Watson Studio democratizes machine learning and deep learning to accelerate infusion of AI in your business to drive innovation. Watson Studio provides a suite of tools and a collaborative environment for data scientists, developers and domain experts.

Objective

- How to download and pre-process the Concrete dataset.

Crack detection has vital importance for structural health monitoring and inspection. We would like to train a network to detect Cracks, we will denote the images that contain cracks as positive and images with no cracks as negative. In this lab you are going to have to download the data and study the dataset. There are two questions in this lab, including listing the path to some of the image files as well as plotting a few images. Remember the results as you will be quizzed on them.

Table of Contents

Estimated Time Needed: **25 min**

Download Data

In this section, you are going to download the data from IBM object storage using wget, then unzip them. wget is a command the retrieves content from web servers, in this case its a zip file. Locally we store the data in the directory /resources/data . The -p creates the entire directory tree up to the given directory.

First, we download the file that contains the images:

[ ]:
```
!wget https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DL0321EN/data/images/concrete_crack_images_for_classification.zip -P /resources/data
```

We then unzip the file, this ma take a while:

[ ]:
```
!unzip -q  /resources/data/concrete_crack_images_for_classification.zip -d  /resources/data
```

We then download the files that contain the negative images:

Imports and Auxiliary Functions

The following are the libraries we are going to use for this lab:

[ ]:
```
from PIL import Image
from matplotlib.pyplot import imshow
import pandas
import matplotlib.pylab as plt
import os
import glob
```

We will use this function in the lab to plot:

[ ]:
```
def show_data(data_sample, shape = (28, 28)):
    plt.imshow(data_sample[0].numpy().reshape(shape), cmap='gray')
    plt.title('y = ' + data_sample[1])
```

Examine Files

In this section we are going to get a list of the negative image files, then plot them. Then for the first question your job to do something similar to the positive files.

The path to all the images are stored in the variable `directory` .

[ ]:
```
directory="/resources/data"
```

The images with out the cracks are stored in the file Negative

[ ]:
```
negative='Negative'
```

We can find the path to the file with all the negative images by using the function `os.path.join` . Inputs are the variable directory as well as the variable `negative` .

[ ]:
```
negative_file_path=os.path.join(directory,negative)
negative_file_path
```

Loading the File Path of Each Image

We need each the path of each image, we can find the all the file in the directory `negative_file_path` using the function `os.listdir`, the result is a list. We print out the first three elements of the list.

[ ]:
```
os.listdir(negative_file_path)[0:3]
```

We need the full path of the image so we join them as above. Here are a few samples three samples:

[ ]:
```
[os.path.join(negative_file_path,file) for file in  os.listdir(negative_file_path)][0:3]
```

In some cases, we may have files of a different type, so we have to ensure it's of type jpg. We have to check the extension using the method `endswith()`. The method `endswith()` returns True if the string ends with the specified suffix, otherwise, it will return False. Let's do a quick example:

[ ]:
```
print("test.jpg".endswith(".jpg"))
print("test.mpg".endswith(".jpg"))
```

We now have all the tools to create a list with the path to each image file. We use a List Comprehensions to make the code more compact. We assign it to the variable `negative_files` , `sort it in and display the first three elements:`

[ ]:
```
negative_files=[os.path.join(negative_file_path,file) for file in  os.listdir(negative_file_path) if file.endswith(".jpg")]
negative_files.sort()
negative_files[0:3]
```

Question 1

Using the procedure above, load all the images with cracks paths into a list called positive files, the directory of these images is called Positive. Make sure the list is sorted and display the first three elements of the list you will need this for the question so remember it.

[ ]:
```
positive="Positive"
```

Display and Analyze Image With No Cracks

We can open an image by using the `Image` Module in the PIL library, using the function open. We only require the image path; the input is the path of the image. For example we can load the first image as follows:

[ ]:
```
image1 = Image.open(negative_files[0])
# you can view the image directly
#image
```

we can plot the image

[ ]:
```
plt.imshow(image1)
plt.title("1st Image With No Cracks")
plt.show()
```

We can also plot the second image.

[ ]:
```
image2 = Image.open(negative_files[1])
plt.imshow(image2)
plt.title("2nd Image With No Cracks")
plt.show()
```

Question 2

Plot the first three images for the dataset with cracks. Don't forget. You will be asked in the quiz, so remember the image.

[ ]:
```

```

About the Authors:

Joseph Santarcangelo has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Alex Aklson. Ph.D., is a data scientist in the Digital Business Group at IBM Canada. Alex has been intensively involved in many exciting data science projects such as designing a smart system that could detect the onset of dementia in older adults using longitudinal trajectories of walking speed and home activity. Before joining IBM, Alex worked as a data scientist at Datascope Analytics, a data science consulting firm in Chicago, IL, where he designed solutions and products using a human-centred, data-driven approach. Alex received his Ph.D. in Biomedical Engineering from the University of Toronto.

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-09-18	2.0	Shubham	Migrated Lab to Markdown and added to course repo in GitLab

---

[ ]:
```

```


Simple
0
1
Python | Idle
Fully initialized
Mem: 278.83 / 6144.00 MB
Saving completed
Mode: Command
Ln 1, Col 1
English (American)
1.0_load_and_display_data.ipynb