AI-F19
Assignment 5 (A5) - Deep Learning Techniques

General Info
10 Point Individual Assignment Due by 2 pm Friday December 13th. However we will not include any late penalties until 2pm Monday December 16th. A 10% penalty will apply until 2pm Tuesday December 17th. **Submissions will not be accepted beyond this point since we will be finalizing the grades.**

Learning Objective
This assignment satisfies learning objective 2 (LO2) as specified in the syllabus. You will apply conceptual knowledge of core AI concepts by implementing AI algorithms, analyzing existing intelligent systems (including humans), and using existing AI tools to solve problems.

Getting Help
We want these projects to be rewarding and instructional, not frustrating and demoralizing. But, we don't know when or how to help unless you ask. If you find yourself stuck on something, contact us via Piazza or come by the office hours. If you can't make our office hours, let us know and we will be happy to schedule alternate times.

Submission
You should submit the deliverables on Gradescope under Assignment 5. If you encounter any difficulties during the submission process please contact us via email and also include a copy of your submission files before the deadline.

Deliverables

You will need to submit the following files:

- A5.py
- <Identikey>_A5_report.pdf
- **DO NOT submit** the images used

Instructions - MUST READ BEFORE GETTING STARTED

- You may email us your submission at any time to account for upload difficulties to Gradescope and we can mark your submission “as-is” and timestamped with the email receipt time.
- **This assignment will not have an autograding component.** We will still be running your code however, so be sure that whatever you submit should compile and run.
- For both parts, we have provided initial code. Please complete all questions within these files. **Do not submit iPython notebooks.**

Prerequisites
For this assignment you will need some additional python libraries. You can install these using pip. More information can be found on their respective manual pages.

```
pip install matplotlib, scikit-learn, pandas, pillow, tensorflow, keras
```

Installing tensorflow can be tricky depending on your setup, but there is a lot of support on their website to get you up and running

Overview

In this assignment, you will learn to work with keras and Google's TensorFlow framework to build a neural network-based face recognition system. These are the state of the art tools for neural network models. You will consider both standard approaches as well as deep learning approaches using convolutional neural networks.

You will work with a subset of the FaceScrub dataset (available under a CC license). A subset of male actors is here and a subset of female actors is here (also included in the assignment documents). The dataset consists of URLs of images with faces, as well as the bounding boxes of the faces. The format of the bounding boxes is as follows (from the FaceScrub readme.txt file):

> *The format is x1,y1,x2,y2, where (x1,y1) is the coordinate of the top-left corner of the bounding box and (x2,y2) is that of the bottom-right corner, with (0,0) as the top-left corner of the image. Assuming the image is represented as a Python NumPy array M, a face in M can be obtained as M[y1:y2, x1:x2].*

Part 1 - Data Preprocessing - 2pts

We have provided the code to pull each of the images that we will be using. To make sure your models run in a reasonable time, we have limited the assignment to six actors. To pull the data run:

```
python getData.py
```

You will now need to preprocess each of these images before you can start. We have provided two skeleton functions as described below. **Hint**: for both of these functions the Python image library (PIL) will be very useful.

- **cropImage**
  - Parameters: Image, face bounding box as parameters
  - Return: Cropped image based on the bounding box provided
- **standardizeImage**
  - Parameters: Image, required size
  - Return: Resized image

In the function **preProcessImages** you should first crop all the images and save them to a separate folder. You should then standardize each of the cropped images to be 60 x 60. The final dataset should be 60x60 images each depicting only a face. To save time later, you may also wish to save these images so that you don't have to repeat the process, but be sure not to overwrite earlier versions of your images.

In Your Report:
Describe the dataset of faces. In particular, provide at least three examples of the images in the dataset as well as at least three examples of cropped out faces.

A good description will:

- Comment on the quality of the dataset: are the provided bounding boxes accurate?

- Explain why the preprocessing is important in the context of classifying faces
- Explain the effect of resizing on the images

Part 2 – Deep Learning Framework - 3pts

We have provided sample code that trains a neural network on the MNIST dataset using the keras library. Remember, MNIST focuses on classifying handwritten digits. You will now adapt this code to build a face recognition network in keras. Your new network will take the 3,600 (60x60) pixels of your image as inputs and output a classification of one of the six actors. Thus, it uses raw pixel encodes as features.

Your images currently have three channels (RGB). To make things a little easier, first convert each image to greyscale so that you only have one channel per image. The code snippet below may help you.

```
from PIL import Image
img = Image.open('image.png').convert('L')
```

To make the network more interpretable, you should also scale all input to be between 0 and 1, you can do this by dividing your matrix by 255.

Using the sample code as an example, complete the function **trainFaceClassifier (**described below). The sample code gives you much of the setup, but you will need to adjust both the input and output layers relative to the new dataset.

Use a fully-connected neural network with a single hidden layer. You may choose the number of nodes in your hidden layer based on experimentation, but list this in your report.

Before you start training your model, you will need to split the data into training, validation and test data (**this is not the case in the sample code**). Initially the validation data is just used by Keras in the backend, you may wish to use it more extensively in Part 4.

- **trainFaceClassifier**
  - Parameters**:** Preprocessed data, data labels
  - Returns: Trained model

Note: The sample code is provided to help you, but that does not mean you can just copy and paste. You should take time to make sure you understand each element of the sample code and what it is doing. Spending time reviewing this code before implementing your own will save you a lot of trouble shooting!

In your Report:

- Plot the loss graphs by epoch (as shown in the class slides) This should have two lines one for the training loss and one for the validation loss.
- Evaluate your final model on the test set.
- Include a description of your model. In particular, describe how you separated the data into training, text, and validation sets, how you preprocessed the inputs and initialized the weights, what activation function you used, and how many nodes in the hidden layer you selected.

Part 3 - Transfer Learning - 2pts

VGG16 (also called OxfordNet) is a convolutional neural network architecture named after the Visual Geometry Group from Oxford, who developed it. It was used to win the ILSVR (ImageNet) competition in 2014. To this day it is still considered to be an excellent vision model, although it has been somewhat outperformed by more recent advances such as Inception and ResNet. We will use the VGG16 to help us derive new features for transfer learning.

We have provided the function **getVGGFeatures**, which takes as input an image file and the name of a layer and returns the activations of that layer. Details about each layer of the network can be found here.

You will now complete the function **trainFaceClassifier_VGG (**described below). Build a new network using the activations from VGG16 as features. You may choose which layer of VGG16 you draw from. We recommend either "`block4_pool`" or "`block5_pool`". You may wish to experiment with a few. **You should be able to reuse a lot of the code from part 2**.

- **trainFaceClassifier_VGG**
    - Parameters**:** Data extracted from VGG16, data labels
    - Returns: Trained model

In Your Report

- Explain your choice of VGG16 layer used as input.
- Compare the performance of your new network which used features from VGG6 to your network using the raw images from part 2. Is it better or worse? Why do you think this is?

Part 4 - Experimentation – 3pts

Now that you have the frameworks in place, we can now experiment.

Start with your best performing network from parts 2 and 4, and then experiment with:

- Dropout
- Number of hidden layers
- Number of nodes in each hidden layer

This is the final part of the final assignment and is open to your creativity. We are excited to see your work! You may experiment with additional network features if you wish.

In your report:

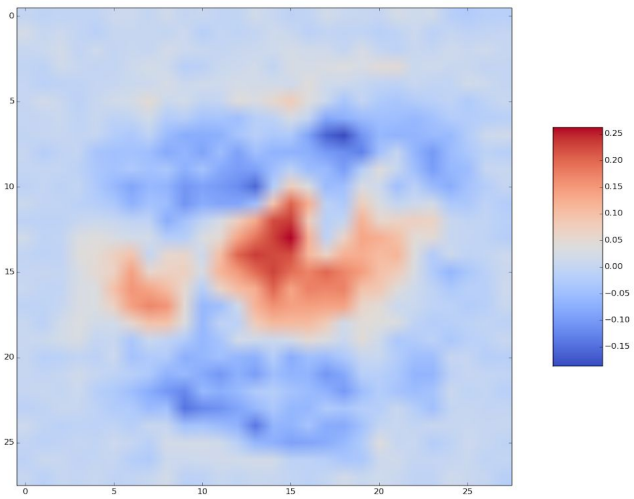
- Write a 1-2 page summary of your experiments, including the parameter settings that produced the best strategy.
- You should discuss how efficient the model is as well as the variation between experiments.
- Be sure to use evidence (screenshots, graphs, etc) to back your claims.
- Compare the best models from these experiments to performance in parts 2 and 4 – if you saw an improvement, why do you think that is? Likewise if you were not able to get an improvement, why do you think this was the case?

Part 5 - Extra Credit - Visualizations – 1pt available

We can visualize the weights of the networks that classify digits and faces in order to gain insight into how they work.

Because the network is fully connected (each input is linked to each node in the hidden layer), for each node in the hidden layer we can retrieve the weights for each of the inputs.

For a one-hidden-layer neural network, you can visualize the weights that connect the input layer to each hidden unit. For example, in the MNIST network developed in the sample code, the weights of that connect the input layer to **one of the hidden units** can be visualized as follows:



The weights connecting the hidden unit visualized above to the output units that correspond to the digits 0..9 are: [-0.17553222, 0.09433381, -0.75548565, 0.13704424, 0.17520368, -0.02166308, 0.15751396, -0.31243968, 0.12079471, 0.66215879] .

Based on the weights connecting this node to the output units (listed above), a positive output from this node indicates a 9 (weight 0.66) whereas a negative output from this node indicates a 2 (weight -0.76). We can verify this in the image above. If you were to draw a 9 in the box in the figure above, it would likely pass through some of the red areas.

The matplotlib code used for the figure is as follows

```
# Code for displaying a feature from the weight matrix nodeInputWeights

fig = figure(1)
ax = fig.gca()
heatmap = ax.imshow(nodeInputWeights.reshape((28,28)), cmap = cm.coolwarm)
fig.colorbar(heatmap, shrink = 0.5, aspect=5)
show()
```

Using the model trained in Part 2, select two actors. Using the weights connecting the hidden nodes to the output layer, select two hidden nodes that are useful for classifying input photos as those particular actors (nodes that have highly weighted connections to the output layer).

For each of the nodes selected, visualize the input weights as done in the figure.

In Your Report

- Include your visualizations. Label your visualizations with the name of the actor, and display any other relevant information.
- Explain how you selected the hidden nodes
- Give a brief description (2-3 sentences) of any insights provided by the visualisations

APPENDIX 1 : Scoring Rubric

The scoring rubric **for your report** is based on the Kentucky General Scoring Rubric from the Kentucky Department of Education (KDE).

Score	Description
Category 4 (Score 90%-100%)	● The student completes all important components of the task and communicates ideas clearly. ● The student demonstrates in-depth understanding of the relevant concepts and/or process. ● Where appropriate, the student chooses more efficient and/or sophisticated processes. ● Where appropriate, the student offers insightful interpretations or extensions (generalizations, applications, analogies).
Category 3 (Score 70%-90%)	● The student completes most important components of the task and communicates clearly. ● The student demonstrates an understanding of major concepts even though he/she overlooks or misunderstands some less important ideas or details.
Category 2 (Score 60%-70%)	● The student completes some important components of the task and communicates those clearly. ● The student demonstrates that there are gaps in his/her conceptual understanding.
Category 1 (Score 10%-60%)	● The student shows minimal understanding. ● The student addresses only a small portion of the required task(s).
Category 0 (Score 0)	● Response is totally incorrect or irrelevant.
Blank (Score 0)	● No response.