01.116 Healthcare and AI Lab 1: Classification

December 2020

1 Introduction

In this lab we would be studying the processes of Retinal Optical coherence tomography (OCT), which is an imaging technique used to capture high resolution cross sections of the retina in patients. Approximately, 30 million scans are conducted yearly where each of the images have to be examined by a trained medical professional. This creates a problem , where a seemingly simple task becomes time consuming and inherently expensive.

The recent advancement in computer vision and machine learning techniques has given rise to an opportunity to automate such tedious task. In this lab , we would be exploring the option of a classification model that takes in the scanned images and output a diagnosis(prediction) for a patient.

2 OCT imagery

We assume that there are only four possible diagnosis for a OCT scan

1. Choroidal Neovascularization(CNV) - we observe a neovascular membrane and associated sub-retinal fluid
2. Diabetic macular edema (DME)- retinal start to thicken and the presence of intra-retinal fluid
3. Multiple drusen - drunsen(fatty deposit are present)
4. Normal retina (Smooth consistent imagery)

3 Preprocessing the Dataset

In this section , we will start by preprocessing the data set. Three task are fundamental when dealing with images

- Loading the images (Via a pipeline) : when dealing with larger datasets, especially images, memory becomes an essential resource as such we would like to minimise the use of memory at a given instance. A data pipeline allows us to do that.

  Useful Syntax:

  ```
  #Utilise to create a tensorflow pipeline
  data=tf.data.Dataset.list_files(.)

  #Returns a partial objection for callable function
  from functools import partial
  pf=partial(*Preprocessing_func*, target_size=height)

  #Apply function to dataset
  data.map(pf)
  ```

- Normalising the data
- Splitting our data into training and testing

__Hint:__ Notice that the OCT images are in grey scale i.e. single channel and this would be handled differently as compared to a RGB image i.e. 3 channels

Checkpoint 1: Define Training/Test data size and dimension and explain the rationale for your selection
Checkpoint 2: Show the range of data sample and explain the purpose of normalisation

4 Building the Model

Build a Convolution Neural Network that would be utilised to classify the different images into its respective classes as per section 2.

Checkpoint 3: Display the graph printout of the model
Useful Syntax: #Plot model tf.keras.utils.plot_model(.)

5 Training

Train the model with the necessary forward propagation and backward propagation. The components required for the training process would include:

- Loss function
- Optimizer

Checkpoint 4: Explain selection of loss function and Optimizer,
Checkpoint 5: Display training loss-epoch graph and the training per epoch print out. Document any abnormalities in your training and explain how you mitigated it.

Tip: Collecting section 5 and 6 checkpoints results concurrently will save you training time, but please display the solution in its respective sections

6 Validation

Validate the results using the test data that you had segmented in section 3. Validate that the model is accurate.

Checkpoint 6: Display Accuracy-epoch graph and accuracy print out. Document any abnormalities in your validation and explain how you mitigated it.

7 Building a transfer Model

Image Classification is a common task and many available trained models are readily available. During a training process, features are learnt e.g. lines and curve in the lower layers and eyes, mouths in the higher level layers which are similar for many different task. Thus we take advantage of this two components and perform a operation called *Transfer Learning*.

Transfer learning takes advantage of the reusable features extraction layer and stacks on our task-specific operations on the higher levels. Repeat the training

process and complete the following task

- Load a base model (VGG19,RESNET50,InceptionV2)
- Extend on the existing base model for fulfil classification task i.e. OCT images classification

Checkpoint 7: Define and explain the choice of transfer base model for transfer learning
Checkpoint 8: Display graph printout of the base model with the OCT image classification extension
Checkpoint 9: The training data currently being used is rather simple and as such transfer learning isn't as advantageous, however some observations can be made when comparing the first model with the second, discuss these observation.

8 Open ended Question: Bias data

The training data provided is bias i.e. The class CNV has less representation that the other class.

Checkpoint 10: Discuss why this may not be favourable and the problems it presents.
Checkpoint 11: Show some methods that can be utilised to negate or minimise these effects. Compare the accuracy and explain the pros and cons of these techniques (If any) .

Useful Syntax: #Plot model tf.keras.utils.plot_model(.)

9 Hardware/Software Guide

For this Lab we will be using Tensorflow 2.x and python 3.x , if you do not already have Tensorflow 2.0 installed, you may follow the following steps.

9.1 Setting up tensorflow

#Installing using conda
conda install -c anaconda tensorflow-gpu (***Recommended***)

#Installing using pip
pip install "tensorflow $\bar{2}$.0.0"$\#CPU$
$pipinstalltensorflow - gpu = 2.0.0 - rc1\#GPU$

9.2 Basic Tensorflow Libraries:

tf.keras.layer.xx #Handling the network hidden layers (Dense,Cov2d,Relu,tanh..)
tf.keras.optimizers.xx #Selecting optimizers