Final project
(Alg. 2016/2017)

Logistics

Team size

Three students. Use link below to report your group's constituency.

Grade

Final project grade for each student = group's grade + individual presentation grade

Group's grade = code's performance on benchmark (see Evaluation Score section) + group's report.

Individual performance grade = performance and contribution during live presentation.

Group's report

A *doc* file with no more than 1500 words and 3 pages describing the main ideas used in your project and why you choose them. It is to be submitted one per group.

Live presentation

Your group will have to present to present their work. Each presentation is 26min total.

You have 5 minutes as a group to present project's ideas and explain your individual contributions.

Then each one of you will present individually and alone for 7min.

During individual presentations you can be asked about anything. Things about the project or topics covered in class or homework. Please be well prepared.

Dates

Final code submission deadline = 11th December.

Submission

Your submission should be made using CANVAS in the “final project” assignment.

Your submission should be a *zip* file. Inside this *zip* file there should be three files. One *doc* file with your report (no more than 1500 words and 3 pages). One *zip* file with all your (well documented and functional code). One *pdf* file with the slides used during group presentation.

Each member of each group should make exactly the same submission. So there will be three copies of each project on CANVAS in the end.

Presentation date = 26min per group any day from the 12th to the 15th of December.

Choose your time slot and your group using the following link.

http://doodle.com/poll/2g5u58pphcs69z77

Project description

You have to develop an algorithm that given a query image finds the “closest” entries to it on a dataset of images.

Input 1

The path to a folder with images. All images files are of the same format, size and black and white. They are represented by a matrix of zeros and ones in space-separated format. They represent “meaningful shapes”.

For example, the folder

/myproject/input1

might have three pictures files

asdfsd
aaa342
rdsddf

where the file *asdfsd*, and all other files, are of the form

```
0 0 1 1
1 0 1 0
0 1 1 1
1 1 1 1
1 1 1 0
```

Input 2

The path to another folder of images. This folder and images are in the same format as above.

Input 3

The path to a file where the output will be stored.

Input 4

The number, *K*, of close entries to return. *K* must be either 1, 2, 3, etc.

Output

A list of lines where each line is of the following from

queryfilename databasefilename1 databasefilename2 databasefilename3 ... databasefilenameK

And the meaning of each line is the following.

The approximate closest *K* images to the input file *queryfilename* are the following *K* files *databasefilename1 databasefilename2 databasefilename3 … databasefilenameK*.

Calling your algorithm

<u>Your final solution must be an executable that can be called from the command line in Linux running on an Intel processor.</u>

Calling your program should work exactly as follows (e.g. for *K* = 10)

```
mylinuxcomputer:~ user$ finalproject   /inputdirectory1 /inputdirectory2
/outputdirectory 10
```

Evaluation score

1) Your solution will be evaluated based on how its run-time scales with the size of the input dataset, the number of input queries and the value of *K*.

2) It will also be evaluated on how its memory usage scale with the input.

3) Finally, your solution will also be evaluated based on its accuracy to solve a classification task. Let us be more specific about this. We will run your final code on a folder with a dataset of images and another folder with a several query images. The images are just like described above. We will run your code on these input folders for a value of K = 9. These images fall into several category. We know the categories in which each image falls but you do not.
We will read the output file produced by your program where you stored the K files closest to each query. We will read it line by line. If the category that most frequently appears on these K files is the same category as the input file, then you get one point. Otherwise you get zero points. This will produce a total score. We will then run your code, just like described above, also for K = 1, 3, 5, 7. Your final accuracy score will the highest score you get among the values of K tested.

Notice that, since the performance of your algorithm will be evaluated on an image classification task you need to be careful such that when you say "Image 1 is close to image 2" you have a high chance of having "image 1 is in the same category as image 2". For example, an image and a rotated or scaled version of the same image should be on the

Important points

There are, of course, many things about the project that are not well and fully specified. This is on purpose. It is your responsibility to make a design choice and to make your algorithm as robust as possible to whatever it might find.

You can tune these decisions to a particular evaluation dataset to gain an advantage over other groups on that dataset. However, if you overspecialize, you might not do as well if it turns out the dataset used for the final evaluation is not similar to the one you used.

If you have questions, you can come to office hours and ask them in public. But be careful not to give ideas to other groups that will be competing with you.

You will more points during your presentation if you can show interesting applications of your algorithm to other datasets, potentially not even images. Therefore, try to write your code in a modular way and as generic as possible such that it can be easily adapted to different datasets.

Also, you will get more points if you can include in your presentation some theoretical analysis of your algorithms' performance.

You are free to use ideas (not code) you find in other sources. However, you must cite them appropriately in your report and during your presentation. You **cannot** use any code or libraries you find online. If you find an algorithm or library that you find interesting and useful, you must code it yourself, cite it, and be ready to answer questions about it. This excludes things like libraries that read images from files, write images to files, etc. If you have questions of whether using a certain piece of code as-is is valid or not, please come to office hours and ask me about it. The main idea, of course, is that you learn how to think algorithmically and learn the interplay between ideas and actually implementations. Anything that shortcut this process is not allowed. Anything that hinders this process is allowed to be shortcut. Finally, using someone's ideas is not as impressive and might not translate into a good grade for your presentation and report.

Sometimes, a not so good algorithm implemented in the correct way can be better than the correct algorithm implemented in the wrong way. You will thus have a great advantage over other groups if, for example, you use C and implement things as efficiently as possible. A good strategy might be to first use an "easy" language to prototype different ideas and see how they work. Closer to the end of the semester, you can then spend a few weeks porting your code to a faster language.