Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

MSE | MASTER OF SCIENCE IN ENGINEERING

Multimodal Processing, Recognition and Interaction

Practical work: Sudoku ocr with SVM

Joël Dumoulin
Elena Mugellini, Jean Hennebert

Goal

- The goal of this practical work is to build an OCR application for Sudoku grid image recognition, using an SVM classifier.
- We will use scikit-learn and scikit-image python toolkits.

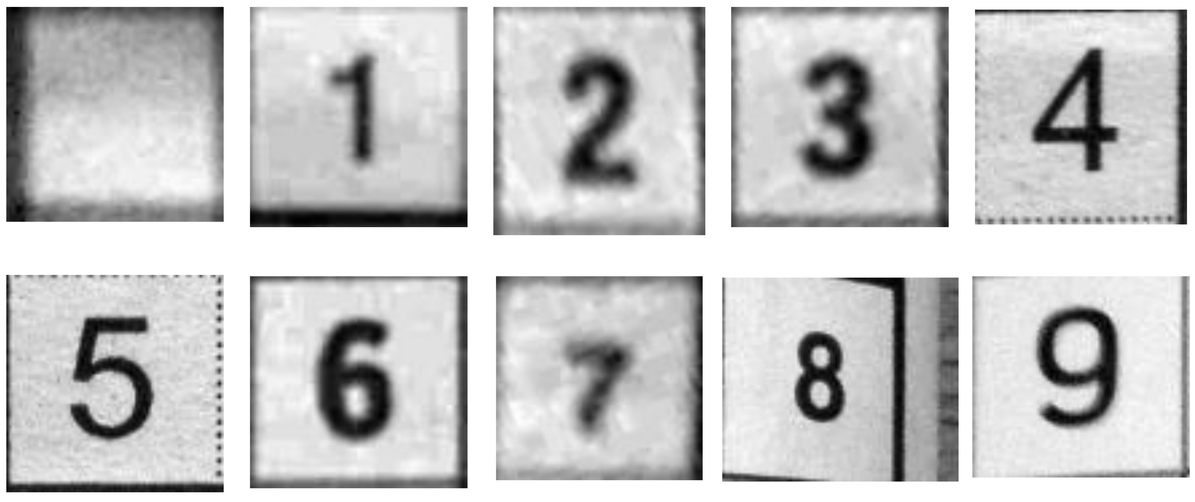
Introduction

- The skeleton of the application is available.
- All the parts you have to code by yourself are designed by # TODO.

Introduction

- The data directory contains
  - ocr_data
    - JPEG images for the 10 classes (0 to 9, 0 means empty) in this naming format: `class idx.jpg`

This is our training dataset!



Introduction

- and
  - sudokus
    - Sample images of sudoku grids (some are more challenging than others!). For each grid, there is a .sud text file, containing the real grid values.

2			5	7		4		6
4	9	1				7		5
			2		4			
	6		9			3		
		5	6	2	8	9		
		9			1		6	
			1		5			
5		4				1	9	3
7		6		9	3			4

Introduction

- Advices
  - Refer to what you already did in the previous 2 practical sessions.
  - Look for help on the toolkits websites or other online ressources (ex: StackOverflow)

Summary

- Part 1: Feature extraction
- Part 2: Classifier training
- Part 3: Sudoku cells extraction
- Part 4: Classification
- Part 5: Tuning
- Optionnal
  - Part 6: Sudoku solving!

Part 1: Feature extraction

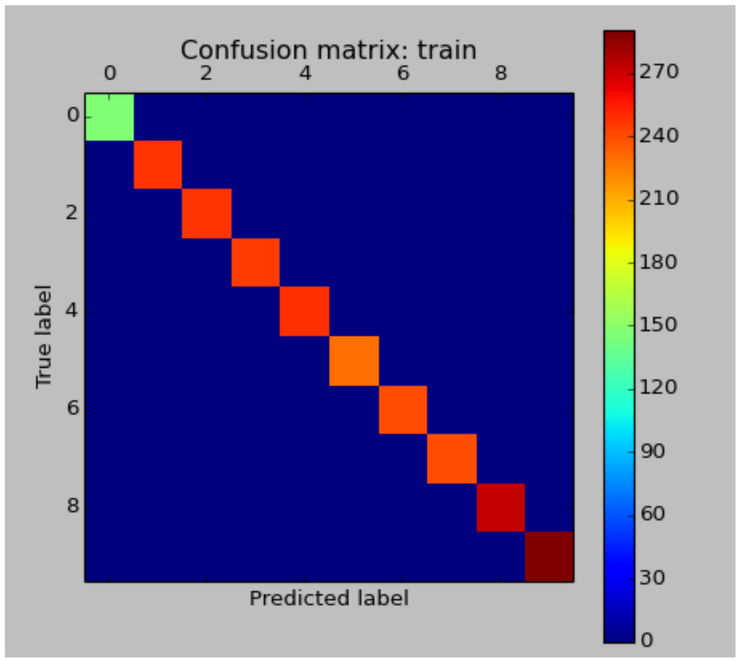
- We first need to load all training data (images features + labels)
  - Iterate over images
    - Load image as gray level

      ```
      np.array(Image.open(im_name).convert('L')
      ```

    - Process image (remove borders + resize)
    - Extract features (just the flattened image as a starting point...)

Part 2: Classifier training

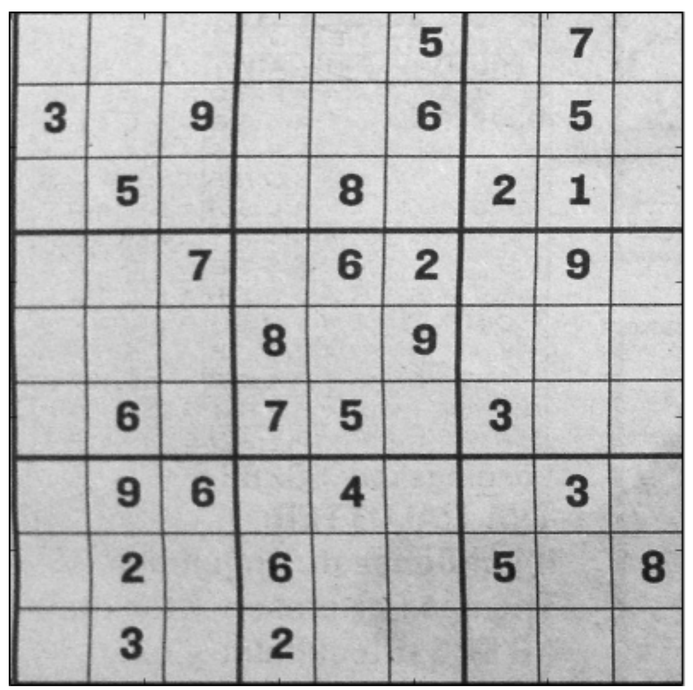
- Instantiate and train a classifier
  - Choose an adapted kernel
- Do cross-validation to get a coherant classification accuracy indication
  - You should get a classification accuracy between 0.98 and 1
- Train the classifier on the whole set



Part 3: Sudoku cells extraction

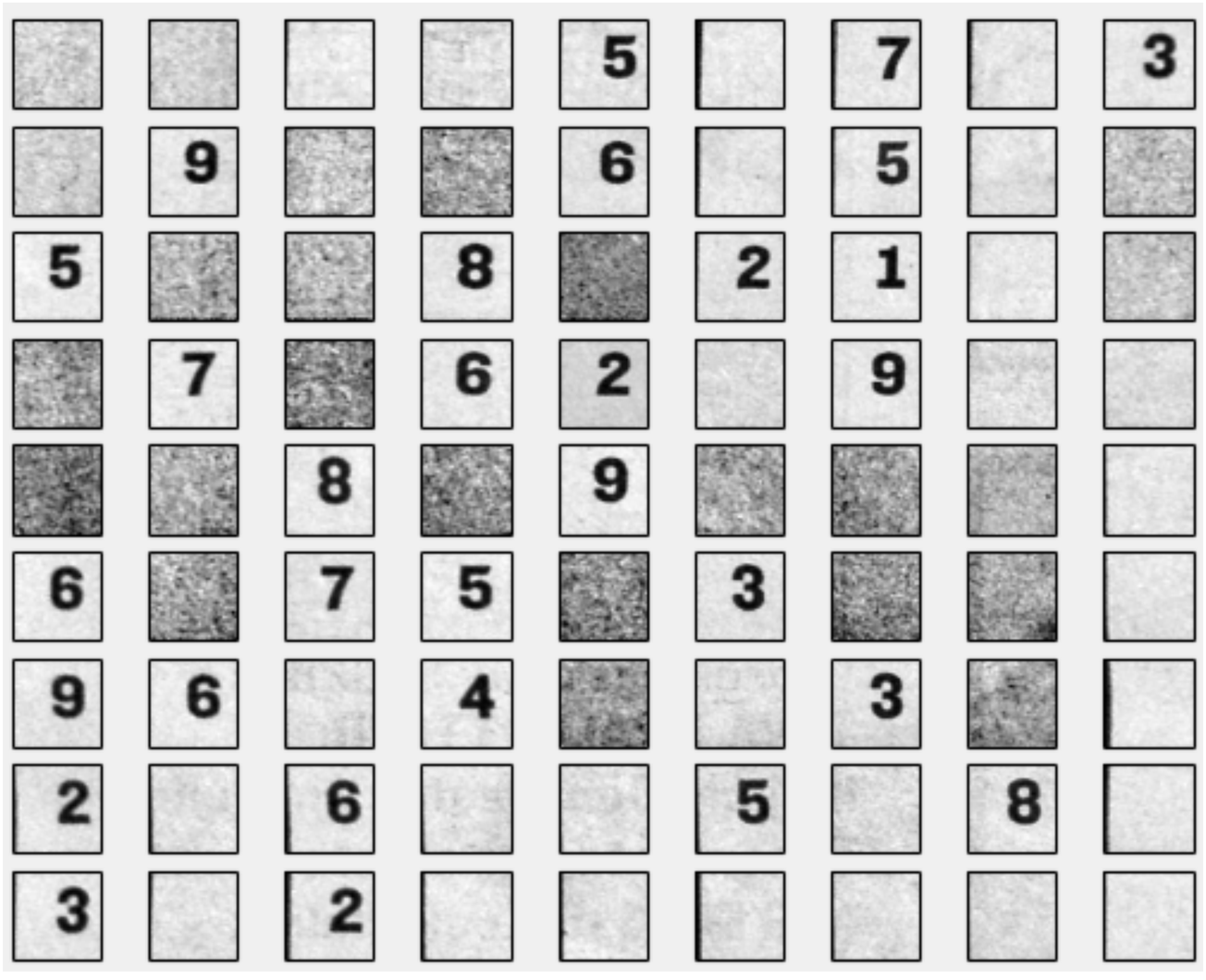
- As it is mostly python code and not ML-related, most of the code is done for you ☹/☺
  - Anyway, try to read the code and understand what is done!
  - Basically, once we get the squared grid, we just divide it (9x9)
- Use the scikit-image warping function to apply the transformation

`skimage.transform.warp()`



Part 3: Sudoku cells extraction

- Result example of cell extraction



Part 4: Classification

- Load the sudoku image as gray level
- Extract cells
- For each cell, do feature extraction
- Then, simply use the trained classifier to do the prediction
- Print a classification report and a confusion matrix to evaluate the classification
- Finally, print the resulting sudoku grid as a 9x9 array

Part 5: Tuning

- There are several parts in this application to tune
  - First, try to tune the classifier
  - Then, try to use better features
    - Ex: HOG (available in scikit-image)
  - Finally, think about the image pre-processing (you don't necessary need to implement this last part)

Part 6: Sudoku solving

- Now we have a sudoku grid, why not try to solve it?!
- Once the grid is solved, the solution could be added to the input grid!

Questions

- Which kernel did you use and why?
- If you have good cross-validation results, does that means that you will have good sudoku classification performance?
- What is the role of the dataset size? Try your system with only a few training samples.
- What are all the parts of this application that could be enhanced?
  - Think in terms of machine learning first, and then in terms of features.
- If you would like to code a CAPTCHA reader, would it be possible to adapt this application to do this? Which part could be reused? What should be added?

Expected work

- On Moodle, an archive containing
  - Your code (without the data folder)
  - A report documenting your work, observations, and answers

Ressources

- Learn python: http://www.learnpython.org
- Scikit-learn: http://scikit-learn.org
- Scikit-image: http://scikit-image.org

References

- Programming computer vision with Python: http://programmingcomputervision.com

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences
Western Switzerland

Joël Dumoulin

joel.dumoulin@hes-so.ch

http://joel.dumoulin.ch