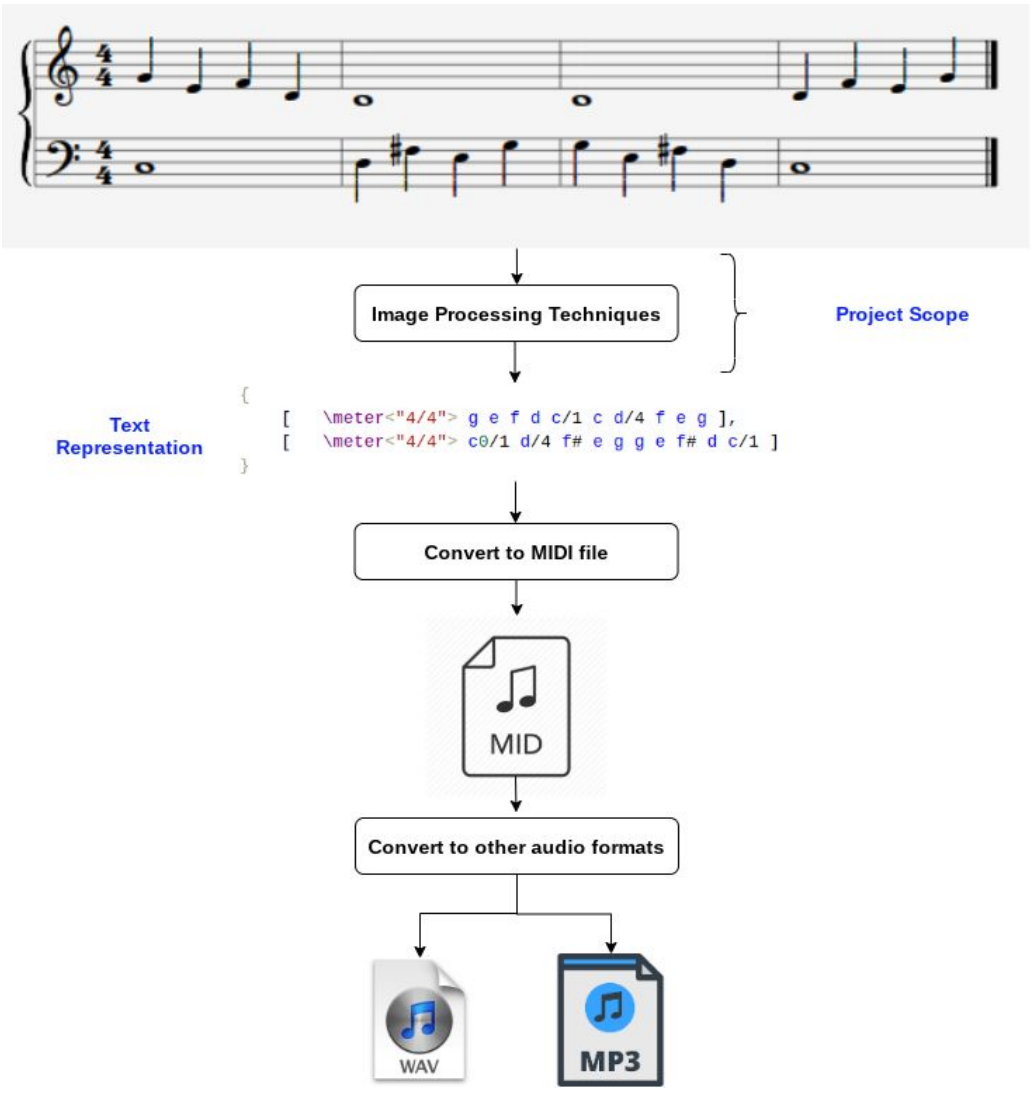
# Sheet Music Reader

## Objectives

- Apply concepts of image processing learnt in lectures and labs to a real-life problem.
- Enhance your research skills by examining the problem.
- Be introduced to new topics and practical problems.
- Learn how to build easily-deployable code.
- Work as a team in Image Processing problem starting from research to prototype.

## Project description

The aim of this project is to develop a sheet music reader. This field is called Optical Music Recognition (OMR). Its objective is to convert sheet music to a machine-readable version. We take a simplified version where we convert an image of sheet music to a textual representation that can be further processed to produce midi files or audio files like wav or mp3.

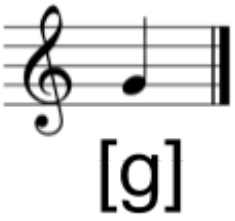


# The Guido Project

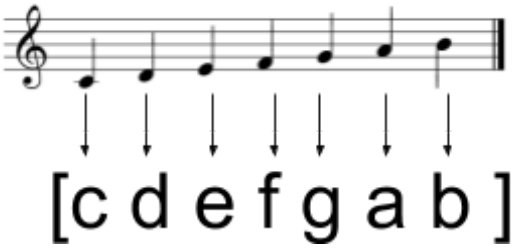
The Guido Project is an open source project hosted on github that encompasses a music notation format, a score rendering engine and various music score utilities. The Guido Music Notation Format is a general purpose formal language for representing score-level music in a platform-independent plain text and human readable way.[1]
In the following sections, we introduce the representation with examples of the included and excluded samples.

## 1. Text Representation

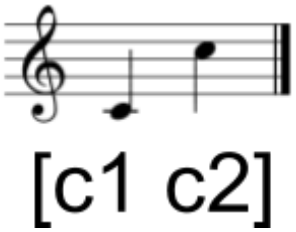
1. One Note:



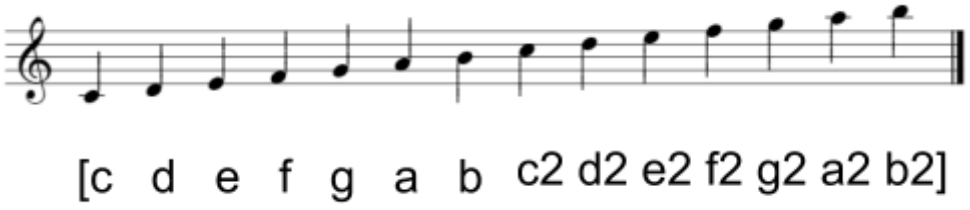

2. Notes change its value by the position relative to the lines:




3. Octaves, notes change their position using the octave value:

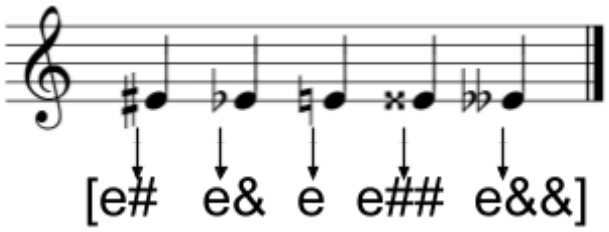



The values repeat itself (a->g then repeat):

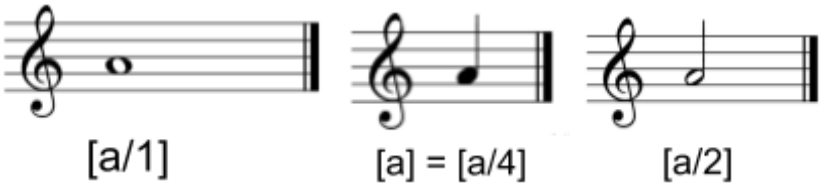


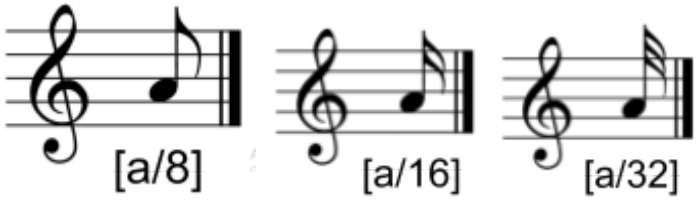

[1] https://guido.grame.fr/

4. Accidentals, putting these characters after each note add these shapes:

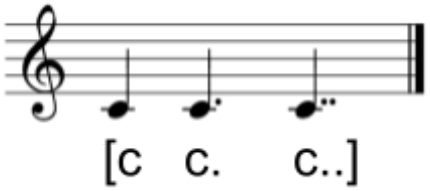



5. Duration, dividing the character by (1,2,4,8,16,32) and it change the shape:

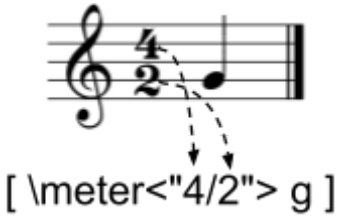


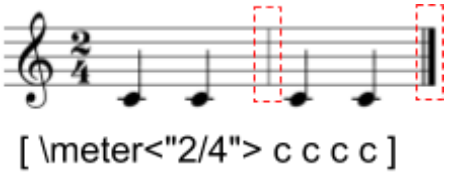

6. Augmentation dots:



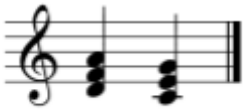

7. Time Signature:




8. Barlines:

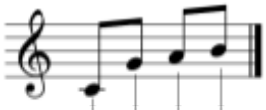


9. Chords:

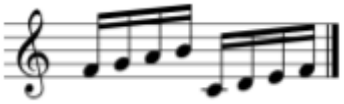


[ {d,f,a} {c,e,g} ]

10. Beaming

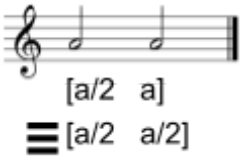


[c/8 g a b]



[f/16 g a b c d e f]

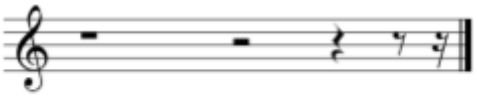
11. Sticky concept:
    Sticky means that this property will stay on the next notes even if it is not written, it applies on (Octave, Duration) [For the project: use the complete description. Don't use the sticky representation]



[a/2 a]
[a/2 a/2]

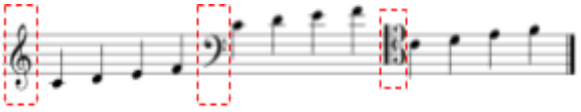
Examples of the not included shapes (other excluded shapes exist) :

1. Rests:



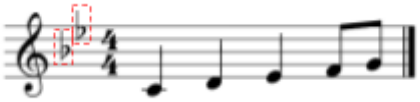
[ _/1 _/2 _/4 _/8 _/16 ]

2. Clefs:



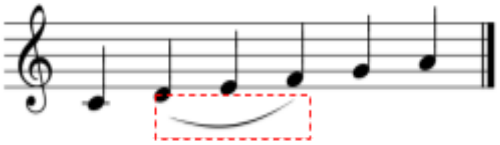
[ \clef<"g2"> c d e f
\clef<"f4"> c d e f
\clef<"c4"> c d e f ]

3. Key Signature



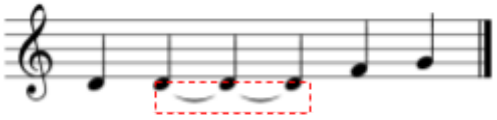
[ \meter<"4/4"> \key<-2> c d e& f/8 g ]

4. Slurs



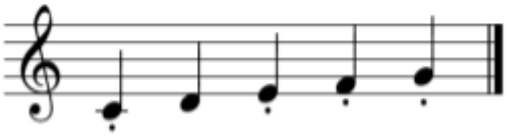
[ c \slur( d e f ) g a ]

5. Ties



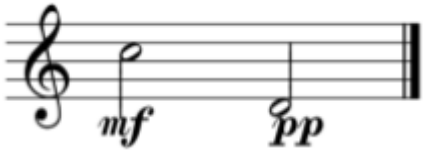
[ d \tie( d d d ) f g ]

6. Articulations



[ \stacc(c) d \stacc(e f g) ]

7. Expressions




[ \intens<"mf"> c2/2 \i<"pp"> d1/2 ]

## 2. Further resources

Online Editor: can be used to convert text to sheet music:https://guidoeditor.grame.fr/

# Evaluation & Grading

## Grading Criteria

1. Accuracy: 70%:
2. Quality of deliverables: 5%
3. Code modularity, readability and style: 10%
4. Code Performance [Memory & Time]: 5%
5. Methodology: 10%
6. Individual Work & Understanding: Each individual is given a percentage of team grade based on his/her work and understanding.

## Notes about evaluation

1. Non-working code will result in zero grade.
2. The project will be automatically graded on competition based metrics. In other words, your work will be compared to other students in the evaluation metrics and you will get your grades accordingly.
3. To calculate accuracy, we will run an automated script using a test set, afterwards, accuracy will be ordered descendly and your grade will be assigned accordingly.

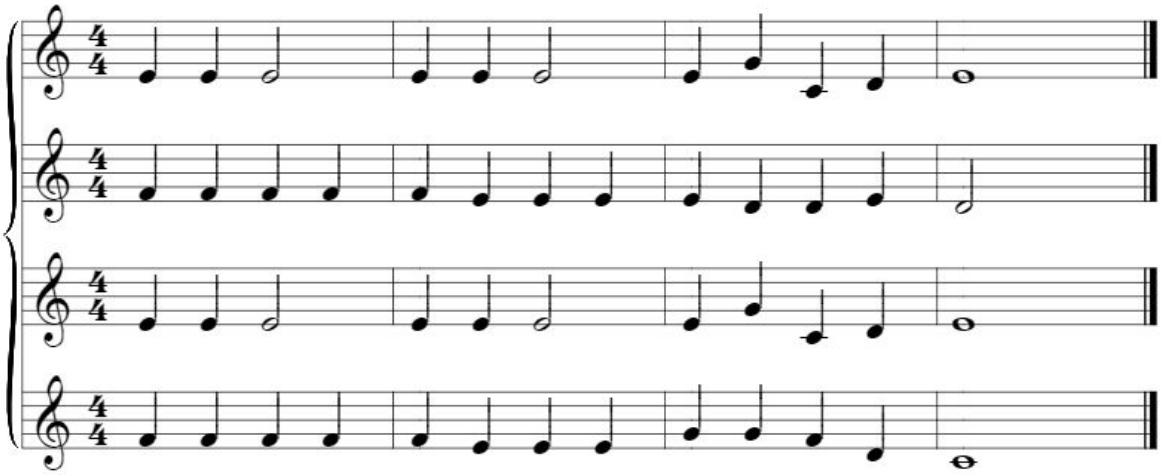
## Accuracy Calculation

For each output text representation, Accuracy will be calculated based on:

1. Difference between correct text representation [Ground truth] and the output text representation.
2. Other similar metrics.

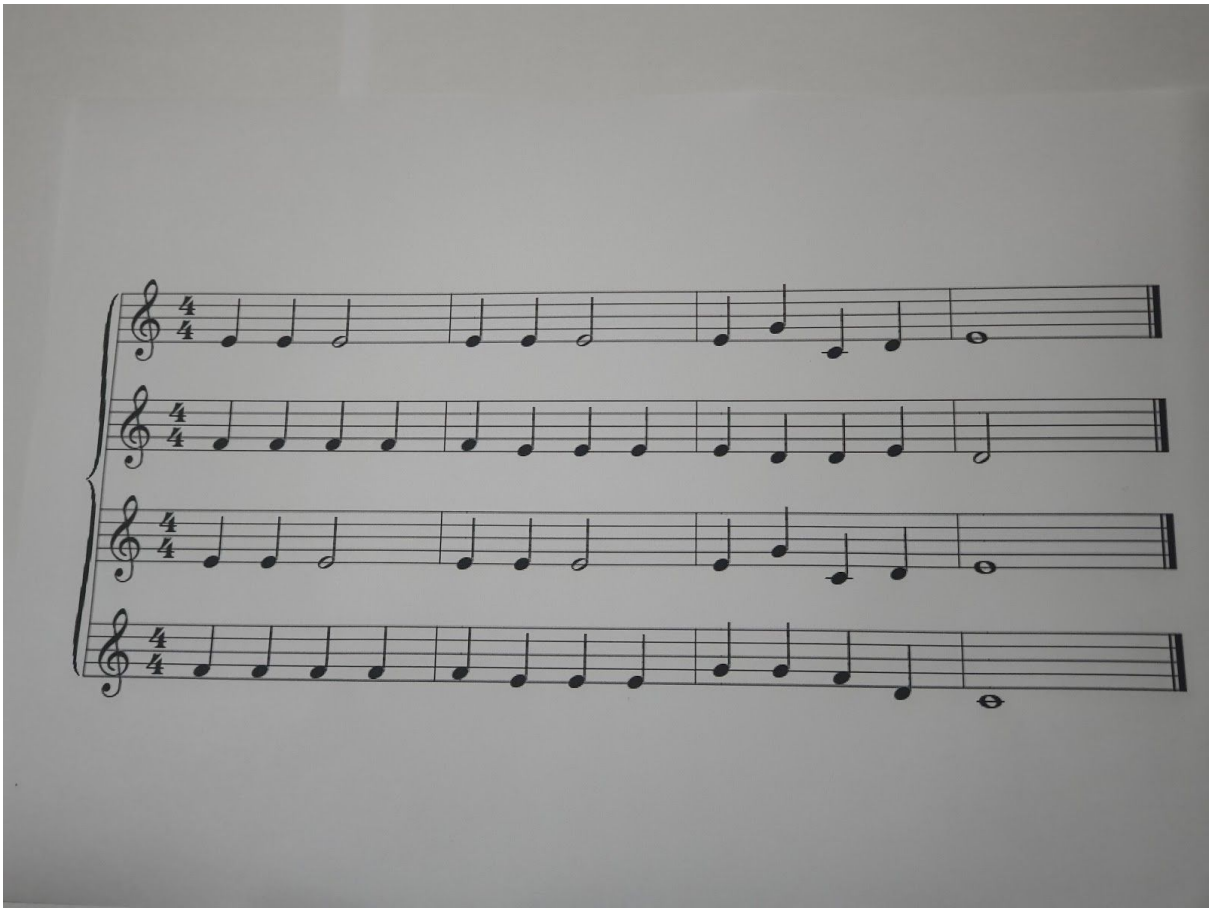
# Test cases levels

1. Printed & scanned musical notes.

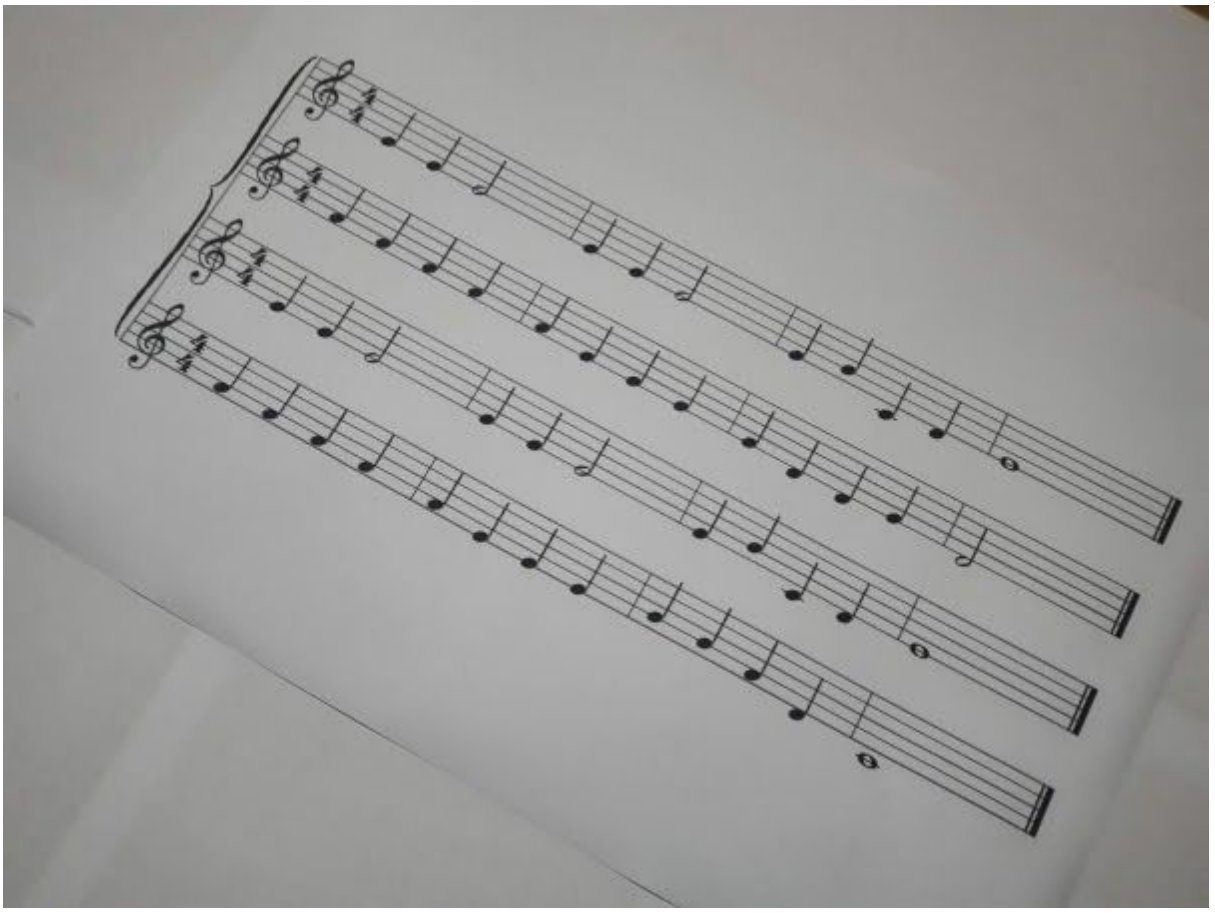


***Jingle Bell Note***

2. Printed & camera-captured musical notes.



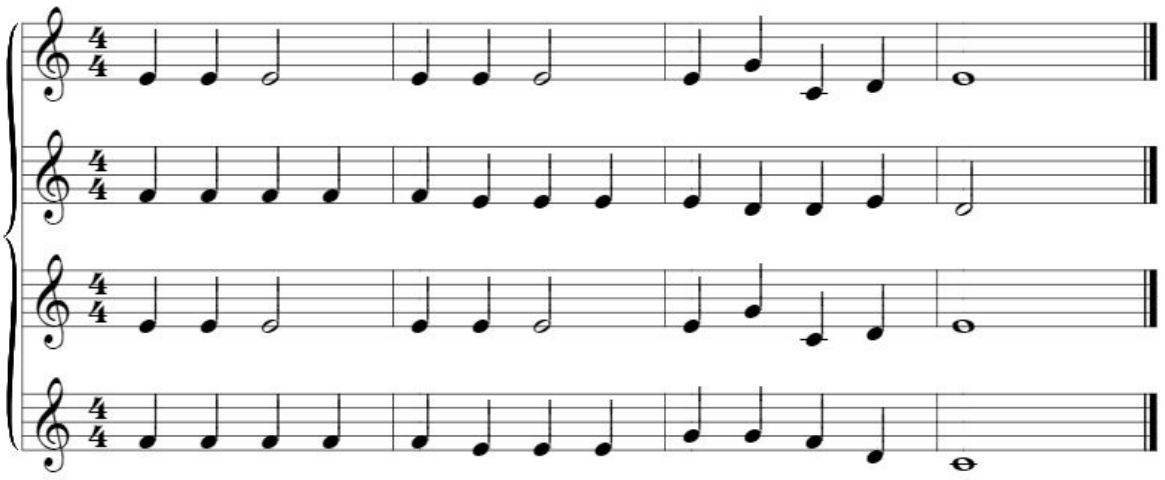
3. Printed & camera-captured with non-perfect modifications[ orientation, skewing, …]



4. Hand-written & scanned musical notes.
5. Hand-written & camera-captured musical notes.
6. Hand-written & camera-captured with non-perfect modifications[ orientation, skewing, light and etc. ]

# I/O Samples

Input Image



## Output Text

```
{
[\meter<"4/4"> e1/4 e1/4 e1/2  e1/4 e1/4 e1/2 e1/4 g1/4 c1/4 d1/4 e1/1],

[\meter<"4/4"> f1/4 f1/4 f1/4 f1/4  f1/4 e1/4 e1/4 e1/4 e1/4 d1/4 d1/4 e1/4 d1/2 ],

[\meter<"4/4"> e1/4 e1/4 e1/2 e1/4 e1/4 e1/2 e1/4 g1/4 c1/4 d1/4 e1/1],

[\meter<"4/4"> f1/4 f1/4 f1/4 f1/4 f1/4 e1/4 e1/4 e1/4 g1/4 g1/4 f1/4 d1/4 c1/1]
}
```

## Important note

Since:

e/2 e e e ☰ **e/2 e/2 e/2 e/2**

and

e2 e e e ☰ **e2 e2 e2 e2**

You must represent the note using the whole representation each time, going with the bold representation.
**Don’t use the short sticky representation.**

# Deliverables

Constraints and more details will be updated later.

## Code

- Compatible with: Linux Ubuntu 20.04. Don't use native libraries unless you take explicit permission from the TA. To install the required packages, we should only need to run **conda install** given the requirements file.
- You code should be able to run for a complete folder as following:
  **`python main.py <input-folder> <output-folder>`**
  Where <*input-folder*> is the absolute path for input folder which contains the input samples and <*output-folder*> is the absolute path for the output folder.
  for example:
  **`python main.py /home/user1/input-folder /home/user1/output-folder`**
  For each image in input-folder, you should process it and output the result in a file the same name with ".out" extension. For example: "1.png" should be processed and added to the output folder named "1.out"

- After creating a new conda environment for your project with the requirements.txt file, we are expecting your code to work out of the box.
- For comparison, all your code will run on the same machine using the CPU.

## Project Report

The project report should contain the following:

- Used algorithms
- Experiment results and analysis
- Work division between team members
- Accuracy, performance
- Conclusion and references.
- Any additional comments

## Presentation

# Hints & Rules

- CNN, RNN and any neural-network approach for automated feature extraction isn't permitted, however, you can use classifiers or naive neural networks after manual features extraction.
- No support will be provided during the last 48 hours before any delivery, submission or discussion.
- It is expected that you will consume considerable time to read about the subject and skim some papers to know more about the problem.
- Only teams of 4 members are allowed.

- Usage of open-source code or functions:
    - Clear permission should be taken from a TA before using any implemented function or open-source code except primitive functions.
    - If the TA allowed the usage of any open-source code, attribution should be written in the report and inside the code itself.
    - Any violation to these rules will be considered cheating.
- Any cheating is penalized by 0 in the project and -10 in the other work-grades.
- You are recommended to use Private repository (For examples: Github or Gitlab) to provide:
    - Work management.
    - Backup, in case the final project doesn't work, you can revert to an older version.
- Reasonable performance is a must, however, high speed up (using GPU, parallel execution, cython or so on) is a plus [For high speed up, a bonus might be given].