Analysis of CT scans for lung cancer detection in early stages

What is this project for?

- Experimenting with approaches for processing the lung CT scans that are publicly available in the kaggle competition Data Science Bowl 2017.
- Experimenting with deep neural networks for training a model that helps early cancer detection.

How do I get set up?

1. The project is written in python. The easiest way to set up the environment is using Anaconda. Here is the link for downloading https://www.continuum.io/downloads. Python 3.x is required, the preferred version of Anaconda is the one using python 3.6.
2. Required python modules
    - **pydicom**
    - **scikit-image**
    - **scikit-learn**
    - **scipy**
    - **sklearn**
    - **pandas**
    - **imutils**
    - **tensorflow**
    - **google-api-python-client**
    - **google-cloud-storage**
    - **opencv** module for python

All of the modules instead of Tensorflow and cv2 could be easily installed using either:

```
pip install <module_name>
```

or

```
conda install <module_name>
```

Though some of the modules are recommended to be installed using conda, since Anaconda is handling some of the environment issues and provides precompiled binaries for the modules.

For example:

```
conda install scikit-image
```

Installing opencv:

```
conda install -c menpo opencv3
```

Before installing any modules, start with setting up the Anaconda environment suitable for installing the Tensorflow library.

Creating CPU Tensorflow environment:

```
conda create --name tensorflow python=3.5
activate tensorflow
conda install jupyter
conda install scipy
pip install tensorflow
```

If you are running on windows you should specify the python version to be 3.5 when creating the environment.

Creating GPU Tensorflow environment:

```
conda create --name tensorflow-gpu python=3.5
activate tensorflow-gpu
conda install jupyter
conda install scipy
pip install tensorflow-gpu
```

To switch between environments you can simply use ***deactivate*** to deactivate the current one and ***activate tensorflow-gpu*** for example, if you want to switch to the Tensorflow environment with gpu support.

In order to run Tensorflow with GPU support you must install Guda Toolkit and cuDNN.

- Installing on Windows - https://www.tensorflow.org/install/install_windows
- Installing on Ubuntu - https://www.tensorflow.org/install/install_linux
- Installing on Mac OS X - https://www.tensorflow.org/install/install_mac

After installing Tensorflow, you can simply use the requirements.txt file provided in the project. Execute the following line:

```
pip install -r requirements.txt
```

In case you are unable to install some of the modules listed in the requirements file, do remove it and try installing it using:

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

How to start data preprocessing?

To start processing the dicom files you need to run:

```
python preprocess_dicoms.py
```

Although this step is not required, since original images are too big and data preprocessing is time consuming. First stage of image preprocessing has been already executed and the data is stored in Google Cloud using several buckets:

- Baseline - https://console.cloud.google.com/storage/browser/baseline-preprocess/baseline_preprocessing/?project=lung-cancer-tests
- Morphological operations segmentation - https://console.cloud.google.com/storage/browser/segmented-lungs/segmented_morph_op/?project=lung-cancer-tests
- Waterhsed segmentation - https://console.cloud.google.com/storage/browser/segmented-lungs-watershed/segmented_watershed/?project=lung-cancer-tests

To simply download the data required for the model to be trained you need to execute:

```
python data_collector.py
```

Compressed 3D patient images will be downloaded and by default stored under ***./fetch_data*** directory.

To select which model will be trained, you need to change the value of SELECTED_MODEL in ***config.py*** (simply choose one of the predefined model names and the other configurations will be changed correspondingly).

Source and destination directories are configurable using the ***config.py***:

- ALL_IMGS points to the directory with the original dicom files for each patient (you do not need to edit it, if you have used the download script to store the preprocessed data as mentioned in the previous step).

- SEGMENTED_LUNGS_DIR points to the directory where the segmented lungs will be stored in a .npz file for each patient (compressed numpy array). The directory is configured automatically depending on the selected model to be trained.

How to start model training?

To start model training simply execute:

```
python model_train.py
```

Configuration and definitions of the layers for the CNN are described in python files located under ***model_definition***. Three configurations are currently available:

- baseline.py - Baseline configuration with three convolutional layers and two fully connected.
- additional_layers.py - Deeper network with four convolutional and three fully connected layers.
- default.py - The default configuration also has seven layers in total, but some of the filters have different sizes from those in the previous configuration.

How to evaluate stored model?

You can evaluate the results for the training set for an already stored model. Network states are stored under the directory pointed from the MODELS_STORE_DIR configuration. For each of the trained models a directory with the name of the model is created and all stored states are saved there. To evaluate the network at some of the saved states you must change the RESTORE_MODEL_CKPT to point to the checkpoint file with the desired name. Then simply execute from the command line:

```
python trained_model_loader.py
```

The solution will be stored in a csv file with location and name configured using SOLUTION_FILE_PATH, by default it is '***./solution_last.csv***'. In the command line you will also see an evaluation of the solution – confusion matrix for the test set, logarithmic loss, accuracy, sensitivity / recall and specificity. Also a csv report will be generated with the predicted results and the exact labels of the test data. The report name is constructed from the solution file name appending the prefix *report_ and* is located in the same directory as the solution file.

Other configurations

Other properties that might be configured are related with storing model states and summary exported during training.

- SUMMARIES_DIR - points to the directory where summary for the error, accuracy and sensitivity is exported during training. The data can be viewed using Tensorboard - https://www.tensorflow.org/get_started/summaries_and_tensorboard . In the selected location for the summaries you would see two directories – ***train*** and ***validation*** each containing data for metrics calculated correspondingly on the train and validation samples. You can start the board and check the graphs generated with the exported metrics summaries. For example to start the board and get the training metrics displayed, you must execute:

```
tensorboard --logdir=<SUMMARIES_DIR>/train
```

  By default the board is accessible on http://localhost:6006.

- RESTORE_MODEL_CKPT - point to the checkpoint file, you might want to resume training from or simply use for evaluating test examples with the saved state of the network (if you want to resume training set RESTORE to True and point out the START_STEP for proper counting of the epochs).