The Chinese University of Hong Kong, Shenzhen
SDS · School of Data Science
Andre Milzarek · Fall Semester 2023/24

MDS 6106 / CIE 6010-1 — Optimization and Modeling

Final Project

Image Inpainting and Nonconvex Image Compression

The goal of this project is to investigate different optimization models and to utilize minimization methodologies that were introduced in the lectures to reconstruct images from partial data.

Project Description. Typically, a *grey-scale* image $\boldsymbol{U} \in \mathbb{R}^{m \times n}$ is represented as a $m \times n$ matrix where each entry U_{ij} represents a specific pixel of the image containing the color information. If the columns of $\boldsymbol{U} = (\boldsymbol{u}_{(1)}, \boldsymbol{u}_{(2)}, \dots, \boldsymbol{u}_{(n)})$ are stacked, we obtain the vector form

$$\boldsymbol{u} = \mathrm{vec}(\boldsymbol{U}) = (\boldsymbol{u}_{(1)}^\top, \boldsymbol{u}_{(2)}^\top, \dots, \boldsymbol{u}_{(n)}^\top)^\top \in \mathbb{R}^{mn}$$

of the image $\boldsymbol{U}$. In this project, we consider a class of imaging problems that is known as *inpainting problems*. As introduced in the lectures, inpainting describes the task of recovering an image $\boldsymbol{U}$ from partial data and observations. Specifically, we assume that parts of the image $\boldsymbol{U}$ are missing or damaged, (e.g., due to scratches, stains, or compression), and the aim is to reconstruct this missing or damaged information via solving a suitable inpainting or optimization problem.

Figure 1: Examples of different damaged images. We want to recover the missing image information in the green areas.

The images in Figure 1 show several typical situations where inpainting techniques can be applied. Our overall task is to reconstruct the green target areas. In this project, we assume that these target areas are known, i.e., we have access to a binary mask $\texttt{Ind} \in \mathbb{R}^{m \times n}$ with

$$\texttt{Ind}_{ij} = \begin{cases} 1 & \text{the pixel } (i,j) \text{ is not damaged,} \\ 0 & \text{the pixel } (i,j) \text{ is damaged.} \end{cases}$$

This mask contains the relevant information about missing or damaged parts in the image.

Let us set $\texttt{ind} := \mathrm{vec}(\texttt{Ind}) \in \mathbb{R}^{mn}$, $s := \sum_{i=1}^{mn} \texttt{ind}_i$, and $\mathcal{I} := \{i : \texttt{ind}_i = 1\}$. Furthermore, let $\mathcal{I} = \{q_1, q_2, \dots, q_s\}$ denote the different elements of the index set $\mathcal{I}$. Then, we can define the following *selection matrix*

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{e}_{q_1}^\top \\ \vdots \\ \boldsymbol{e}_{q_s}^\top \end{pmatrix} \in \mathbb{R}^{s \times mn}, \quad \boldsymbol{e}_j \in \mathbb{R}^{mn}, \quad [\boldsymbol{e}_j]_i = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases} \quad \forall\, j \in \mathcal{I}.$$

The vectors $\boldsymbol{e}_j$ are unit vectors in $\mathbb{R}^{mn}$ and the matrix $\boldsymbol{P}$ selects all undamaged pixels of a stacked image according to the mask `ind`. In particular, if $\boldsymbol{U} \in \mathbb{R}^{m\times n}$ is the input image and $\boldsymbol{u} = \mathrm{vec}(\boldsymbol{U})$ is its stacked version, then the vector

$$\boldsymbol{b} = \boldsymbol{P}\boldsymbol{u} \in \mathbb{R}^s$$

contains the color information of all undamaged pixels of the original image $\boldsymbol{U}$. Hence, we now want to find a reconstruction $\boldsymbol{y} \in \mathbb{R}^{mn}$ of $\boldsymbol{u}$ such that:

(i) Original information of the undamaged parts in $\boldsymbol{U}$ is maintained, i.e., $\boldsymbol{y}$ satisfies

$$\boldsymbol{P}\boldsymbol{y} = \boldsymbol{b} \quad \text{or} \quad \boldsymbol{P}\boldsymbol{y} \approx \boldsymbol{b}. \tag{1}$$

(ii) The image $\boldsymbol{y}$ can recover the missing parts in $\boldsymbol{U}$ in a "suitable" way.

In this project, we will discuss different models and strategies to achieve these goals.

Project Tasks.

1. *Sparse Reconstruction and L-BFGS*. The linear system of equations (1) is underdetermined and has infinitely many possible solutions. In order to recover solutions with a "natural image structure", we (re)consider the so-called ℓ_1-regularized image reconstruction problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^{mn}} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2 + \mu\|\boldsymbol{x}\|_1, \tag{2}$$

where $\boldsymbol{b} \in \mathbb{R}^s$ is derived as in (1) and $\mu > 0$ is given. The efficiency and success of this model is based on the fact that the ℓ_1-regularization promotes sparsity and that there exist canonical sparse representations of the image data. The idea is to use a linear transformation $\boldsymbol{x} = \boldsymbol{\Psi}\boldsymbol{y}$ of the image $\boldsymbol{y}$ in (1) and to transfer it to the *frequency domain*. In this new space, many images have a very sparse representation and many of the components x_i are zero or close to zero. This motivates the choice of the ℓ_1-norm, $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{mn} |x_i|$, in the model (2). The quadratic term in (2) corresponds to the condition (1) and is small when the pixels of undamaged parts of $\boldsymbol{u}$ and of the corresponding reconstruction have close values.

In this part, we want to use the discrete cosine transformation as sparse basis for the images, i.e., we can set $\boldsymbol{x} = \boldsymbol{\Psi}\boldsymbol{y} = \mathtt{dct}(\boldsymbol{y})$. Since the DCT-transformation is orthogonal, this leads to the following definitions:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{P}\mathtt{idct}(\boldsymbol{x}), \quad \boldsymbol{A}^\top\boldsymbol{v} = \boldsymbol{A}^\top(\boldsymbol{v}) = \mathtt{dct}(\boldsymbol{P}^\top\boldsymbol{v}), \quad \boldsymbol{x} \in \mathbb{R}^{mn}, \quad \boldsymbol{v} \in \mathbb{R}^s,$$

where `idct` denotes the inverse DCT-transformation. In `MATLAB`, these operations can be represented compactly as functions via

$\boldsymbol{P} =$ `@(x) x(ind);` $\quad\boldsymbol{A} =$ `@(x) P(idct(x));` $\quad\boldsymbol{A}^\top\boldsymbol{A} =$ `@(x) dct(ind.*idct(x))`.

(Similar implementations are also possible in `Python`, e.g., using `scipy`-packages and code). Since the ℓ_1-norm is not differentiable, we also want to consider a smoothed and popular variant of the ℓ_1-problem:

$$\min_{x\in\mathbb{R}^{mn}} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2 + \mu\sum_{i=1}^{mn}\varphi_{\mathrm{hub}}(x_i), \tag{3}$$

where the *Huber-function* is defined as follows:

$$\varphi_{\mathrm{hub}}(z) = \begin{cases} \frac{1}{2\delta}z^2 & \text{if } |z| \le \delta, \\ |z| - \frac{1}{2}\delta & \text{if } |z| > \delta, \end{cases} \qquad z \in \mathbb{R}, \quad \delta > 0.$$

In contrast to the ℓ_1-norm, the Huber function is continuously differentiable and hence, gradient-type methods can be applied to solve the problem (3).

- Implement the accelerated gradient method (AGM) for the smooth Huber-loss formulation (3). (It holds that $\lambda_{\max}(\boldsymbol{A}^\top \boldsymbol{A}) \leq 1$).
- Implement the globalized L-BFGS method (the algorithm in Lecture L-19 with L-BFGS updates) for the problem (3). You can use backtracking to perform the line search and the two-loop recursion to calculate the L-BFGS update. You can choose

$$\boldsymbol{H}_k^0 = \frac{(\boldsymbol{s}^k)^\top \boldsymbol{y}^k}{\|\boldsymbol{y}^k\|^2} \cdot \boldsymbol{I}, \quad \boldsymbol{s}^k = \boldsymbol{x}^k - \boldsymbol{x}^{k-1}, \quad \boldsymbol{y}^k = \nabla f(\boldsymbol{x}^k) - \nabla f(\boldsymbol{x}^{k-1})$$

  as initial matrix for the update. In order to guarantee positive definiteness of the L-BFGS update, the pair $\{\boldsymbol{s}^k, \boldsymbol{y}^k\}$ should only be added to the current curvature pairs if the condition $(\boldsymbol{s}^k)^\top \boldsymbol{y}^k > 10^{-14}$ is satisfied. Suitable choices for the memory parameter are $m \in \{5, 7, 10, 12\}$.
- A suitable image quality measure is the so-called PSNR value. Suppose that $\boldsymbol{u}^* = \mathrm{vec}(\boldsymbol{U}^*)$ is the original true (and undamaged) image, then we have:

$$\mathrm{PSNR} := 10 \cdot \log_{10} \left[\frac{mn}{\|\boldsymbol{y} - \boldsymbol{u}^*\|^2} \right] \quad \text{where} \quad \boldsymbol{y} = \texttt{idct}(\boldsymbol{x}).$$

  Compare the results of the two methods and models with respect to the PSNR value, the number of iterations, and the required cpu-time. Test different parameter settings and variants of the methods to improve the performance of your implementations. Plot the reconstructed images and the convergence behavior of accelerated gradient method and L-BFGS. How does the PSNR value behave for the different methods?
- The choice of the parameter μ and δ can depend on the tested images; good choices are $\mu \in [0.001, 0.1]$ and $\delta \in [0.01, 0.5]$. (The Huber function is closer to the absolute value $|\cdot|$ for smaller choices of δ).

Hints and Guidelines:

- On Blackboard, we have provided two datasets containing test images and test inpainting masks. Compare the performance of your methods on several images and different type of masks.
- Typically, the pixel values U_{ij} are represented by integer numbers in $[0, 255]$. Scale the images to $[0, 1]$. A common initial point is zero: $\boldsymbol{x}^0 = \texttt{zeros(m*n,1)}$.
- For debugging, you can test your implementation of the L-BFGS strategy first on a simpler example.

2. *Total Variation Minimization.* The total variation minimization problem utilizes a different regularization to solve the inpainting task (1). As discussed in the lectures and exercises, the optimization problem is given by

$$\min_{\boldsymbol{x} \in \mathbb{R}^{mn}} \ \frac{1}{2}\|\boldsymbol{P}\boldsymbol{x} - \boldsymbol{b}\|^2 + \mu \sum\nolimits_{i=1}^{m-1} \sum\nolimits_{j=1}^{n-1} \|\boldsymbol{D}_{(i,j)}\boldsymbol{x}\|_2, \quad \mu > 0, \tag{4}$$

Here, $\boldsymbol{x} = \mathrm{vec}(\boldsymbol{X})$ corresponds to a vectorized image $\boldsymbol{X} \in \mathbb{R}^{m \times n}$, $\boldsymbol{P} \in \mathbb{R}^{s \times mn}$ is the selection matrix that selects pixels according to a given inpainting mask, and $\boldsymbol{b} \in \mathbb{R}^s$ contains the undamaged image information (as introduced in (1)). The matrices $\boldsymbol{D}_{(i,j)}$ are discretized image gradients and $\mu > 0$ is a given regularization parameter.

In contrast to the ℓ_1-problem, this regularization also works in the pixel domain and we do not need to transform the image $\boldsymbol{x}$ to the frequency space.

- The original total variation model (4) utilizes the nondifferentiable Euclidean norm $\|\cdot\|_2$ to penalize the image gradients. Design a suitable smooth version of this model and apply and implement the Newton-CG method (L-18) to solve your model.
- Run your code for different test images and masks and report your results and the performance of your implementation! Compare the PSNR values with the results in part 1.) – does the total variation model achieve better results?
- Discuss the effect of your smoothed model (e.g., in case it contains additional hyper-parameters). Consider possible refinements and improvements of your model choice based on your general observations and numerical results.

Hints and Further Guidelines:

- On Blackboard, we have provided functions `D = get_D(m,n)` to compute the operator $\boldsymbol{D}$ for given dimensions m and n. You can use this code (or your own solutions) to generate $\boldsymbol{D}$.

3. *Nonconvex Image Compression.* In this final part of the project, we try to investigate a slightly different question related to inpainting. Given an image $\boldsymbol{u} \in \mathbb{R}^{mn}$, can we construct a binary mask $\boldsymbol{c} = \texttt{ind} \in \{0,1\}^{mn}$ such that $s = \sum_{i=1}^{mn} c_i$ is small as possible, but the image $\boldsymbol{u}$ can still be reconstructed with reasonable quality by only using pixels $j \in \{1,\dots,mn\}$ with $c_j = 1$? In this part, we want to consider a PDE-based image compression technique that allows to compute such a mask $\boldsymbol{c}$ and the corresponding reconstruction $\boldsymbol{x}$ of $\boldsymbol{u}$ simultaneously. The model is given by

$$\min_{\boldsymbol{x},\boldsymbol{c}} \ \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{u}\|^2 + \mu\|\boldsymbol{c}\|_1 \quad \text{s.t.} \quad \mathrm{diag}(\boldsymbol{c})(\boldsymbol{x}-\boldsymbol{u}) - (\boldsymbol{I} - \mathrm{diag}(\boldsymbol{c}))\boldsymbol{L}\boldsymbol{x} = \boldsymbol{0}, \quad \boldsymbol{c} \in [\boldsymbol{0}, \mathbb{1}], \tag{5}$$

where $\boldsymbol{u} = \mathrm{vec}(\boldsymbol{U}) \in \mathbb{R}^{mn}$ is the *ground truth image*, $\boldsymbol{x} \in \mathbb{R}^{mn}$ is the reconstructed image, $\boldsymbol{c} \in \mathbb{R}^{mn}$ denotes the *inpainting mask*, and $\boldsymbol{L} = -\boldsymbol{D}^\top\boldsymbol{D} \in \mathbb{R}^{mn\times mn}$ is the Laplacian operator.

As long as one element in $\boldsymbol{c}$ is nonzero, the matrix $\boldsymbol{A}(\boldsymbol{c}) = \mathrm{diag}(\boldsymbol{c}) + (\mathrm{diag}(\boldsymbol{c}) - \boldsymbol{I})\boldsymbol{L}$ can be shown to be invertible and hence, $\boldsymbol{x}$ can be expressed explicitly via $\boldsymbol{x} = \boldsymbol{A}(\boldsymbol{c})^{-1}\mathrm{diag}(\boldsymbol{c})\boldsymbol{u}$. In this case, the problem (5) can be reduced to

$$\min_{\boldsymbol{c}} \ \frac{1}{2}\|\boldsymbol{A}(\boldsymbol{c})^{-1}\mathrm{diag}(\boldsymbol{c})\boldsymbol{u} - \boldsymbol{u}\|^2 + \mu\|\boldsymbol{c}\|_1 \quad \text{s.t.} \quad \boldsymbol{c} \in [\boldsymbol{0}, \mathbb{1}]. \tag{6}$$

Setting $f(\boldsymbol{c}) := \frac{1}{2}\|\boldsymbol{A}(\boldsymbol{c})^{-1}\mathrm{diag}(\boldsymbol{c})\boldsymbol{u} - \boldsymbol{u}\|^2$, this model has a standard form. Furthermore, the gradient of f is given by

$$\nabla f(\boldsymbol{c}) = \mathrm{diag}(-\boldsymbol{L}\boldsymbol{x} + \boldsymbol{u} - \boldsymbol{x})[\boldsymbol{A}(\boldsymbol{c})^\top]^{-1}(\boldsymbol{x} - \boldsymbol{u}) \quad \text{where} \quad \boldsymbol{x} = \boldsymbol{A}(\boldsymbol{c})^{-1}\mathrm{diag}(\boldsymbol{c})\boldsymbol{u}.$$

In Figure 2, an exemplary output or solution of this problem is shown. The figure in the middle depicts the mask $\boldsymbol{c}$ and the associated reconstruction $\boldsymbol{x}$ of $\boldsymbol{u}$ is shown on the right. In this example, the mask has a density of $\frac{s}{mn} \approx 6.38\%$ pixels, i.e., only 6.38% of the pixels of the ground truth image (which is shown on the left) are used.

- Explain why the objective function g of problem (6) can be additionally simplified to $g(\boldsymbol{c}) = f(\boldsymbol{c}) + \mu\mathbb{1}^\top\boldsymbol{c}$.
- Implement the inertial gradient method that was presented in the lecture (L-17) to solve the nonconvex optimization problem (6). The constraints "$\boldsymbol{c} \in [\boldsymbol{0}, \mathbb{1}]$" can be handled by utilizing projected inertial gradient steps of the form

$$\boldsymbol{c}^{k+1} = \mathcal{P}_{[\boldsymbol{0},\mathbb{1}]}(\boldsymbol{c}^k - \alpha_k[\nabla f(\boldsymbol{c}^k) + \mu\mathbb{1}] + \beta_k[\boldsymbol{c}^k - \boldsymbol{c}^{k-1}]),$$

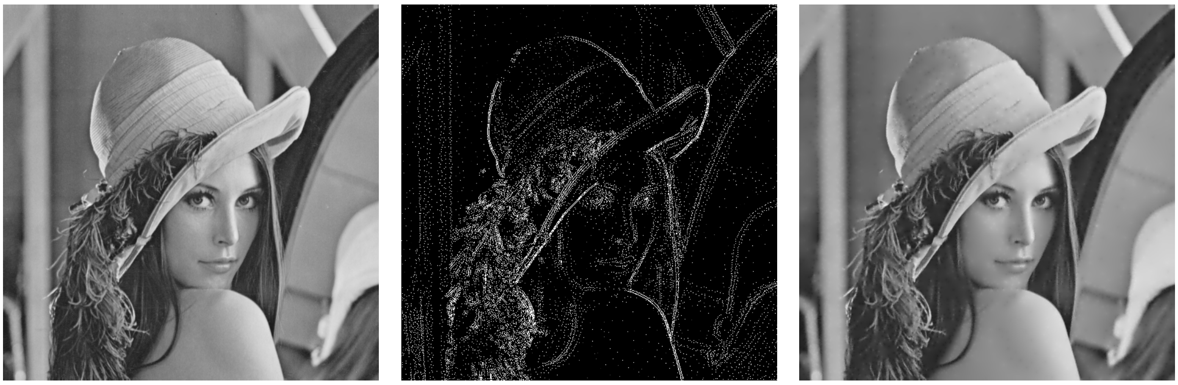

Figure 2: Image compression via optimal selection masks. Left: ground truth image $\boldsymbol{u}$. Middle: inpainting mask $\boldsymbol{c}$. Right: reconstructed image $\boldsymbol{x}$. The mask $\boldsymbol{c}$ only selects 6.38% of the available pixels. The PSNR value of the recovered image is 35.1.

where $[\mathcal{P}_{[\mathbf{0},\mathbb{1}]}(\boldsymbol{x})]_i = \mathcal{P}_{[0,1]}(x_i) = \max\{0, \min\{x_i, 1\}\}$ is a component-wise projection onto the interval $[\mathbf{0}, \mathbb{1}]$. (We will discuss projections in the forthcoming lectures in more detail).

You can use the method `get_D` to generate the Laplacian operator $\boldsymbol{L} = -\boldsymbol{D}^\top \boldsymbol{D}$. We suggest to use $\boldsymbol{c}^0 =$ `ones(m*n,1)` as initial point and $\beta_k \equiv \beta = 0.8$ (or a different value close to 0.8). The step size α_k can be set as $\alpha_k = 1.99(1-\beta)/\ell$, where ℓ is an estimate of the Lipschitz constant of ∇f. As the Lipschitz constant ℓ of the gradient ∇f is unknown, a simple line search-type procedure can be used to determine ℓ adaptively:

while $f(\boldsymbol{c}^{k+1}) - f(\boldsymbol{c}^k) \geq \nabla f(\boldsymbol{c}^k)^\top (\boldsymbol{c}^{k+1} - \boldsymbol{c}^k) + \frac{\ell}{2}\|\boldsymbol{c}^{k+1} - \boldsymbol{c}^k\|^2$ **do:**

- Set $\ell = 2\ell$ and $\alpha_k = 1.99(1-\beta)/\ell$;
- Recompute $\boldsymbol{c}^{k+1} = \mathcal{P}_{[\mathbf{0},\mathbb{1}]}(\boldsymbol{c}^k - \alpha_k[\nabla f(\boldsymbol{c}^k) + \mu\mathbb{1}] + \beta_k[\boldsymbol{c}^k - \boldsymbol{c}^{k-1}])$;

Hints and Guidelines:

- The parameter μ depends on the specific ground truth image $\boldsymbol{u}$. Possible choices are $\mu \in [0.001, 0.01]$. If μ is too large, the mask $\boldsymbol{c}$ will be set to zero which will cause numerical issues and the reconstruction fails.
- This problem can be computationally very demanding. Try to implement the inertial gradient method as efficiently as possible! You can test your implementation first on smaller images. For instance, you can first scale the original image $\boldsymbol{u}$ to half of its size.
- Ensure that the matrix $\boldsymbol{A}(\boldsymbol{c})$ is in a `sparse` format. In this case, the backslash operator or similar methods for *sparse linear systems* can be utilized to compute the solution $\boldsymbol{x} = \boldsymbol{A}(\boldsymbol{c})^{-1}\mathrm{diag}(\boldsymbol{c})\boldsymbol{u}$ efficiently.
- In order to prevent the Lipschitz constant ℓ from being too large, you can decrease it by a certain factor after a fixed number of iterations, i.e., a possible strategy is

  `if mod(iter,5) = 0,` $\ell = 0.95 \cdot \ell$, $\alpha = 1.99(1-\beta)/\ell$, `end`

  You can also experiment with continuation strategies for the parameter μ: start with a larger parameter $\mu_0 > \mu$ and reduce it after a fixed number of iterations until it coincides with the original parameter μ.
- You can either use a number of iterations (250, 500, or 1000, ...) or the condition

$$\|\boldsymbol{c}^k - \boldsymbol{c}^{k-1}\|/\|\boldsymbol{c}^{k-1}\| \leq \texttt{tol}, \quad \texttt{tol} \in \{10^{-4}, 10^{-6}\},$$

as stopping criterion.

4. *Performance and Extensions.* In this final part of the project, we try to investigate additional improvements and variants of the algorithms discussed in the first three parts.

   – Based on your numerical experience, is it possible to further improve the performance of your implementation – i.e., by choosing different linesearch strategies, parameters (within the algorithm), or update rules? Try to revise your code and implement one (some) of the algorithms as efficient as possible. Report your changes and adjustments.

     Potential Key-Words and Ideas: Barzilai-Borwein step sizes; adjust line-search parameters; minimize the number of matrix-calls and computations (e.g., within CG); alternating (proximal) minimization strategies for (5) (i.e., can we solve (5) directly in an alternating fashion?); cheaper / potential approximations of $\boldsymbol{A}(\boldsymbol{c})^{-1}$ ...

   – Investigate other potential and possible extensions of your models (for instance, for colorful `RGB` images, ...).

   This part of the project is more open and not all of the mentioned points need to be addressed. In particular, other extensions and discussions are possible. Add comments if you have already improved your code and implementations while working on the first parts.

Project Report and Presentation. This project is designed for groups of four or five students. Please send the following information to `217012017@link.cuhk.edu.cn` before **December, 12th, 12:12 pm**:

- Name and student ID number of the participating students in your group, group name.
- The project presentation is scheduled for **Dec. 25th** or **Dec. 26th**. Please indicate which of the dates and which time you prefer to give your presentation. Let us also know if not all group members can attend or if an online presentation via zoom is required.

Please contact the instructor in case your group is smaller to allow adjustments of the project outline and requirements.

A report should be written to summarize the project and to collect and present your different results. The report itself should take no more than 15–20 typed pages plus a possible additional appendix. It should cover the following topics and items:

- What is the project about?
- What have you done in the project? Which algorithmic components have you chosen to implement? What are your main contributions?
- Summarize your main results and observations.
- Describe your conclusions about the different problems and methods that you have studied.

You can organize your report following the outlined structure in this project description. As the different parts in this project only depend very loosely on each other, you can choose to distribute the different tasks and parts among the group members. Please clearly indicate the responsibilities and contributions of each student and mention if you have received help from other groups, the teaching assistant, or the instructor.

Try to be brief, precise and fact-based. Main results should be presented by highly condensed and organized data and not just piles of raw data. To support your summary and conclusions, you can put more selected and organized data into an appendix which is not counted in the page limit. Please use a cover page that shows the names and student ID numbers of your group members.

The deadline for the report submission is **December, 24th, midnight**. Please submit your report (as pdf-document) and all supporting materials and code online using Blackboard.

The individual presentations of the project are scheduled for **December, 25th** or **December, 26th**. More information about the schedule will be shared later.