AI4M_C1_W3_lecture_ex_01

June 25, 2020

1 AI4M Course 1 week 3 lecture notebook

Explore the data

In this week's assignment, you'll be working with 3D MRI brain scans from the public Medical Segmentation Decathlon challenge project. This is an incredibly rich dataset that provides you with labels associated with each point (voxel) inside a 3D representation of a patient's brain. Ultimately, in this week's assignment, you will train a neural network to make three-dimensional spatial segmentation predictions for common brain disorders.

In this notebook, you're all set up to explore this exciting dataset. Run the code below and tweak it to explore further!

1.0.1 Import packages

For this lab, you'll import some of the packages you've seen before (`numpy`, `matplotlib` and `seaborn`) as well as some new ones for reading (`nibabel`) and visualizing (`itk`, `itkwidgets`, `ipywidgets`) the data. Run the next cell to import these packages.

```
In [1]:  # Import all the necessary packages
         import numpy as np
         import nibabel as nib
         import itk
         import itkwidgets
         from ipywidgets import interact, interactive, IntSlider, ToggleButtons
         import matplotlib.pyplot as plt
         %matplotlib inline
         import seaborn as sns
         sns.set_style('darkgrid')
```

1.0.2 Loading Images of the brain

Run the next cell to grab a single 3D MRI brain scan

```
In [2]:  # Define the image path and load the data
         image_path = "BraTS-Data/imagesTr/BRATS_001.nii.gz"
         image_obj = nib.load(image_path)
         print(f'Type of the image {type(image_obj)}')
```

```
Type of the image <class 'nibabel.nifti1.Nifti1Image'>
```

1.0.3 Extract the data as a numpy array

Run the next cell to extract the data using the `get_fdata()` method of the image object

```
In [3]: # Extract data as numpy ndarray
        image_data = image_obj.get_fdata()
        type(image_data)

Out[3]: numpy.ndarray

In [4]: # Get the image shape and print it out
        height, width, depth, channels = image_data.shape
        print(f"The image object has the following dimensions: height: {height}, width:{width}
```

The image object has the following dimensions: height: 240, width:240, depth:155, channels:4

As you can see these "image objects" are actually 4 dimensional! With the exploratory steps below you'll get a better sense of exactly what each of these dimensions represents.
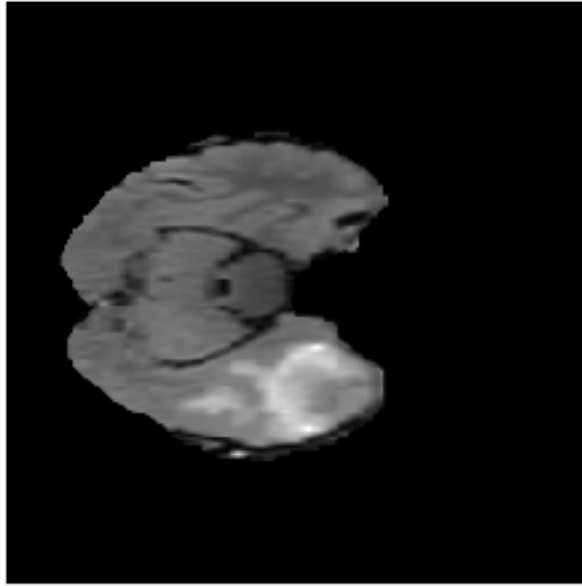
1.0.4 Visualize the data

The "depth" listed above indicates that there are 155 layers (slices through the brain) in every image object. To visualize a single layer, run the cell below. Note that if the layer is one of the first or the last (`i` near 0 or 154), you won't find much information and the screen will be dark. Run this cell multiple times to look at different layers.

The code is set up to grab a random layer but you can select a specific layer by choosing a value for `i` from 0 to 154. You can also change which channel you're looking at by changing the `channel` variable.

Keep in mind that you could just as easily look at slices of this image object along the height or width dimensions. If you wish to do so, just shift `i` to a different dimension in the `plt.imshow()` command below. Which slice direction looks the most interesting to you?

```
In [5]: # Select random layer number
        maxval = 154
        i = np.random.randint(0, maxval)
        # Define a channel to look at
        channel = 0
        print(f"Plotting Layer {i} Channel {channel} of Image")
        plt.imshow(image_data[:, :, i, channel], cmap='gray')
        plt.axis('off');
```

Plotting Layer 51 Channel 0 of Image

1.0.5 Interactive exploration

Another way to visualize this dataset is by using IPython Widgets to allow for an interactive exploration of the data.

Run the next cell to explore across different layers of the data. Move the slider to explore different layers. Change the `channel` value to explore different channels. See if you can tell which layer corresponds to the top of the brain and which corresponds to the bottom!

If you're feeling ambitious, try modifying the code below to slice along a different axis through the image object and look at other channels to see what you can discover!

```python
In [7]: # Define a function to visualize the data
        def explore_3dimage(layer):
            plt.figure(figsize=(10, 5))
            channel = 3
            plt.imshow(image_data[:, :, layer, channel], cmap='gray');
            plt.title('Explore Layers of Brain MRI', fontsize=20)
            plt.axis('off')
            return layer

        # Run the ipywidgets interact() function to explore the data
        interact(explore_3dimage, layer=(0, image_data.shape[2] - 1));

interactive(children=(IntSlider(value=77, description='layer', max=154), Output()), _dom_classe
```

1.0.6 Explore the data labels

In this section, you'll read in a new dataset containing the labels for the MRI scan you loaded above.

Run the cell below to load the labels dataset for the image object you inspected above.

```
In [8]: # Define the data path and load the data
        label_path = "./BraTS-Data/labelsTr/BRATS_001.nii.gz"
        label_obj = nib.load(label_path)
        type(label_obj)
```

```
Out[8]: nibabel.nifti1.Nifti1Image
```

1.0.7 Extract the data as a numpy array

Run the next cell to extract the data labels using the `get_fdata()` method of the image object

```
In [9]: # Extract data labels
        label_array = label_obj.get_fdata()
        type(label_array)
```

```
Out[9]: numpy.ndarray
```

```
In [10]: # Extract and print out the shape of the labels data
         height, width, depth = label_array.shape
         print(f"Dimensions of labels data array height: {height}, width: {width}, depth: {dept
         print(f'With the unique values: {np.unique(label_array)}')
         print("""Corresponding to the following label categories:
         0: for normal
         1: for edema
         2: for non-enhancing tumor
         3: for enhancing tumor""")
```

```
Dimensions of labels data array height: 240, width: 240, depth: 155
With the unique values: [0. 1. 2. 3.]
Corresponding to the following label categories:
0: for normal
1: for edema
2: for non-enhancing tumor
3: for enhancing tumor
```

1.0.8 Visualize the labels for a specific layer

Run the next cell to visualize a single layer of the labeled data. The code below is set up to show a single layer and you can set `i` to any value from 0 to 154 to look at a different layer.
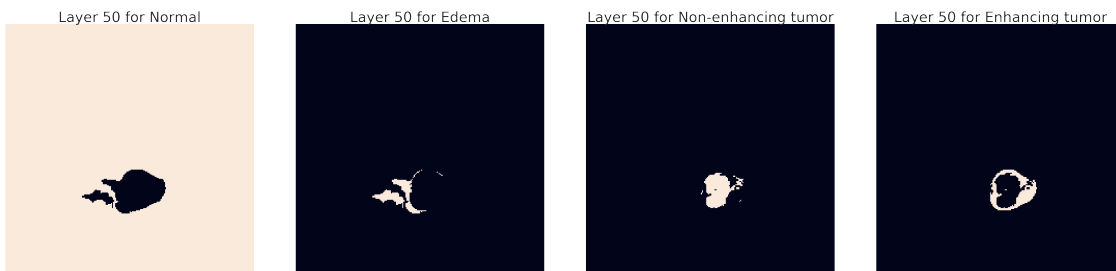
Note that if you choose a layer near 0 or 154 there might not be much to look at in the images.

```
In [11]: # Define a single layer for plotting
         layer = 50
         # Define a dictionary of class labels
         classes_dict = {
             'Normal': 0.,
```

4

```
        'Edema': 1.,
        'Non-enhancing tumor': 2.,
        'Enhancing tumor': 3.
    }
    # Set up for plotting
    fig, ax = plt.subplots(nrows=1, ncols=4, figsize=(50, 30))
    for i in range(4):
        img_label_str = list(classes_dict.keys())[i]
        img = label_array[:,:,layer]
        mask = np.where(img == classes_dict[img_label_str], 255, 0)
        ax[i].imshow(mask)
        ax[i].set_title(f"Layer {layer} for {img_label_str}", fontsize=45)
        ax[i].axis('off')
    plt.tight_layout()
```


Layer 50 for Normal   Layer 50 for Edema   Layer 50 for Non-enhancing tumor   Layer 50 for Enhancing tumor

1.0.9 Interactive visualization across layers

As another way of looking at the data, run the code below to create a visualization where you can choose the class you want to look at by clicking a button to choose a particular label and scrolling across layers using the slider!

```
In [13]: # Create button values
         select_class = ToggleButtons(
             options=['Normal','Edema', 'Non-enhancing tumor', 'Enhancing tumor'],
             description='Select Class:',
             disabled=False,
             button_style='info',

         )
         # Create layer slider
         select_layer = IntSlider(min=0, max=154, description='Select Layer', continuous_update

         # Define a function for plotting images
         def plot_image(seg_class, layer):
             print(f"Plotting {layer} Layer Label: {seg_class}")
             img_label = classes_dict[seg_class]
```

5

```python
        mask = np.where(label_array[:,:,layer] == img_label, 255, 0)
        plt.figure(figsize=(10,5))
        plt.imshow(mask, cmap='gray')
        plt.axis('off');

    # Use the interactive() tool to create the visualization
    interactive(plot_image, seg_class=select_class, layer=select_layer)

interactive(children=(ToggleButtons(button_style='info', description='Select Class:', options=
```

And there you have it! We hope this lab has helped you get a better sense of the data you'll be working with in this week's assignment.