Coursework Assignment

1 Assignment Overview

This assignment will involve you designing, building, testing and critiquing a system for performing face alignment, aka. locating facial landmarks in images. There is also a secondary extension task detailed below.

This assignment is worth 80% of the grade for this module. It is designed to ensure you can demonstrate achieving the learning outcomes for this module, which are:

- Write and document a computer program to extract useful information from image data.
- Propose designs for simple computer vision systems.
- Determine the applicability of a variety of computer vision techniques to practical problems.
- Describe and recognise the effects of a variety of image processing operations.

1.1 Secondary Task

You will design and implement a system for modifying the colour of the lips and/or the eyes in the image. This could be achieved through segmentation, purely using the estimated landmarks. or more advanced means (e.g. generative adversarial networks). This aspect is worth 25% of the marks for this assignment.

2 What to hand in?

1. A report that comprises a *maximum* of 8 pages and 1500 words, including captions but excluding references. I'm expecting several pictures, diagrams, flowcharts and charts to be included.
    - A summary and justification for all the steps in your face alignment system, including preprocessing, choice of image features and prediction model. Explaining diagrammatically is very welcome.
    - Results of your experiments: This should include some discussion of qualitative (example based) and quantitative (number based) comparisons between different approaches that you have experimented with.
    - Qualitative examples of your face alignment approach running on the small set of provided example images, found in the compressed numpy file (examples.npz) here.

- Examples of failure cases in the face alignment system and a critical analysis of these, identifying potential causes, biases and solutions.
- A summary of how your lip/eye colour modification system works with several example results.

2. A .csv file that contains the face landmark positions on the **test** set of images, found in the compressed numpy file ("test_images".npz) here. You **must** use the provided "save_as_csv" function in the colab worksheet to process an array of shape (number_test_image, number_points, 2) to a csv file. Please make sure you run this on the right data and submit in the correct format to avoid losing marks.

3. Either .ipynb files or .py files containing annotated code for all data preprocessing, model training and testing.

4. You may optionally include your trained model parameters, but please **do not** hand in any other additional files, datasets or supplementary results as this complicates the marking process .

3 How will this be graded?

The breakdown of marks for this assignment are given below:

20 Marks **Accuracy and robustness of face alignment**
These marks are allocated based on the performance of the face alignment method. This will be evaluated on the held out test set, which includes some difficult cases. The test images, without annotations are provided in the compressed numpy file (test_images.npz) here and the error on the predicted points will be calculated after submission. Marks will be awarded for average accuracy and robustness (% of images with error below a certain threshold).

30 Marks **Outline of methods employed**
Justifying and explaining design decisions for the landmark finding. This does not have to be in depth, and I do not expect you to regurgitate the contents of the lecture notes/papers. You should state clearly:

- what methods you have used, with what parameters and why.
- what image features you have used, briefly describe how they were calculated, and why you chose them.
- any image pre-processing steps you have used, and why.

For top marks, you should clearly demonstrate a creative and methodical approach for designing your system, drawing ideas from different sources and critically evaluating your choices. Explaining using diagrams and/or flowcharts is very welcome.

20 Marks **Analysing results and failure cases**
Critically evaluate the results produced by your system on test/validation data. You should include quantitative (number based) and qualitative (example based) comparisons between different approaches that you have tried. (on the validation set).

Quantitative measures including measuring the cumulative error distribution (see lecture slides) or using boxplots or other plots to compare methods. Please note that we are interested in your final prediction results, rather than how the cost function changes during training.

A detailed qualitative analysis would investigate and identify systematic failure cases and biases, providing visual examples, and proposing potential solutions.

25 Marks **Lip/Eye colour modification**
Outline the employed methodology, ideally using a diagram or flowchart to explain the steps. Provide qualitative results for editing several images using different colours, illustrate and discuss both success and failure cases. The solutions for this section do not need to be complicated, but they should be clearly explained and appropriate for the task. Marks will be allocated for the realism of the results, the elegance of the solution and the quality of the method explanation.

5 Marks **Code annotation** is for annotating sections of the training/testing code with what they do. To get maximum marks, explain each algorithmic step (not necessarily each line) in your notebook/.py files.

General Points on the report

- Read things! Provide references to anything you find useful. You can take figures from other works as long as you reference them appropriately.
- Diagrams, flowcharts and pictures are very welcome, make sure you label them properly and refer to them from the text.
- All plots should have correctly labelled axis.

4 What resources are provided for me?

The training images are provided for you in a compressed Python array. They have already been preprocessed to be the same size with the faces roughly in the middle of the image and the eye corners in the same position. The training data can be downloaded as a compressed numpy file (training_images.npz) here.

The data can be read by:

```
import numpy as np

# Load the data using np.load
data = np.load('training_images.npz', allow_pickle=True)

# Extract the images: shape = (2811, 242, 242, 3)
images = data['images']
# and the data points: shape =  (2811, 32, 2)
pts = data['points']
```

In this very basic colab worksheet I provide code for:

- Loading the data

- Estimating the error between predictions and ground truth.
- Visualising points on an image
- Saving the results to a .csv file, which contains some checks to make sure you're predicting on the correct dataset.

A set of test images, without landmarks is provided in the compressed numpy array (test_images.npz) here. This data is loaded the same way as before, but there are no points stored in the file.

I also include 6 images to use for qualitative comparisons found in the compressed numpy array (examples.npz) here. These images should be included in your report to demonstrate face alignment performance across different genders, ethnicities and poses.

4.1 Notes on using Colab

Either you can complete this project using Google colab, which gives you a few hours of computing time completely free of charge, or you can use your personal/lab machine. If you are using Google colab, try and familiarise yourself with some of it's useful features. To keep your saved models, preprocessed data etc. you can save it to Google drive following the instructions here. You can also directly download a file you make in colab using the code below:

```
from google.colab import files
files.download(filename)
```

If you're refactor code into extra .py files, these should be stored in your google drive as well, or on Box such that they are easy to load into your Colab worksheet.

4.2 Most important links

Contents	**filetype**	**links**
Training images and points	compressed numpy array (training_images.npz)	link or link
Test images	compressed numpy file (test_images.npz)	link or link
Examples images for qualitative comparisons	compressed numpy file (examples.npz)	link or link
Colab worksheet with some useful functions	colab worksheet	link

4.3 What library functionality can I use?

You're free to use fundamental components and functions from libraries such as OpenCV, numpy, scipy, scikitlearn to solve this assignment, although you don't have to. Here, fundamental components refers to things like regression/classification models and pre-processing/feature extraction steps and other basic functionality. What you are **not** allowed to use are library functions that have been written to directly solve the tasks you have been given, i.e. face alignment. You **cannot use the dlib face alignment tool**. Also, face detection is **not** required on this data.

In terms of tools and frameworks, it's absolutely fine to use convolutional neural networks (CNNs) if you want to, which are introduced in fundamentals of machine learning. The best packages would be either TensorFlow (probably with Keras) or PyTorch. If you use such an approach you should be sure to document how you chose the architecture and loss functions.

A well justified and high performing CNN approach will receive equivalently high marks as if you'd built it any other way.

In terms of sourcing additional labelled data, this is **not** allowed for this assignment. This is because in real-world commercial projects you will typically have a finite dataset, and even if there are possibly useful public datasets available, their license normally prohibits commercial use. On the other hand data augmentation, which effectively synthesises additional training examples from the labelled data that you have, is highly encouraged. If you use this, please try and add some text or a flow-chart of this process in your report.

5 Where do I start

5.1 Face Alignment

Face alignment is covered in lecture 14, so that's a good place to look for information. I briefly discussed the assignment at the end of the lecture, which you can listen to on Canvas. I've also included some references below.

I have included a very basic colab worksheet illustrating how to load the data and visualise the points on the face.

The simplest approach would be to treat this as either a regular or a cascaded **regression problem**, where given an image you want to predict the set of continuous landmark coordinate locations. To follow this approach you will need to consider what image features are helpful to predict the landmarks and what pre-processing is required on the data. Although you could directly use the flattened image as input, this will not be the optimal data representation for this task.

A better representation would be to describe a set of locations, either evenly spaced across the image, or in some more useful pattern (think about where in the image you might want to calculate more informations) using a feature descriptor, such as SIFT. These descriptions can then be concatenated together and used as input into a linear regression model. Note that you **do not** need to use the keypoint detection process for this task - rather the descriptors should be computed at defined locations (hint: look at sift.compute() or similar) to create a representation of the image that is comparable across the dataset.

You're not restricted to taking this approach, and for higher marks creativity is very much encouraged. Face alignment has seen a lot of interesting and varied ideas, and if you find some good ideas while reading around the topic that would be great.

5.2 Lips/Eye colour modification

We're looking for simple solutions for this task, which could be based on the landmarks you are predicting and/or colour. One approach would be to segment the required pixels and then modify the colour within the segmented region, although you could investigate other solutions. I am intentionally not providing a training set of data for this task. There's some useful code in OpenCV, take a look at cv2.fillPoly.

6 Top Tips for Success

- Remember Occam's razor, complexity should not be added unnecessarily. The more complicated your system the more things to explain/justify etc.

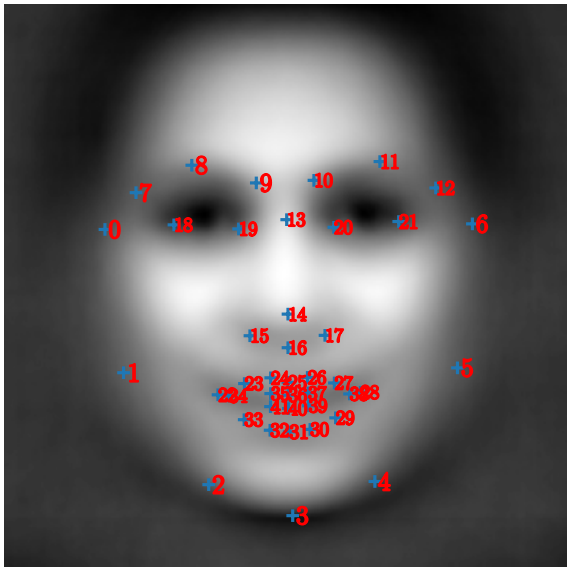

Figure 1: Illustration of the **0-indexed** (counting from 0 as you would in Python) locations of the points on the face. For example, if we wanted to find the tip of the nose, that's index 14 so we would look up points[14,:], which would give you the x and y coordinate of the tip of the nose.

- Start with a simple achievable goal and use that as a baseline to test against. Keep track of early models/results to use as points of comparison.
- Remember that even if it doesn't work well, having a go at the extension tasks is worth a few marks. We're only looking for simple solutions.
- You don't need to work at very high resolution to get accurate results. Particularly when doing initial tests, resize your images to a lower resolution images. Make sure you also transform your training points so they are in the same geometry as the image. For your predicted points, make sure these are all at the same resolution as the original images.
- Think about things that you've learned about in FML as well as Computer Vision. Dimensionality reduction could be helpful. Overfitting and outliers may be an issue, and you should consider using methods to minimise this.

7 Further reading

Face alignment is a reasonably well researched field, and a wide variety of methods have been proposed. Some relatively recent approaches are documented below. [1] is probably a good one to look at, [2] contains a survey of methods, which might give you some ideas and [3] describes the results of a competition. The other references are very much optional reading on other popular recent methods.

References

[1] Xiong X, De la Torre F. Supervised descent method and its applications to face alignment. InProceedings of the IEEE conference on computer vision and pattern recognition 2013 (pp. 532-539). Paper link.

[2] Learned-Miller E, Huang GB, RoyChowdhury A, Li H, Hua G. Labeled faces in the wild: A survey. InAdvances in face detection and facial image analysis 2016 (pp. 189-248). Springer, Cham. Paper link.

[3] Sagonas C, Antonakos E, Tzimiropoulos G, Zafeiriou S, Pantic M. 300 faces in-the-wild challenge: Database and results. Image and vision computing. 2016 Mar 1;47:3-18. Paper link.

[4] Cao X, Wei Y, Wen F, Sun J. Face alignment by explicit shape regression. International Journal of Computer Vision. 2014 Apr 1;107(2):177-90. Paper link.

[5] Burgos-Artizzu XP, Perona P, Dollár P. Robust face landmark estimation under occlusion. InProceedings of the IEEE international conference on computer vision 2013 (pp. 1513-1520). Paper link

[6] Zhu S, Li C, Change Loy C, Tang X. Face alignment by coarse-to-fine shape searching. InProceedings of the IEEE conference on computer vision and pattern recognition 2015 (pp. 4998-5006). Paper link