In [10]:

```
import os
import random
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

class InfiniteDataLoader:
    def __init__(self, data_dir, batch_size, target_size=(224, 224), shuffle
        """
        Constructer.

        Args:
            data_dir: dataset Path.
            batch_size: batch_size.
            target_size: image size.
            shuffle: shuffle.
        """
        self.data_dir = data_dir
        self.batch_size = batch_size
        self.target_size = target_size
        self.shuffle = shuffle
        self.classes = sorted(os.listdir(data_dir))
        self.num_classes = len(self.classes) # folder count to count classes
#         print(self.num_classes)
        self.image_data_generator = ImageDataGenerator() #keras function
        self.image_generator = self.image_data_generator.flow_from_directory
            data_dir,
            target_size=target_size,
            batch_size=batch_size,
            class_mode='categorical',
            shuffle=shuffle
        )

    def __iter__(self): # to get all batch at once
        return self

    def __next__(self): # to get on by on batch
        images, labels = next(self.image_generator)
        return images, labels
```

executed in 10ms, finished 21:53:22 2024-03-31

In [13]:

```
# Example usage:
data_dir = 'train/train/'
batch_size = 8

data_loader = InfiniteDataLoader(data_dir, batch_size)

# Iterate and get batches of images infinitely
for i in range(10):  # example: iterate for 10 batches
    images, labels = next(data_loader)
    print(f"Batch {i+1}:")
    print(images.shape, labels)
```

executed in 593ms, finished 21:53:51 2024-03-31

```
3
Found 1034 images belonging to 3 classes.
Batch 1:
(5, 224, 224, 3) [[0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 0. 1.]
 [0. 0. 1.]]
Batch 2:
(5, 224, 224, 3) [[0. 0. 1.]
 [0. 1. 0.]
 [0. 1. 0.]
 [1. 0. 0.]
 [1. 0. 0.]]
Batch 3:
(5, 224, 224, 3) [[0. 0. 1.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
Batch 4:
(5, 224, 224, 3) [[1. 0. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]]
Batch 5:
(5, 224, 224, 3) [[1. 0. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]]
Batch 6:
(5, 224, 224, 3) [[1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 0. 1.]
 [0. 0. 1.]]
Batch 7:
(5, 224, 224, 3) [[1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]]
Batch 8:
(5, 224, 224, 3) [[0. 1. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
Batch 9:
(5, 224, 224, 3) [[0. 1. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]]
Batch 10:
```

```
(5, 224, 224, 3) [[0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 1. 0.]]
```

In [9]:

```
next(data_loader)
```

executed in 60ms, finished 21:51:02 2024-03-31

```
          [[ 61.,  44.,  18.],
           [ 50.,  31.,  14.],
           [ 43.,  25.,  11.],
           ...,
           [138., 162., 112.],
           [142., 156., 107.],
           [132., 142.,  92.]],

          [[ 47.,  33.,   6.],
           [ 44.,  27.,   9.],
           [ 40.,  22.,  10.],
           ...,
           [139., 167., 118.],
           [140., 164., 114.],
           [153., 175., 126.]]]], dtype=float32),
 array([[1.],
        [1.],
        [1.],
        [1.],
        [1.]], dtype=float32))
```

In [ ]: