UCL Engineering
Convolutional Neural Networks Tutorial

The first part of this document contains user notes, the second is developer info including how to add new content to the app.

Table of Contents

USER NOTES:

Prerequisites

You should have already completed the Basics of Deep Learning Tutorial before starting this tutorial. If you are already familiar with Deep Learning/AI concepts, or want to research topics as they are mentioned then feel free to continue.

Installation

Download the entire ‘CNNapp’ folder to your computer.

The GUI will only work with Python up to version 3.5.4. These are the steps to get this working:

- Install Miniconda 3 from their website: https://docs.conda.io/en/latest/miniconda.html
- Load conda environment
    - **Mac:**
        - Open Terminal
        - Execute: `source opt/miniconda3/bin/activate`
            - This is how we can begin using conda
    - **Windows:**
        - Open the Anaconda Prompt (Miniconda 3)
    - **BOTH:**
    - Execute: `conda create --name tutorial python=3.5`
        - This creates the environment ‘tutorial’ with Python version 3.5
    - Execute: `conda activate tutorial`
        - This activates the environment we have just created
    - Once that is done, install tk and pillow with the command:
        - `conda install tk pillow`
        - Other modules may be installed, this is ok. Enter ‘y’ to proceed.
    - To exit the environment, enter:
        - `conda deactivate`
- **This is the environment you will use to run the tutorials.**

- The neural networks work with the latest version of python, so create another environment called ‘nn’ without the python=3.5 term:
    - `conda create --name nn`
        - Creates another environment, this time called ‘nn’
    - `conda activate nn`
        - Enter the newly created environment
    - `conda install keras pandas pillow`
        - Install all necessary modules, and their dependencies, including Python

If you encounter any errors, visit the possible errors section.

How to run app

In order to use the application:

- You must have Python installed on your computer (follow the Installation steps)
- Ensure the entire application folder is downloaded locally
- In your terminal, change the working directory to the directory in which this file is located, i.e. the CNNapp directory (example below)
- Run 'app.py' using Python (python app.py)
- If the application does not work, read on to troubleshoot or search online for the error.

You can navigate to other sections of the tutorial at the bottom of each page. You may need to move your mouse for buttons to appear.
When scrolling, you must use the scroll bar on the right of the screen. Other methods of scrolling won't work.

On Windows, you can access conda through the Anaconda Prompt for Windows.
On Mac, follow the steps in "Possible Errors" to initialise conda if it doesn't work by default in Terminal.

An example of what to enter in your Terminal/Anaconda Prompt, line by line, without the percent signs:

```
% conda activate tutorial
% cd Desktop/CNNapp
% python app.py
```

- The 1st line activates the 'tutorial' conda environment created in Installation.
- The 2nd line changes the working directory (cd) to the path specified.
    - You should change the working directory to the downloaded application folder where this file is located.
- The 3rd line runs the application.

Possible Errors

If the application does not run, diagnose the problem by searching the internet for the error. You may need to install additional modules, which you can do using:

```
conda install <module name>
```

You can install multiple modules with one command. For example:

```
conda install pillow tensorflow theano
```

On Mac, conda commands may not work immediately after install. If this is the case, you will have to activate the conda environment.

- Do this by locating the path to 'miniconda3'. This is usually from the user (home directory) and may be within 'opt'.
- Open Terminal and enter the command:

```
source opt/miniconda3/bin/activate
```

This should activate conda in the current Terminal session.
By default, Terminal opens in the user's directory.

Running Neural Networks

When running the neural network, you should change the working directory using cd to the directory containing your NN script and data.
For example, in Terminal use:

```
% conda activate nn
% cd Documents/'Neural Nets'/'1 ships ex'
% python ships_CNN.py
```

NOTE: You may not need the quotation marks if using Anaconda Prompt.

DEVELOPER NOTES:

SECTION 1: Add button to the HomePage

The content of each page is imported from the 'tcont' folder as required.
A page can be added by adding a new class, as described in section 2.
A button must then be added to the 'HomePage' class that creates an instance of the new page class.
For example, the 'Convolution Step' (the first step) button, destroys the 'HomePage' instance and creates an instance of 'ConvStep' from convstep.py:

```
# Button to create instance of Convolution Step class in HomePage
convStepButton = tk.Button(buttonFrame, text = 'Convolution Step',
        command = self.conv_step)
# Add the button to the buttonFrame
convStepButton.grid(row = 0, column = 0)

def conv_step(self):
      # destroy home frame
      self.home.destroy()

      # Create ConvStep instance from ConvStep class in master window
      # 'from convstep' for Mac, 'from tcont.convstep' for Windows
      try:
          import convstep
      except ImportError:
          import tcont.convstep as convstep
      convstep.ConvStep(self.master)
```

SECTION 2: Create and add a new page

Follow the general idea from convstep.py:

1. Create a new .py file in 'tcont' with an appropriate name
2. Import tkinter / Tkinter as tk
3. Import the 'Page' class from the page module (use Try-Except for ImportError)
4. Feed 'master' to the new class, initialise 'Page' parent class feeding 'master'
5. (recommended) Create a scrollable 'content' frame in the master by calling content = Page.scrollable_canvas(self, master)
6. Add the desired content to the 'content' frame
7. Add a link to the new page in the 'buttonFrame' of the 'HomePage', as in Section 1
8. A button to the new page should also be added to the bottombuttons.py file, similarly to how the Home button was added to 'BottomButtons' 'buttonFrame'

SECTION 3: Importing a module from tcont

Whenever an import is done, the Try-Except method should be used to ensure import is successful.
For example, when importing the page class:

```
 # 'from page' for Mac, 'from tcont.page' for Windows, depending on where
the import is being taken from
  try:
      from page import Page
  except ImportError:
      from tcont.page import Page
```

SECTION 4: Adding an image

To add an image:

- Save the image to 'tcont/images' as a GIF file type.
- Ensure the image size is appropriate by opening and resize if necessary.
- The script you are adding an image to should be located in tcont, and not in a subdirectory.
- You must have the Page class imported.

- Get the image's path:
    - `filepath = Page.load_image(self, file_name)`

- Page.load_image will return the path to the script's directory plus '/images/file_name'
- So, since the script is located in tcont, filepath returned:
  `'~/Documents/CNNapp/tcont/images/file_name'`

- If the image file is within a directory in images, this must be included, e.g.
    - `Page.load_image(self, 'shipsEx/fig2imports.gif')`
- filepath would be:
    - `'~/Documents/CNNapp/tcont/images/shipsEx/fig2imports.gif'`

- Then load as self.imageobject = tk.PhotoImage(file = filepath)
- file_name and filepath are both strings.
- imageobject should be an appropriate variable name for the image.
- Add the PhotoImage object to a Label:
    - `fig = tk.Label(content, image = self.imageobject)`
    - `fig.pack() # as desired`

NB: imageobject variable must be stored in 'self', else image will not appear when run

EXAMPLE:

```
# Figure 11: Applying preprocessing to the data
self.fig11prcssImgs = tk.PhotoImage(
    file = Page.load_image(self, 'shipsEx/fig11prcssImgs.gif')
)
fig11 = tk.Label(content, image = self.fig11prcssImgs)
fig11.pack()

fig11caption = 'Figure 11: Defining which data to apply the image '\
'pre-processing data generation to.'
fig11cap = tk.Label(content, text = fig11caption, font = self.italic)
fig11cap.pack()
```

SECTION 5: Version updates

- Clarified README.txt
- Added Retinal OCT content and References pages