Imperial College London, MiM Lab UROP:

Using Deep Learning to Recognise White and Grey Matter from Optical Coherence Tomography Brain Scans

Jonathan Peel
Supervisor: Vani Virdyawan

July - September 2019

1 Introduction

This project is for the Mechatronics in Medicine (MiM) lab at Imperial College London and contributes to the multi-institutional EDEN2020 project. The EDEN2020 project aims to develop a flexible and steerable catheter for enhanced drug delivery in the brain. This project is a proof-of-concept and shows that deep learning neural networks can be used to identify white matter and grey matter from Optical Coherence Tomography (OCT) scans of the brain.

The project directory can be accessed via GitHub at `https://github.com/jonathan-peel/urop-project/`. This contains the OCT scan data in the form of a nested directory of bitmap (`.bmp`) images (in the `Needle OCT_Ian` folder) as well as the python program responsible for the training and testing of the model using Tensorflow (`GM_WM_or_both_classification.py`). The models (described in Section 3.1) can be downloaded from the 'Saved models' subdirectory.

2 OCT scan data

Optical Coherence Tomography (OCT) is an technique used in medical imaging. It uses the light-scattering properties of biological media (similarly to how an ultrasound uses sound) to scan approximately 3 mm into tissue with micrometer precision. Many OCT scans were taken from human brain samples consisting of White Matter (WM), Grey Matter (GM), or some combination of the two. These scans can be found in the `'Needle OCT_Ian'` directory.

An example OCT scan is shown in Figure 1. The OCT scanner used produces only one-dimensional depth scans, however to generate more data, the scanner was held stationary above a particular brain sample for 2000 seconds (approx. 33 minutes).

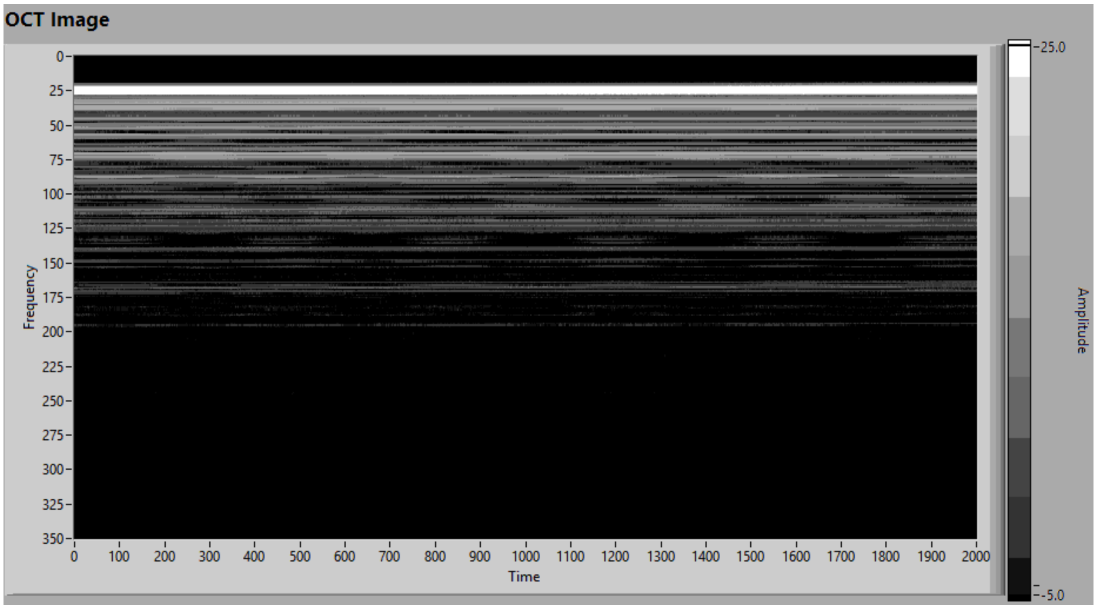

Figure 1: GM on bottom WM on top example OCT scan

This is the reason for the second, *Time*, axis. The variation with time is caused only by noise. The vertical axis, *Frequency*, scales to the depth into the sample. It is not clear from Figure 1 whether the scanned sample was white matter, grey matter, or a mixture of the two. This is the motivation for training a deep learning neural network to recognise these classes.

3 Program

This section describes the program, `GM_WM_or_both_classification.py`, which preprocesses the OCT scan data and trains a model to recognise the three classes (grey matter, white matter, and a mixture of the two).

The main steps of the program are as follows:

1. All the OCT image paths are collected into a list, each paired with its class label. There are approximately twice as many samples in the White matter and grey matter class, therefore a class_weights `dict` is instantiated to allow the model to compensate for this during training.

2. This list is shuffled and separated into training, validation, and testing lists.

3. Since each image displays the data from many one-dimensional scans alongside one another in a two-dimensional graph, the program separates each image of

each list into many so-called 'slices'. The image is cropped to exclude the axes and the other peripheral information shown in Figure 1. Note that only half of the scans in each image are used to keep the datasets from becoming too large.

4. The lists are converted into TensorFlow `dataset`s, where each element is a tuple of a slice as a numpy `array` and its class as an `integer`. These are then batched, shuffled, and prefetched accordingly.

5. A model is built in a variable way according to Section 3.1. Separate model architectures are separated into separate functions to enable each to be tested individually with ease.

6. The model was trained using the training `dataset`, and validated using the validation `dataset`, to stop the training process if the model started to simply memorise the training data.

7. The model was tested on the previously-unseen test `dataset` and given a percentage score as shown in Section 4.

Initially, the images were separated into slices before the data was separated into training, validation, and testing. However it was realised that this would result in the model over-fitting as it was tested on slices from the same scanning it was trained from. This is why the images are separated into training, validation, and testing lists in Item 2.

3.1 Model

Two model architectures were tested. Firstly, a simple Neural Network (NN) using just one densely connected layer was built as shown in Figure 2a. Secondly, a more complex Convolutional Neural Network (CNN) was built as shown in Figure 2b. The accuracy on testing for each model is shown in Table 1.

4 Results

Table 1 shows the results when two models with different architectures were trained using the training `dataset` and tested using the testing `dataset`. Note that with every training iteration, there is a degree of randomness therefore the relative accuracies in Table 1 are only indicative.

5 Discussion

Table 1 shows that a network with just one densely-connected layer of neurons was able to achieve a very high (97.4%) accuracy on the test data. This suggests that the

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 256)               105216
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 771
=================================================================
Total params: 105,987
Trainable params: 105,987
Non-trainable params: 0
_________________________________________________________________
```

(a) A simple neural network

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 1, 16)             6576
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 1, 16)             0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 1, 32)             544
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 1, 32)             0
_________________________________________________________________
dropout (Dropout)            (None, 1, 32)             0
_________________________________________________________________
flatten (Flatten)            (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 256)               8448
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 771
=================================================================
Total params: 16,339
Trainable params: 16,339
Non-trainable params: 0
_________________________________________________________________
```

(b) A convolutional neural network

Figure 2: The two variations of deep learning model architectures tested

Model architecture	Accuracy
Simple neural network	97.4%
Convolutional neural network	95.7%

Table 1: Accuracy of different model architectures

model was overfitting to the particular samples included in the training data as each individual position of each sample scan was split into hundreds of slices.

Purely dense networks are sensitive to the position of the features which they are trying to recognise. Therefore the simple network's high accuracy suggests that there was little variation in the position of the identifying elements of the white and grey matter in each sample. This is especially noticeable since the CNN had a decreased accuracy, as CNNs are insensitive to the position of recognisable features. This disparity in performance may also be explained by the fact that the CNN had many fewer parameters than the NN and contained a dropout layer which probably contributed to the versatility of the model, but not its performance with the specific samples in the training and testing datasets.

Only two variations of model were studied in this report, and these were compared using an unfair method as the program includes an inherent degree of randomness in Step 2 and 4 in Section 3, when the datasets are shuffled. To fix this, the datasets should be saved after Step 4. This may be achieved by serialising them to a `TFRecord` protobuf and storing them to file to be loaded and used by a separate program. A tutorial on how this is done may be found at `https://www.tensorflow.org/tutorials/load_data/tf_records`.

When either model was used to predict the class of unseen test data, it was able to do so within 0.3 seconds (on a mid-range laptop). Therefore it is very viable that this technology could be used to indicate to a surgeon in real-time the nature of the tissue ahead of the OCT scanner.

6 Conclusions

It is advisable to collect more data from different brain tissue specimens, and to split each single positional scan into fewer individual samples to increase the versatility of the model. Additionally, the model should be tested on OCT scans taken from completely new tissue samples to be more realistic.

Convolutional neural networks should be used to give the model positional insensitivity when recognising white matter and grey matter.

This study has shown that simple and convolutional neural networks can be used to recognise white matter, grey matter, and a mixture of the two in OCT scans of human brain samples. In the future this recognition capability could be used to build a user interface which tells a surgeon the nature and position of the tissue ahead of the OCT scanner.