In [2]:
```
import os
import tensorflow as tf
```

```
2023-08-12 12:45:38.862229: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

Summary

The following module will take in a pre organized directory collection of jpg images and produce a TFRecord file (the tensor flow reccomended data input format). The module assumes that the images have been sorted into a directory hierarchy as follows:

- Main data directory
  - Train
    - ClassName
    - ClassName
    - ClassName
  - Test
    - ClassName
    - ClassName
    - ClassName
  - Validate
    - ClassName
    - ClassName
    - ClassName

The module also assumes the images are in jpg encoding. To change encoding, go to serialize_image( ):

- image = tf.image.decode_jpeg(image_string, channels = 3) #confirm image encoding is jpg

  change ft.image.decode_jpeg to the correct method found here: https://www.tensorflow.org/api_docs/python/tf/io/decode_image

Functions

In [3]:
```
# Main Function
def record_writer(data_dir, dim, class_list, file_names):
    image_paths, labels = get_paths_and_labels(data_dir, class_list)
    serialize_images(image_paths, labels, filename, dim)


# Necessary functions for feature organization per the TensorFlow Documentation:
# url:  https://www.tensorflow.org/tutorials/load_data/tfrecord#walkthrough_reading_and_writing_image_data

def _int64_feature(value):
  """Returns an int64_list from a bool / enum / int / uint."""
  return tf.train.Feature(int64_list=tf.train.Int64List(value=[value]))

def _bytes_feature(value):
  """Returns a bytes_list from a string / byte."""
  if isinstance(value, type(tf.constant(0))):
    value = value.numpy() # BytesList won't unpack a string from an EagerTensor.
  return tf.train.Feature(bytes_list=tf.train.BytesList(value=[value]))


# Create a dictionary with features that may be relevant.
def image_example(image_string, label):
  image_shape = tf.io.decode_jpeg(image_string).shape

  feature = {
      'height': _int64_feature(image_shape[0]),
      'width': _int64_feature(image_shape[1]),
      'depth': _int64_feature(image_shape[2]),
      'label': _int64_feature(label),
      'image_raw': _bytes_feature(image_string),
  }

  return tf.train.Example(features=tf.train.Features(feature=feature))


# Get image file paths and labels
def get_paths_and_labels(data_dir, class_list):
    class_names = class_list
    image_paths = []
    labels = []
    for i, class_name in enumerate(class_names):
        class_path = os.path.join(data_dir, class_name)
        if os.path.isdir(class_path):
            label = i
            for filename in os.listdir(class_path):
                if filename.lower().endswith(('.jpg')):
                    image_path = os.path.join(class_path, filename)
                    image_paths.append(image_path)
                    labels.append(label)
    return image_paths, labels

# Serialize to string per tensorflow documentation
def serialize_images(image_paths, labels, filename, dim):
    with tf.io.TFRecordWriter(filename) as writer:
        for image_path, label in zip(image_paths, labels):
            image_string = open(image_path, 'rb').read()
            image = tf.image.decode_jpeg(image_string, channels = 3) #confirm image encoding is jpg
            image = tf.image.resize(image, dim)
            image = tf.cast(image, tf.uint8)
            image_string = tf.image.encode_jpeg(image).numpy()
            tf_example = image_example(image_string, label)
            writer.write(tf_example.SerializeToString())
```

Main: TFRecord writing

In [4]:
```
###Main###

# Definitions; Enter parameters

#Name of the main directory containing data, plus its path if it isn't in the same directory as this file; String
data_dir =

#Resize dimmensions; List (eg. [64,64] [128,128])
dim =

#List of classification labels; List
class_list =

#Filename for TFRecord (eg. 'train.tfrecord');
filename =


## Execute with given definitions
record_writer(data_dir, dim, class_list, filename)
```

In [ ]:
```
###Example###

# In this example I had pre split my training, testing, and validating data into separate folders inside data main.
# Each of those sub folders were pre split into their classifications (exo_neg, exo_pos)

#Name of the main directory containing data, plus its path if it isn't in the same directory as this file; String
data_dir = 'MyProject/data_main/train'

#Resize dimmensions; List (eg. [64,64] [128,128])
dim = [128,128]

#List of classification labels; List
class_list = ['exo_neg', 'exo_pos']

#Filename for TFRecord (eg. 'train.tfrecord');
filename = 'test_delete.tfrecords'


## Execute with given definitions
record_writer(data_dir, dim, class_list, filename)
```