AI4M Course 1 week 3 lecture notebook

Extract a sub-section

In the assignment you will be extracting sub-sections of the MRI data to train your network. The reason for this is that training on a full MRI scan would be too memory intensive to be practical. To extract a sub-section in the assignment, you will need to write a function to isolate a small "cube" of the data for training. This example is meant to show you how to do such an extraction for 1D arrays. In the assignment you will apply the same logic in 3D.

In [1]:
```
import numpy as np
import keras
import pandas as pd
```

```
Using TensorFlow backend.
```

In [2]:
```
# Define a simple one dimensional "image" to extract from
image = np.array([10,11,12,13,14,15])
image
```

Out[2]:
```
array([10, 11, 12, 13, 14, 15])
```

In [3]:
```
# Compute the dimensions of your "image"
image_length = image.shape[0]
image_length
```

Out[3]:
```
6
```

Sub-sections

In the assignment, you will define a "patch size" in three dimensions, that will be the size of the sub-section you want to extract. For this exercise, you only need to define a patch size in one dimension.

In [4]:
```
# Define a patch length, which will be the size of your extracted s
ub-section
patch_length = 3
```

To extract a patch of length `patch_length` you will first define an index at which to start the patch.

Run the next cell to define your start index

In [5]:
```
# Define your start index
start_i = 0
```

At the end of the next cell you are adding 1 to the start index. Run cell a few times to extract some one dimensional sub-sections from your "image"

What happens when you run into the edge of the image (when `start_index` is > 3)?

In [6]:
```
# Define an end index given your start index and patch size
print(f"start index {start_i}")
end_i = start_i + patch_length
print(f"end index {end_i}")

# Extract a sub-section from your "image"
sub_section = image[start_i: end_i]
print("output patch length: ", len(sub_section))
print("output patch array: ", sub_section)

# Add one to your start index
start_i +=1
```

```
start index 0
end index 3
output patch length:  3
output patch array:  [10 11 12]
```

You'll notice when you run the above multiple times, that eventually the sub-section returned is no longer of length `patch_length`.

In the assignment, your neural network will be expecting a particular sub-section size and will not accept inputs of other dimensions. For the start indices, you will be randomly choosing values and you need to ensure that your random number generator is set up to avoid the edges of your image object.

The next few code cells include a demonstration of how you could determine the constraints on your start index for the simple one dimensional example.

In [7]:
```
# Set your start index to 3 to extract a valid patch
start_i = 3
print(f"start index {start_i}")
end_i = start_i + patch_length
print(f"end index {end_i}")
sub_section = image[start_i: end_i]
print("output patch array: ", sub_section)
```

```
start index 3
end index 6
output patch array:  [13 14 15]
```

In [8]:
```
# Compute and print the largest valid value for start index
print(f"The largest start index for which "
      f"a sub section is still valid is "
      f"{image_length - patch_length}")
```

```
The largest start index for which a sub section is still valid is
3
```

In [9]:
```
# Compute and print the range of valid start indices
print(f"The range of valid start indices is:")

# Compute valid start indices, note the range() function excludes t
he upper bound
valid_start_i = [i for i in range(image_length - patch_length + 1)]
print(valid_start_i)
```

```
The range of valid start indices is:
[0, 1, 2, 3]
```

Random selection of start indices

In the assignment, you will need to randomly select a valid integer for the start index in each of three dimensions. The way to do this is by following the logic above to identify valid start indices and then selecting randomly from that range of valid numbers.

Run the next cell to select a valid start index for the one dimensional example

In [10]:
```
# Choose a random start index, note the np.random.randint() functio
n excludes the upper bound.
start_i = np.random.randint(image_length - patch_length + 1)
print(f"randomly selected start index {start_i}")
```

```
randomly selected start index 0
```

In [11]:
```
# Randomly select multiple start indices in a loop
for _ in range(10):
    start_i = np.random.randint(image_length - patch_length + 1)
    print(f"randomly selected start index {start_i}")
```

```
randomly selected start index 1
randomly selected start index 2
randomly selected start index 0
randomly selected start index 0
randomly selected start index 0
randomly selected start index 3
randomly selected start index 2
randomly selected start index 1
randomly selected start index 0
randomly selected start index 0
```

Background Ratio

Another thing you will be doing in the assignment is to compute the ratio of background to edema and tumorous regions. You will be provided with a file containing labels with these categories:

- 0: background
- 1: edema
- 2: non-enhancing tumor
- 3: enhancing tumor

Let's try to demonstrate this in 1-D to get some intuition on how to implement it in 3D later in the assignment.

In [12]:
```
# We first simulate input data by defining a random patch of length
16. This will contain labels
# with the categories (0 to 3) as defined above.

patch_labels = np.random.randint(0, 4, (16))
print(patch_labels)
```

```
[2 3 0 2 0 1 1 3 0 0 0 2 3 0 3 0]
```

In [13]:
```
# A straightforward approach to get the background ratio is
# to count the number of 0's and divide by the patch length

bgrd_ratio = np.count_nonzero(patch_labels == 0) / len(patch_labels
)
print("using np.count_nonzero(): ", bgrd_ratio)

bgrd_ratio = len(np.where(patch_labels == 0)[0]) / len(patch_labels
)
print("using np.where(): ", bgrd_ratio)
```

```
using np.count_nonzero():  0.4375
using np.where():  0.4375
```

In [14]:
```
# However, take note that we'll use our label array to train a neural network
# so we can opt to compute the ratio a bit later after we do some preprocessing.
# First, we convert the label's categories into one-hot format so it can be used to train the model

patch_labels_one_hot = keras.utils.to_categorical(patch_labels, num_classes=4)
print(patch_labels_one_hot)
```

```
[[0. 0. 1. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [1. 0. 0. 0.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]]
```

Note: We hardcoded the number of classes to 4 in our simple example above. In the assignment, you should take into account that the label file can have a different number of categories

In [15]:
```
# Let's convert the output to a dataframe just so we can see the la
bels more clearly

pd.DataFrame(patch_labels_one_hot, columns=['background', 'edema',
'non-enhancing tumor', 'enhancing tumor'])
```

Out[15]:

	background	edema	non-enhancing tumor	enhancing tumor
0	0.0	0.0	1.0	0.0
1	0.0	0.0	0.0	1.0
2	1.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0
4	1.0	0.0	0.0	0.0
5	0.0	1.0	0.0	0.0
6	0.0	1.0	0.0	0.0
7	0.0	0.0	0.0	1.0
8	1.0	0.0	0.0	0.0
9	1.0	0.0	0.0	0.0
10	1.0	0.0	0.0	0.0
11	0.0	0.0	1.0	0.0
12	0.0	0.0	0.0	1.0
13	1.0	0.0	0.0	0.0
14	0.0	0.0	0.0	1.0
15	1.0	0.0	0.0	0.0

In [16]:
```
# What we're interested in is the first column because that
# indicates if the element is part of the background
# In this case, 1 = background, 0 = non-background

print("background column: ", patch_labels_one_hot[:,0])
```

```
background column:  [0. 0. 1. 0. 1. 0. 0. 0. 1. 1. 1. 0. 0. 1. 0.
1.]
```

In [17]:
```
# we can compute the background ratio by counting the number of 1's
# in the said column divided by the length of the patch

bgrd_ratio = np.sum(patch_labels_one_hot[:,0])/ len(patch_labels)
print("using one-hot column: ", bgrd_ratio)
```

```
using one-hot column:  0.4375
```

That's all for this lab, now you have the basic tools you need for sub-section extraction in this week's graded assignment!