Assignment 7 – Deep Transfer Learning

This assignment will count towards your grade.

Summary

In this assignment, you will use a deep neural convolutional network to perform transfer learning from one problem domain to another

Introduction

We will be performing a classification task with the same dataset as the one used for Assignment 6. This time, you will have to compute the feature by yourself with the help of a deep neural network originally trained for another visual task.
We will use the InceptionV3 network architecture pretrained on the ImageNet dataset. This network achieves excellent performance in image recognition task with natural images and has been trained with millions of them. This allowed the network to learn excellent general purpose convolutional filters in the lower layers.
This paper describes the network architecture:
https://arxiv.org/pdf/1512.00567.pdf

This site describes the initial dataset used for training:
http://image-net.org/
For our transfer learning task, we will use again the Messidor dataset
http://www.adcis.net/en/Download-Third-Party/Messidor.html
You do not have to download anything from the website, all the images and ground truth are provided in Canvas.

Also, we will be using Keras (https://keras.io/), a high-level python-based neural network package. Keras will allow you to download the InceptionV3 network architecture and its weights with a few lines of code.

The following assignment needs to be presented as a Jupyter notebook. However, it is suggested to write as much as possible of code as an external module and use Jupyter notebooks for visualization and code description only.

We will be using the following shorthand for the libraries:

```
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
import os

import numpy as np
import skimage as sk
import skimage.io as skio

import sklearn.model_selection as le_ms
import sklearn.preprocessing as le_pr
import sklearn.linear_model as le_lm
import sklearn.metrics as le_me

import tensorflow.keras.applications.inception_v3 as ki3
```

Set-up

- Download all images and CSV file of the Messidor dataset from Canvas
- Install the h5py v. 2.10 package. This is required by Tensorflow to automatically load the network weights
    - Start the bmi6331 conda environment from the command line
    - Type: conda install h5py=2.10

Create the dataset (3 points)

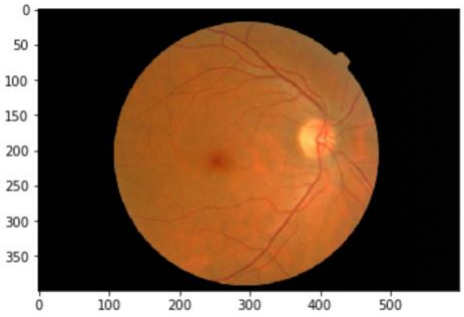
- Load ground truth

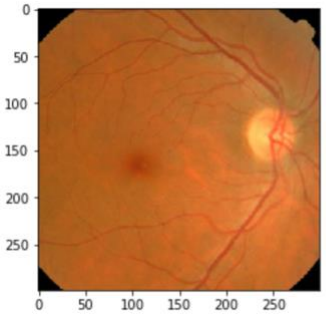
```
IMG_DIR = './imgs/'
#load data (originally from: http://www.adcis.net/en/Download-Third-Party/Messidor.html)
groundTruthFr = pd.read_csv( 'messidorGroundTruth2.csv' )

# select only samples with no retinopathy or with maximum retinopathy
lbl = (groundTruthFr['retinopathy'] == 0) | (groundTruthFr['retinopathy'] > 2)
y = (groundTruthFr[lbl]['retinopathy'].values > 0).astype(int) # target class (0: no retinopathy, 1: retinopathy)

# Filter ground truth according to labels selected
groundTruthFr = groundTruthFr[lbl]
```

- Create a function called cropImg, that loads an image, and crops it into a 299x299 square. This is needed to make it compatible with the current deep learning implementation that we are going to use. You should keep the image in RGB space. Example of the result before and after cropping:





- The crop function should also return the image with a floating point precision (check sk.img_as_float) with a histogram range from 0 to 1
- Now we will crop each image of the dataset and load all of them into a tensor (i.e. numpy array). The tensor will have the following dimensionality: 801 x 299 x 299 x 3. In other words: number of images x height x width x number of color planes. This will require around 1.5GB of RAM. If your machine does not allow that you can load the dataset into smaller chunks (i.e. mini batches).
    - Suggestion: you iterate through the dataset using something like:

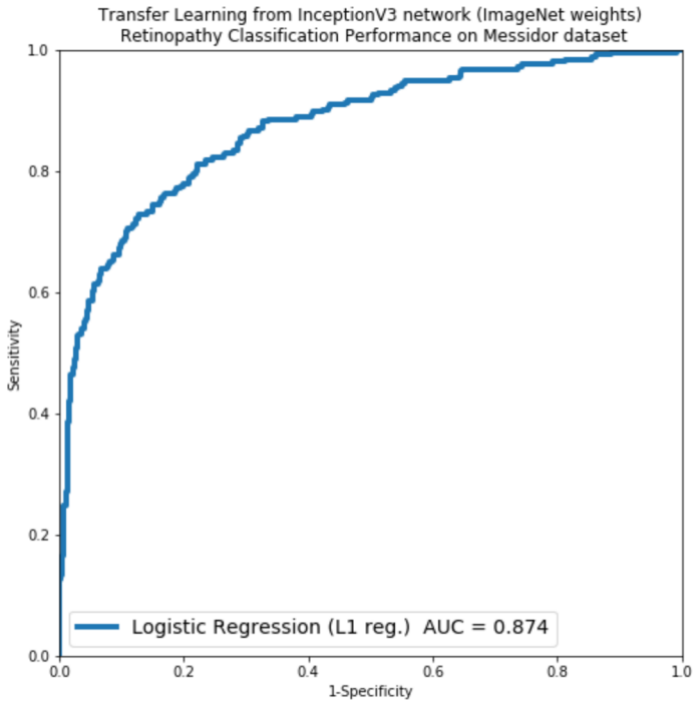
```
for i in range(len( groundTruthFr )):
    imgFile = IMG_DIR + groundTruthFr.image.iloc[i]
```

Transfer Learning as a feature extractor (7 points)

- Load the InceptionV3 model without including the top classification layer, adding a final global average pooling layer and using the imagenet weights. This can be done with a single function call to ki3.InceptionV3
- Print the model architecture with the summary() method of the model
- Preprocess the dataset tensor to make it compatible with the model using the following code:

```
# pre-process vector make sure the vector
# (i.e. scale with x /= 255. ; x -= 0.5;  x *= 2.)
X = X*255
X = ki3.preprocess_input(X)
```

- Create the feature matrix using the predict method of the InceptionV3 model. You should be obtaining a matrix with the following shape: 801 X 2048 (number of samples X number of features from the last average pooling layer of the network).
- Using what you learned in assignment 6, train and test a linear classifier to classify images with retinopathy vs. images without retinopathy with the feature generated from the InceptionV3 model. Use 10 fold cross validation (hyper-parameter tuning is optional).
- Plot the ROC curve and area under the ROC curve.
- This is an example of what we have obtained with a Logistic Regression classifier with L1 regularization.



Bonus tasks - Transfer Learning as fine tuning

- In the previous task, you have not really trained the InceptionV3 model, just used it as a feature extractor to train a separate classifier. Here you will have to perform transfer learning by training the network itself. (3 bonus points)
    - Load the InceptionV3 model without including the top classification layer, adding a final global average pooling layer and at least one additional trainable dense layer
    - Freeze all the pretrained weights and retrain the new top layers network as described in https://www.tensorflow.org/tutorials/images/transfer_learning#create_the_base_model_from_the_pre-trained_convnets with a binary cross entropy loss and Adam as optimizer
    - Plot ROC curves and visualize loss curves either from the code itself or using Tensorboard: https://www.tensorflow.org/tensorboard
    - Normally, you would want to proceed to unfreeze the weights in all layers and fine tune the network as a whole, you can skip this step
- In all previous tasks, the optimization training loops were hidden in custom objects. Here you will use explicit training loops in Tensorflow which will allow you much easier customization (2 bonus points)
    - Repeat the previous “transfer learning as fine tuning” process using a training loop. You have an example here:

https://www.tensorflow.org/guide/keras/writing_a_training_loop_from_scratch/

- Every N iterations (the choice of N is yours), output the content of a convolutional filters of your choice. This can be done by mylayer = your_model.get_layer('layername') function to find a reference to a layer, and then inspecting the properties of the mylayer object

Submission

- Now zip the folder containing your code and the notebook file as you did in assignment 01
- Upload yourname-assignment07-nb.html and yourname-assignment07-src.zip to canvas before the deadline