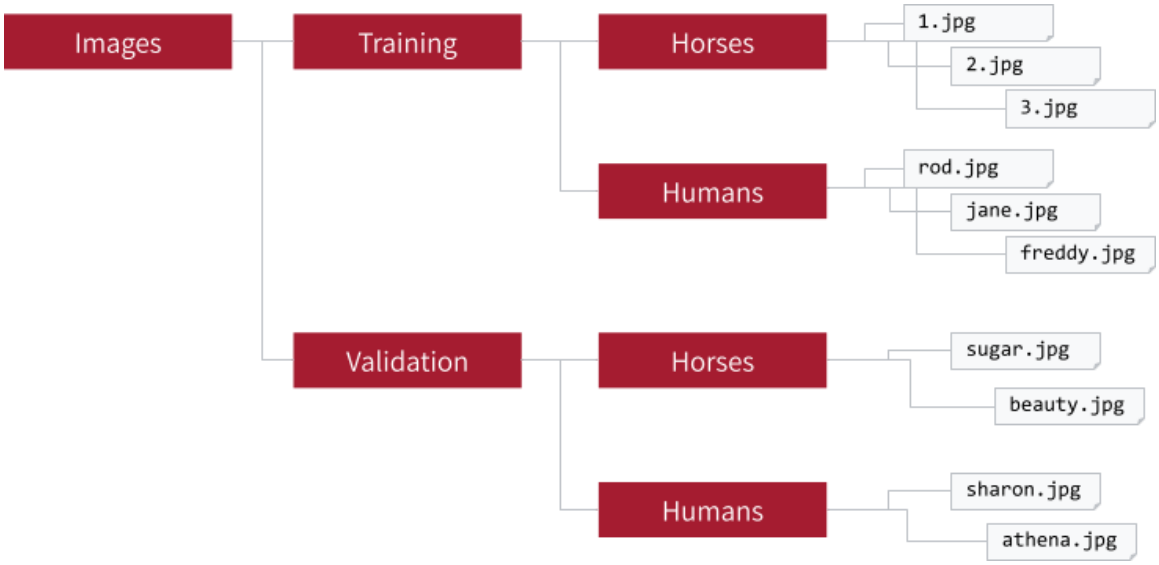
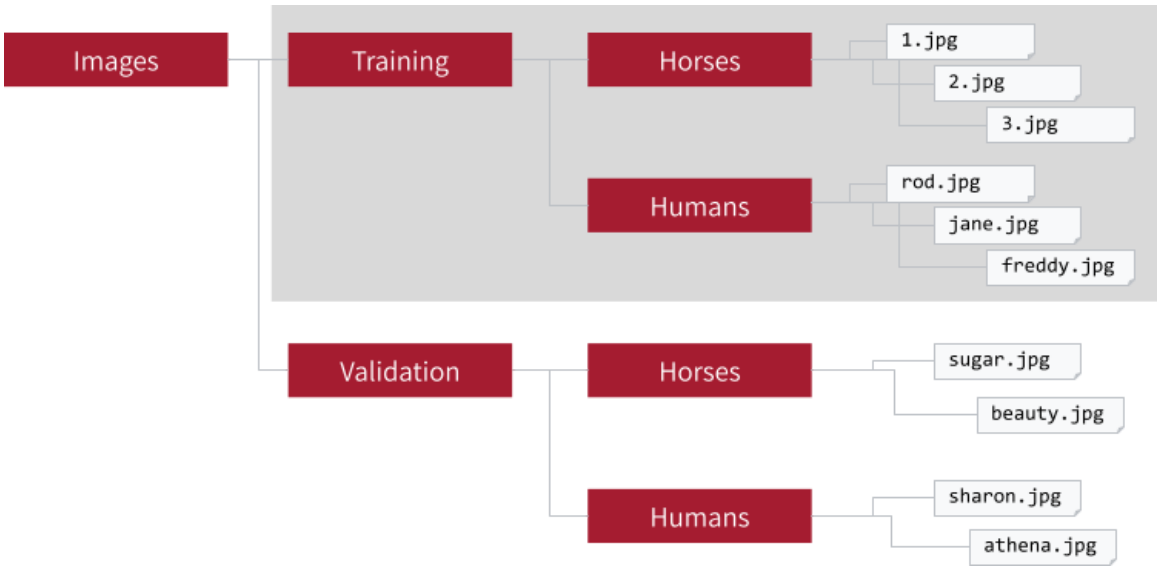
TFDS for Image Data

In the previous examples you saw how to download a ZIP file containing images, and how, if the images are sorted into subdirectories, you could use a generator to automatically label the images based on their directories.

So, for example, in the horses-or-humans dataset, if you downloaded both the training and validation zip files, you'd end up with a structure like this:



Within your 'Images' directory, if you unzip the training files into a 'training' folder, it would have subdirectories for 'Horses and Humans'.



By pointing an ImageDataGenerator at this subdirectory, it would automatically label images in the ‘Horses’ directory as horses, and similarly the images in the Humans directory would be labelled as such.

All of this requires the images to be sorted into named subdirectories, downloaded, and unzipped into the same structure. It works well for training, but it isn’t the *only* way of flowing data into a neural network. And this is just one way of representing image data -- there are many -- and often you have to have some domain-specific knowledge of the data (at the very least the structure of a zip like this, but it’s often more complicated) and code around that to get the data into a way that you can train a neural network with.

TensorFlow Datasets

A common way to flow data into your network for training is TensorFlow Datasets.

The goal behind TensorFlow Datasets (TFDS) is to expose datasets in a way that’s easy to consume, where all the preprocessing steps of acquiring the data and getting it into TensorFlow-friendly APIs is done for you.

You’ve already seen a little of this idea with how Keras handled Fashion MNIST.

As a recap, all you had to do to get the data was this:

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

TensorFlow Datasets builds on this idea, but greatly expands not only the number of datasets available but the diversity of dataset types. The list of available datasets is growing all the time including pictures, text, audio, video and more. Check out the link to see some of the datasets.

TensorFlow Datasets is a separate install from TensorFlow, so be sure to install it before trying out any samples! If you are using Google Colab, it’s already preinstalled.

If you need to install it, you can do so with a `pip` command:

```
pip install tensorflow-datasets
```

Once it’s installed, you can use it to get access to a dataset with `tfds.load`, passing it the name of the desired dataset. For example, if you could use Fashion MNIST as follows:

```
import tensorflow as tf
import tensorflow_datasets as tfds
mnist_data = tfds.load("fashion_mnist")
for item in mnist_data:
  print(item)
```

Two very important concepts to learn with TensorFlow Datasets are the Splits API -- that gives you a flexible way of splitting up data into Training, Testing, Validation sets, and Mapping Functions, which allow you to do things like Augmentation. So, for example, you saw how to do Image Augmentation on a generator, but if you're no longer using generators you'll need an alternative! TFDS makes it simple with the mapping functions.

Here's code as an example:

```
def augmentimages(image, label):
  image = tf.cast(image, tf.float32)
  image = (image/255)
  image = tf.image.random_flip_left_right(image)
  return image, label

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

In this case, tfds.load is used to get the horses or humans dataset. It's pre-split into a 'train' subset with the training data, so you can request that. Then, once you have data, you can call it's 'map' method, passing it a function like 'AugmentImages' as shown, and from within there you can do your image augmentation.

There's a lot to learn with TFDS, and it's a really powerful API. Visit: https://www.tensorflow.org/datasets to go deeper!