Branch: master

Find file | Copy path

AI-Capstone-Project-with-Deep-Learning / DL0321EN-2-1-Data-Preparation-py-v1.0.ipynb

KonuTech Add files via upload

bd6f2a5 on Feb 2

1 contributor

Download | History

1.08 MB

(https://cognitiveclass.ai)

<h1 align=center><font size = 5>Data Preparation</font></h1>

Introduction

In this lab, you will learn how to load images and manipulate them for training using Keras ImageDataGenerator.

Table of Contents

Download Data

For your convenience, I have placed the data on a server which you can retrieve easily using the **wget** command. So let's run the following line of code to get the data. Given the large size of the image dataset, it might take some time depending on your internet speed.

```
In [1]: ## get the data
        #!wget https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DL03
        21EN/data/concrete_data_week2.zip
```

And now if you check the left directory pane, you should see the zipped file *concrete_data_week2.zip* appear. So, let's go ahead and unzip the file to access the images. Given the large number of images in the dataset, this might take a couple of minutes, so please be patient, and wait until the code finishes running.

```
In [2]: #!unzip concrete_data_week2.zip
```

Now, you should see the folder *concrete_data_week2* appear in the left pane. If you open this folder by double-clicking on it, you will find that it contains two folders: *Positive* and *Negative*. These are the same folders that we saw in the lab in the previous module of this course, where *Negative* is the negative class and it represents the concrete images with no cracks. *Positive* on the other hand is the positive class and represents the concrete images with cracks.

Important Note: There are thousands and thousands of images in each folder, so please don't attempt to double click on the *Negative* and *Positive* folders. This may consume all of your memory and you may end up with a **50*** error. So please **DO NOT DO IT**.

Import Libraries and Packages

Before we proceed, let's import the libraries and packages that we will need to complete the rest of this lab.

```
In [3]: import os
        import numpy as np
        import matplotlib.pyplot as plt

        import keras
        from keras.preprocessing.image import ImageDataGenerator
```

```
Using TensorFlow backend.
```

You can check the content of ./concrete_data_week2 by running the following:

You can check the content of ./concrete_data_week2 by running the following:

```
In [12]: !ls ./concrete_data_week2.2
```

```
Negative  Positive
```

or the following:

```
In [13]: os.listdir('concrete_data_week2.2')
```

```
Out[13]: ['Negative', 'Positive']
```

Construct an ImageDataGenerator Instance

In this section, you will learn how to define a Keras ImageDataGenerator instance and use it to load and manipulate data for building a deep learning model.

Before we proceed, let's define a variable that represents the path to the folder containing our data which is `concrete_data_week2` in this case.

```
In [27]: dataset_dir = './concrete_data_week2.2'
```

Keras ImageDataGenerator requires images be arranged in a certain folder hierarchy, where the main directory would contain folders equal to the number of classes in your problem. Since in this case we are trying to build a classifier of two classes, then our main directory, which is `concrete_data_week2`, should contain two folders, one for each class. This has already been done for you as the negative images are in one folder and the positive images are in another folder.

Let's go ahead and define an instance of the Keras ImageDataGenerator.

Standard ImageDataGenerator

You can define a standard one like this, where you are simply using the ImageDataGenerator to train your model in batches.

```
In [57]: # instantiate your image data generator
         data_generator = ImageDataGenerator()
```

Next, you use the `flow_from_directory` methods to loop through the images in batches. In this method, you pass the directory where the images reside, the size of each batch, *batch_size*, and since batches are sampled randomly, then you can also specify a random seed, *seed*, if you would like to reproduce the batch sampling. In case you would like to resize your images, then you can using the *target_size* argument to accomplish that.

```
In [58]: image_generator = data_generator.flow_from_directory(
             dataset_dir,
             batch_size=4,
             class_mode='categorical',
             seed=24
             )
```

```
Found 40000 images belonging to 2 classes.
```

What is great about this method, is it prints a summary of it found in the directory passed. Here, it found 40,000 images in total belonging to 2 classes.

Now, to access the batches, you use the `next` method as follows:

```
In [59]: first_batch = image_generator.next()
         first_batch
```

```
Out[59]: (array([[[[151., 149., 150.],
                   [153., 151., 152.],
                   [155., 153., 154.],
                   ...,
                   [153., 149., 148.],
                   [153., 149., 148.],
                   [153., 149., 148.]],
                  [[152., 150., 151.],
                   [154., 152., 153.],
                   [156., 154., 155.],
                   ...,
                   [154., 150., 149.],
                   [154., 150., 149.],
                   [154., 150., 149.]],

                  [[153., 151., 152.],
                   [155., 153., 154.],
                   [156., 154., 155.],
                   ...,
                   [156., 152., 151.],
                   [156., 152., 151.],
                   [156., 152., 151.]],

                  ...,

                  [[142., 140., 143.],
                   [142., 140., 143.],
                   [142., 140., 143.],
                   ...,
                   [139., 135., 136.],
                   [142., 138., 139.],
                   [145., 141., 142.]],

                  [[142., 140., 143.],
                   [142., 140., 143.],
                   [142., 140., 143.],
                   ...,
                   [140., 136., 137.],
                   [142., 138., 139.],
                   [146., 142., 143.]],

                  [[142., 140., 143.],
                   [142., 140., 143.],
                   [142., 140., 143.],
                   ...,
                   [141., 137., 138.],
                   [144., 140., 141.],
                   [147., 143., 144.]]],


                 [[[189., 186., 181.],
                   [191., 188., 183.],
                   [194., 191., 186.],
                   ...,
                   [186., 185., 181.],
                   [184., 183., 179.],
                   [183., 182., 178.]],

                  [[188., 185., 180.],
                   [190., 187., 182.],
                   [194., 191., 186.],
                   ...,
                   [186., 185., 181.],
                   [185., 184., 180.],
                   [184., 183., 179.]],

                  [[186., 183., 178.],
                   [189., 186., 181.],
                   [193., 190., 185.],
                   ...,
                   [186., 185., 181.],
                   [185., 184., 180.],
                   [185., 184., 180.]],

                  ...,

                  [[188., 185., 180.],
                   [187., 184., 179.],
                   [186., 183., 178.],
                   ...,
                   [170., 170., 162.],
                   [170., 170., 162.],
                   [171., 171., 163.]],

                  [[188., 185., 180.],
                   [187., 184., 179.],
                   [186., 183., 178.],
                   ...,
                   [171., 171., 163.],
                   [171., 171., 163.],
                   [172., 172., 164.]],

                  [[188., 185., 180.],
                   [187., 184., 179.],
                   [186., 183., 178.],
                   ...,
                   [179., 179., 171.],
                   [179., 179., 171.],
                   [179., 179., 171.]]],


                 [[[181., 173., 162.],
                   [176., 168., 157.],
                   [168., 160., 149.],
                   ...,
                   [171., 167., 156.],
                   [170., 166., 155.],
                   [169., 165., 154.]],

                  [[180., 172., 161.],
                   [176., 168., 157.],
                   [170., 162., 151.],
                   ...,
                   [172., 168., 157.],
                   [171., 167., 156.],
                   [170., 166., 155.]],

                  [[179., 171., 160.],
                   [176., 168., 157.],
                   [171., 163., 152.],
                   ...,
                   [174., 170., 159.],
                   [173., 169., 158.],
                   [172., 168., 157.]],

                  ...,

                  [[175., 171., 160.],
                   [178., 174., 163.],
                   [182., 178., 167.],
                   ...,
                   [173., 168., 162.],
                   [173., 168., 162.],
                   [173., 168., 162.]],

                  [[175., 171., 160.],
                   [178., 174., 163.],
                   [182., 178., 167.],
                   ...,
                   [175., 170., 164.],
                   [175., 170., 164.],
                   [175., 170., 164.]],

                  [[175., 171., 160.],
                   [178., 174., 163.],
                   [182., 178., 167.],
                   ...,
                   [177., 172., 166.],
                   [177., 172., 166.],
                   [177., 172., 166.]]],


                 [[[150., 149., 147.],
                   [148., 147., 145.],
                   [146., 145., 143.],
                   ...,
                   [199., 195., 192.],
                   [194., 190., 187.],
                   [189., 185., 182.]],

                  [[153., 152., 150.],
                   [152., 151., 149.],
                   [151., 150., 148.],
                   ...,
                   [196., 192., 189.],
                   [192., 188., 185.],
                   [186., 182., 179.]],

                  [[153., 152., 150.],
                  [155., 154., 152.],
                  [156., 155., 153.],
                  ...,
                  [192., 188., 185.],
                  [188., 184., 181.],
                  [182., 178., 175.]],

                 ...,

                 [[179., 175., 174.],
                  [180., 176., 175.],
                  [181., 177., 176.],
                  ...,
                  [174., 170., 167.],
                  [173., 169., 166.],
                  [173., 169., 166.]],

                 [[180., 176., 175.],
                  [180., 176., 175.],
                  [180., 176., 175.],
                  ...,
                  [176., 172., 169.],
                  [175., 171., 168.],
                  [173., 169., 166.]],

                 [[182., 178., 177.],
                  [181., 177., 176.],
                  [179., 175., 174.],
                  ...,
                  [180., 176., 173.],
                  [177., 173., 170.],
                  [174., 170., 167.]]]], dtype=float32), array([[1., 0.],
                [0., 1.],
                [1., 0.],
                [0., 1.]], dtype=float32))
```

As you can see, this returned the images along with their labels. Therefore, the following returns the images only,

```
In [60]: first_batch_images = image_generator.next()[0]
         first_batch_images
```

```
Out[60]: array([[[[172., 174., 169.],
                  [176., 178., 173.],
                  [181., 183., 178.],
                  ...,
                  [157., 159., 154.],
                  [156., 158., 153.],
                  [154., 156., 151.]],

                 [[170., 172., 167.],
                  [172., 174., 169.],
                  [174., 176., 171.],
                  ...,
                  [155., 157., 152.],
                  [154., 156., 151.],
                  [153., 155., 150.]],

                 [[169., 171., 166.],
                  [167., 169., 164.],
                  [166., 168., 163.],
                  ...,
                  [152., 154., 149.],
                  [152., 154., 149.],
                  [151., 153., 148.]],

                 ...,

                 [[153., 154., 148.],
                  [155., 156., 150.],
                  [158., 159., 153.],
                  ...,
                  [143., 144., 138.],
                  [144., 145., 139.],
                  [145., 146., 140.]],

                 [[147., 148., 142.],
                  [150., 151., 145.],
                  [153., 154., 148.],
                  ...,
                  [142., 143., 137.],
                  [143., 144., 138.],
         [144., 145., 139.]],

        [[141., 142., 136.],
         [145., 146., 140.],
         [149., 150., 144.],
         ...,
         [141., 142., 136.],
         [142., 143., 137.],
         [143., 144., 138.]]],


       [[[183., 180., 173.],
         [173., 170., 163.],
         [171., 168., 161.],
         ...,
         [175., 172., 165.],
         [175., 172., 165.],
         [174., 171., 164.]],

        [[178., 175., 168.],
         [173., 170., 163.],
         [174., 171., 164.],
         ...,
         [171., 168., 161.],
         [171., 168., 161.],
         [170., 167., 160.]],

        [[169., 166., 159.],
         [169., 166., 159.],
         [175., 172., 165.],
         ...,
         [166., 163., 156.],
         [166., 163., 156.],
         [164., 161., 154.]],

        ...,

        [[188., 185., 178.],
         [186., 183., 176.],
         [183., 180., 173.],
         ...,
         [150., 149., 144.],
         [147., 146., 141.],
         [144., 143., 138.]],

        [[186., 183., 176.],
         [184., 181., 174.],
         [181., 178., 171.],
         ...,
         [150., 149., 144.],
         [147., 146., 141.],
         [144., 143., 138.]],

        [[184., 181., 174.],
         [182., 179., 172.],
         [178., 175., 168.],
         ...,
         [150., 149., 144.],
         [148., 147., 142.],
         [145., 144., 139.]]],


       [[[165., 160., 154.],
         [163., 158., 152.],
         [160., 155., 149.],
         ...,
         [155., 150., 146.],
         [156., 151., 147.],
         [157., 152., 148.]],

        [[164., 159., 153.],
         [162., 157., 151.],
         [160., 155., 149.],
         ...,
         [157., 152., 148.],
         [158., 153., 149.],
         [159., 154., 150.]],

        [[163., 158., 152.],
         [162., 157., 151.],
         [161., 156., 150.],
         ...,
         [160., 155., 151.],
         [160., 155., 151.],
         [161., 156., 152.]],

        ...,

        [[159., 154., 148.],
         [164., 159., 153.],
         [169., 164., 158.],
         ...,
         [145., 140., 134.],
         [145., 140., 134.],
         [145., 140., 134.]],

        [[158., 153., 147.],
         [164., 159., 153.],
         [169., 164., 158.],
         ...,
         [146., 141., 135.],
         [146., 141., 135.],
         [146., 141., 135.]],

        [[158., 153., 147.],
         [163., 158., 152.],
         [169., 164., 158.],
         ...,
         [147., 142., 136.],
         [147., 142., 136.],
         [147., 142., 136.]]],


       [[[142., 133., 124.],
         [144., 135., 126.],
         [147., 138., 129.],
         ...,
         [176., 172., 161.],
         [178., 174., 163.],
         [181., 177., 166.]],

        [[140., 131., 122.],
         [144., 135., 126.],
         [150., 141., 132.],
         ...,
         [179., 175., 164.],
         [179., 175., 164.],
         [180., 176., 165.]],

        [[141., 132., 123.],
         [147., 138., 129.],
         [155., 146., 137.],
         ...,
         [176., 172., 161.],
         [175., 171., 160.],
         [174., 170., 159.]],

        ...,

        [[170., 166., 154.],
         [177., 173., 161.],
         [183., 179., 167.],
         ...,
         [171., 168., 161.],
         [172., 169., 162.],
         [173., 170., 163.]],

        [[171., 167., 155.],
         [178., 174., 162.],
         [183., 179., 167.],
         ...,
         [171., 168., 161.],
         [172., 169., 162.],
         [173., 170., 163.]],

        [[173., 169., 157.],
         [179., 175., 163.],
         [182., 178., 166.],
         ...,
         [172., 169., 162.],
         [172., 169., 162.],
         [173., 170., 163.]]]], dtype=float32)
```

and the following returns the labels only.

```
In [32]: first_batch_labels = image_generator.next()[1]
         first_batch_labels
```

```
Out[32]: array([[1., 0.],
                [0., 1.],
                [0., 1.],
                [0., 1.]], dtype=float32)
```

Custom ImageDataGenerator

You can also specify some transforms, like scaling, rotations, and flips, that you would like applied to the images when you define an ImageDataGenerator object. Say you want to normalize your images, then you can define your ImageDataGenerator instance as follows:

```
In [20]: # instantiate your image data generator
         data_generator = ImageDataGenerator(
             rescale=1./255
         )
```

And then you proceed with defining your *image_generator* using the *flow_from_directory* method, just like before.

```
In [21]: image_generator = data_generator.flow_from_directory(
             dataset_dir,
             batch_size=4,
             class_mode='categorical',
             seed=24
             )
```

```
         Found 40000 images belonging to 2 classes.
```

However, now we explore the first batch using the *next* method,

```
In [22]: first_batch = image_generator.next()
         first_batch
```

```
Out[22]: (array([[[[0.5921569 , 0.58431375, 0.5882353 ],
                   [0.6       , 0.5921569 , 0.59607846],
                   [0.60784316, 0.6       , 0.6039216 ],
                   ...,
                   [0.6       , 0.58431375, 0.5803922 ],
                   [0.6       , 0.58431375, 0.5803922 ],
                   [0.6       , 0.58431375, 0.5803922 ]],

                  [[0.59607846, 0.5882353 , 0.5921569 ],
                   [0.6039216 , 0.59607846, 0.6       ],
                   [0.6117647 , 0.6039216 , 0.60784316],
                   ...,
                   [0.6039216 , 0.5882353 , 0.58431375],
                   [0.6039216 , 0.5882353 , 0.58431375],
                   [0.6039216 , 0.5882353 , 0.58431375]],

                  [[0.6       , 0.5921569 , 0.59607846],
                   [0.60784316, 0.6       , 0.6039216 ],
                   [0.6117647 , 0.6039216 , 0.60784316],
                   ...,
                   [0.6117647 , 0.59607846, 0.5921569 ],
                   [0.6117647 , 0.59607846, 0.5921569 ],
                   [0.6117647 , 0.59607846, 0.5921569 ]],

                  ...,

                  [[0.5568628 , 0.54901963, 0.56078434],
                   [0.5568628 , 0.54901963, 0.56078434],
                   [0.5568628 , 0.54901963, 0.56078434],
                   ...,
                   [0.54509807, 0.5294118 , 0.53333336],
                   [0.5568628 , 0.5411765 , 0.54509807],
                   [0.5686275 , 0.5529412 , 0.5568628 ]],

                  [[0.5568628 , 0.54901963, 0.56078434],
                   [0.5568628 , 0.54901963, 0.56078434],
                   [0.5568628 , 0.54901963, 0.56078434],
                   ...,
                   [0.54901963, 0.53333336, 0.5372549 ],
                   [0.5568628 , 0.5411765 , 0.54509807],
   [0.57254905, 0.5568628 , 0.56078434]],

  [[0.5568628 , 0.54901963, 0.56078434],
   [0.5568628 , 0.54901963, 0.56078434],
   [0.5568628 , 0.54901963, 0.56078434],
   ...,
   [0.5529412 , 0.5372549 , 0.5411765 ],
   [0.5647059 , 0.54901963, 0.5529412 ],
   [0.5764706 , 0.56078434, 0.5647059 ]]],


 [[[0.7411765 , 0.7294118 , 0.70980394],
   [0.7490196 , 0.7372549 , 0.7176471 ],
   [0.7607844 , 0.7490196 , 0.7294118 ],
   ...,
   [0.7294118 , 0.7254902 , 0.70980394],
   [0.72156864, 0.7176471 , 0.7019608 ],
   [0.7176471 , 0.7137255 , 0.69803923]],

  [[0.7372549 , 0.7254902 , 0.7058824 ],
   [0.74509805, 0.73333335, 0.7137255 ],
   [0.7607844 , 0.7490196 , 0.7294118 ],
   ...,
   [0.7294118 , 0.7254902 , 0.70980394],
   [0.7254902 , 0.72156864, 0.7058824 ],
   [0.72156864, 0.7176471 , 0.7019608 ]],

  [[0.7294118 , 0.7176471 , 0.69803923],
   [0.7411765 , 0.7294118 , 0.70980394],
   [0.7568628 , 0.74509805, 0.7254902 ],
   ...,
   [0.7294118 , 0.7254902 , 0.70980394],
   [0.7254902 , 0.72156864, 0.7058824 ],
   [0.7254902 , 0.72156864, 0.7058824 ]],

  ...,

  [[0.7372549 , 0.7254902 , 0.7058824 ],
   [0.73333335, 0.72156864, 0.7019608 ],
   [0.7294118 , 0.7176471 , 0.69803923],
   ...,
   [0.6666667 , 0.6666667 , 0.63529414],
   [0.6666667 , 0.6666667 , 0.63529414],
   [0.67058825, 0.67058825, 0.6392157 ]],

  [[0.7372549 , 0.7254902 , 0.7058824 ],
   [0.73333335, 0.72156864, 0.7019608 ],
   [0.7294118 , 0.7176471 , 0.69803923],
   ...,
   [0.67058825, 0.67058825, 0.6392157 ],
   [0.67058825, 0.67058825, 0.6392157 ],
   [0.6745098 , 0.6745098 , 0.6431373 ]],

  [[0.7372549 , 0.7254902 , 0.7058824 ],
   [0.73333335, 0.72156864, 0.7019608 ],
   [0.7294118 , 0.7176471 , 0.69803923],
   ...,
   [0.7019608 , 0.7019608 , 0.67058825],
   [0.7019608 , 0.7019608 , 0.67058825],
   [0.7019608 , 0.7019608 , 0.67058825]]],


 [[[0.70980394, 0.6784314 , 0.63529414],
   [0.6901961 , 0.65882355, 0.6156863 ],
   [0.65882355, 0.627451  , 0.58431375],
   ...,
   [0.67058825, 0.654902  , 0.6117647 ],
   [0.6666667 , 0.6509804 , 0.60784316],
   [0.6627451 , 0.64705884, 0.6039216 ]],

  [[0.7058824 , 0.6745098 , 0.6313726 ],
   [0.6901961 , 0.65882355, 0.6156863 ],
   [0.6666667 , 0.63529414, 0.5921569 ],
   ...,
   [0.6745098 , 0.65882355, 0.6156863 ],
   [0.67058825, 0.654902  , 0.6117647 ],
   [0.6666667 , 0.6509804 , 0.60784316]],

  [[0.7019608 , 0.67058825, 0.627451  ],
   [0.6901961 , 0.65882355, 0.6156863 ],
   [0.67058825, 0.6392157 , 0.59607846],
           ...,
           [0.68235296, 0.6666667 , 0.62352943],
           [0.6784314 , 0.6627451 , 0.61960787],
           [0.6745098 , 0.65882355, 0.6156863 ]],

          ...,

          [[0.6862745 , 0.67058825, 0.627451  ],
           [0.69803923, 0.68235296, 0.6392157 ],
           [0.7137255 , 0.69803923, 0.654902  ],
           ...,
           [0.6784314 , 0.65882355, 0.63529414],
           [0.6784314 , 0.65882355, 0.63529414],
           [0.6784314 , 0.65882355, 0.63529414]],

          [[0.6862745 , 0.67058825, 0.627451  ],
           [0.69803923, 0.68235296, 0.6392157 ],
           [0.7137255 , 0.69803923, 0.654902  ],
           ...,
           [0.6862745 , 0.6666667 , 0.6431373 ],
           [0.6862745 , 0.6666667 , 0.6431373 ],
           [0.6862745 , 0.6666667 , 0.6431373 ]],

          [[0.6862745 , 0.67058825, 0.627451  ],
           [0.69803923, 0.68235296, 0.6392157 ],
           [0.7137255 , 0.69803923, 0.654902  ],
           ...,
           [0.69411767, 0.6745098 , 0.6509804 ],
           [0.69411767, 0.6745098 , 0.6509804 ],
           [0.69411767, 0.6745098 , 0.6509804 ]]],


         [[[0.5882353 , 0.58431375, 0.5764706 ],
           [0.5803922 , 0.5764706 , 0.5686275 ],
           [0.57254905, 0.5686275 , 0.56078434],
           ...,
           [0.7803922 , 0.76470596, 0.75294125],
           [0.7607844 , 0.74509805, 0.73333335],
           [0.7411765 , 0.7254902 , 0.7137255 ]],

          [[0.6       , 0.59607846, 0.5882353 ],
           [0.59607846, 0.5921569 , 0.58431375],
           [0.5921569 , 0.5882353 , 0.5803922 ],
           ...,
           [0.7686275 , 0.75294125, 0.7411765 ],
           [0.75294125, 0.7372549 , 0.7254902 ],
           [0.7294118 , 0.7137255 , 0.7019608 ]],

          [[0.6       , 0.59607846, 0.5882353 ],
           [0.60784316, 0.6039216 , 0.59607846],
           [0.6117647 , 0.60784316, 0.6       ],
           ...,
           [0.75294125, 0.7372549 , 0.7254902 ],
           [0.7372549 , 0.72156864, 0.70980394],
           [0.7137255 , 0.69803923, 0.6862745 ]],

          ...,

          [[0.7019608 , 0.6862745 , 0.68235296],
           [0.7058824 , 0.6901961 , 0.6862745 ],
           [0.70980394, 0.69411767, 0.6901961 ],
           ...,
           [0.68235296, 0.6666667 , 0.654902  ],
           [0.6784314 , 0.6627451 , 0.6509804 ],
           [0.6784314 , 0.6627451 , 0.6509804 ]],

          [[0.7058824 , 0.6901961 , 0.6862745 ],
           [0.7058824 , 0.6901961 , 0.6862745 ],
           [0.7058824 , 0.6901961 , 0.6862745 ],
           ...,
           [0.6901961 , 0.6745098 , 0.6627451 ],
           [0.6862745 , 0.67058825, 0.65882355],
           [0.6784314 , 0.6627451 , 0.6509804 ]],

          [[0.7137255 , 0.69803923, 0.69411767],
           [0.70980394, 0.69411767, 0.6901961 ],
           [0.7019608 , 0.6862745 , 0.68235296],
           ...,
           [0.7058824 , 0.6901961 , 0.6784314 ],
           [0.69411767, 0.6784314 , 0.6666667 ],
           [0.68235296, 0.6666667 , 0.654902  ]]]], dtype=float32),
  array([[1., 0.],
  array([[1., 0.],
         [0., 1.],
         [1., 0.],
         [0., 1.]], dtype=float32))
```

we find that the values are not integer values anymore, but scaled resolution since the original number are divided by 255.

You can learn more about the Keras ImageDataGeneration class here (https://keras.io/preprocessing/image/).

Visualize Batches of Images

Let write some code to visualize a batch. We will use subplots in order to make visualizing the images easier.

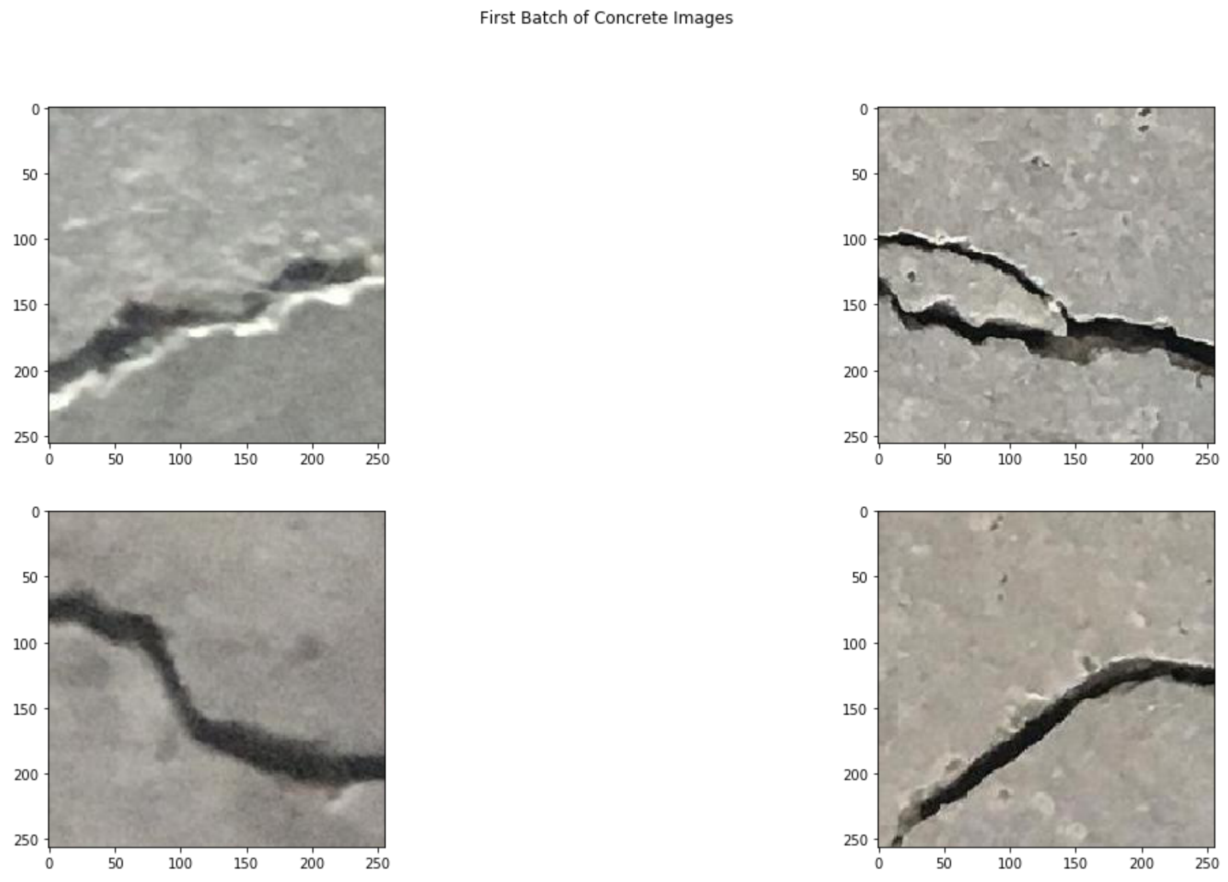
Recall that we can access our batch images as follows:

`first_batch_images = image_generator.next()[0] # first batch`

`second_batch_images = image_generator.next()[0] # second batch`

and so on.

In [61]:
```python
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 10)) # define your figure and axes

ind = 0
for ax1 in axs:
    for ax2 in ax1:
        #image_data = first_batch_images[ind]
        image_data = first_batch_images[ind].astype(np.uint8)
        ax2.imshow(image_data)
        ind += 1

fig.suptitle('First Batch of Concrete Images')
plt.show()
```



Remember that batches are sampled randomly from the data. In our first batch, we ended up with two negative image and two positive images.

Important Note: Because of a bug with the imshow function in Matplotlib, if you are plotting the unscaled RGB images, then the **image_data** in the code above would be like this:

image_data = first_batch_images[ind].astype(np.uint8)

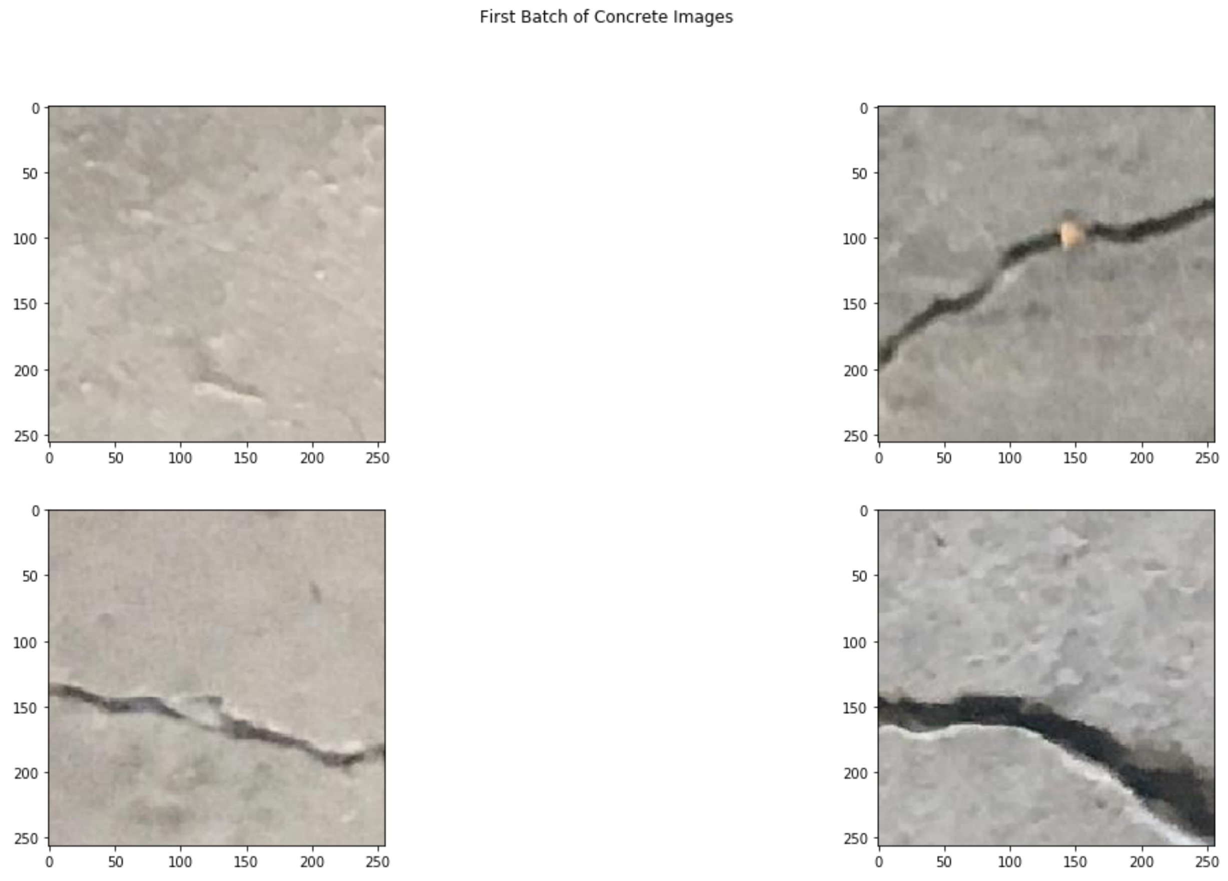
where the image_data is cast to uint8 before you call the `imshow` function.

Questions

Question: Create a plot to visualize the images in the third batch.

```
In [84]: image_generator = data_generator.flow_from_directory(
             dataset_dir,
             batch_size=4,
             class_mode='categorical',
             seed=24
             )

         for i in range(0,3):
             first_batch_images = image_generator.next()[0]

         fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 10)) # define your figure and axes

         ind = 0
         for ax1 in axs:
             for ax2 in ax1:
                 #image_data = first_batch_images[ind]
                 image_data = first_batch_images[ind].astype(np.uint8)
                 ax2.imshow(image_data)
                 ind += 1

         fig.suptitle('First Batch of Concrete Images')
         plt.show()
```

```
Found 40000 images belonging to 2 classes.
```



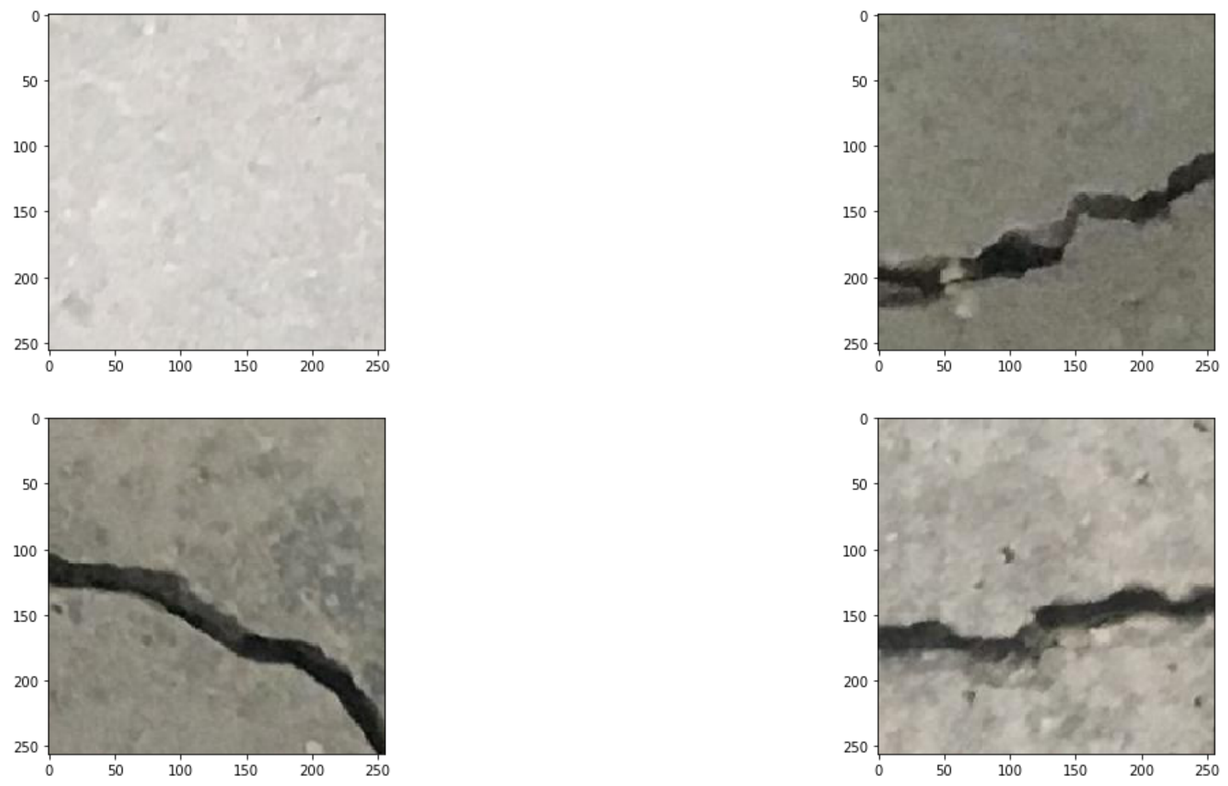
Question: How many images from each class are in the fourth batch?

```
In [85]: image_generator = data_generator.flow_from_directory(
             dataset_dir,
             batch_size=4,
             class_mode='categorical',
             seed=24
             )
```

```
for i in range(0,4):
    first_batch_images = image_generator.next()[0]

fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 10)) # define your figure and axes

ind = 0
for ax1 in axs:
    for ax2 in ax1:
        #image_data = first_batch_images[ind]
        image_data = first_batch_images[ind].astype(np.uint8)
        ax2.imshow(image_data)
        ind += 1

fig.suptitle('First Batch of Concrete Images')
plt.show()## You can use this cell to type your code to answer the above question
```

```
Found 40000 images belonging to 2 classes.
```

First Batch of Concrete Images

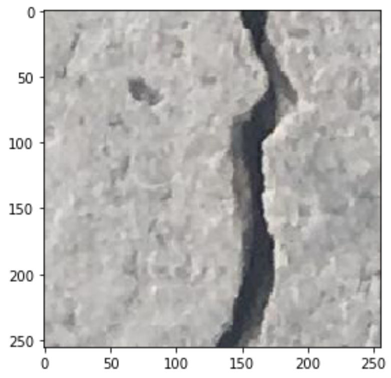


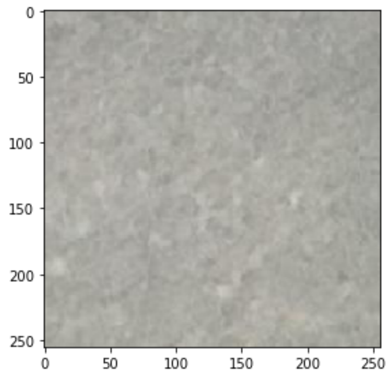
Question: Create a plot to visualize the second image in the fifth batch.

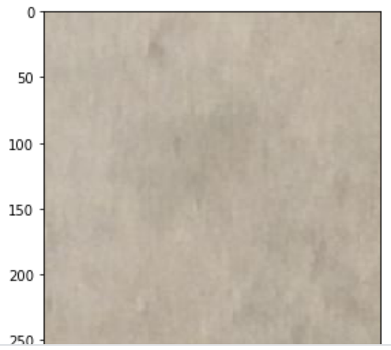
In [81]:
```
image_generator = data_generator.flow_from_directory(
    dataset_dir,
    batch_size=4,
    class_mode='categorical',
    seed=24
    )

for i in range(0,5):
    first_batch_images = image_generator.next()[0]

fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 10)) # define your figure and axes

ind = 0
for ax1 in axs:
    for ax2 in ax1:
        #image_data = first_batch_images[ind]
        image_data = first_batch_images[ind].astype(np.uint8)
        ax2.imshow(image_data)
        ind += 1

fig.suptitle('First Batch of Concrete Images')
plt.show()
```

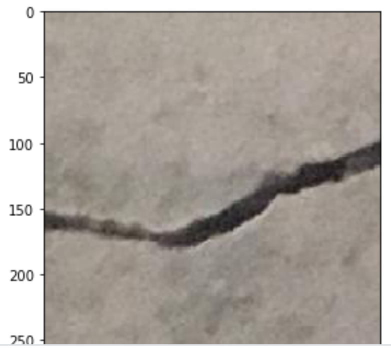
```
Found 40000 images belonging to 2 classes.
```

First Batch of Concrete Images


0
50
100
150
200
250


0
50
100
150
200
250


0
50
100
150
200
250


0
50
100
150
200
250