Data Visualization and Analysis

BINF4245

Prof. Dr. Renato Pajarola

Exercise and Homework Completion Requirements

1. Every **solved** exercise will earn you points, up to 15 in total.
   - *Late submissions (up to one day) will result in "-1" point.*
   - *If the solution works as expected, full points are awarded, else partial points are assessed as outlined in the code skeleton.*
2. The four exercises give rise to the following point distribution: 2 – 3 – 5 – 5.
   - *A **minimum of 7 points** from all four exercises must be achieved to pass the module. Failure to achieve this minimum will result in a failing grade for the entire module.*
   - *Thus at least two exercises must be correctly solved, and one must be from the more advanced ones.*
3. We give **bonus points** for students who have completed more than 8 points from all the exercises.
   - *Thus **7 points** from the exercises is required, **8 points** is still normal passing, and **9 and above** would give 1 or more extra bonus points.*
   - *Only the bonus points can and will be added directly to the final exam.*
4. Do not copy assignments, tools to detect copying and plagiarism will be used.
   - *The exercise results are an integral part of the final course grade and therefore the handed in attempts and solutions to the exercises **must be your personal work**.*

Submission Rules

- Submitted code must run without errors using the indicated Python environment, using the included libraries, packages, and frameworks.
- The whole project source code must be zipped and submitted before the given deadline, including the output results (saved in .html file or as a screenshot picture).
- Submit your .zip archive named *dva_ex1_Firstname_Lastname_MATRIKELNUMBER.zip* (e.g. dva_ex1_Hans_Muster_01_234_567.zip) through the OLAT course page.
- You zip file should contain:
  - Your working code as main.py file
  - A screenshot or export of your tool as .jpg, .png, .pdf or .html
  - An readme.txt if you needed additional libraries and good reasons why you needed them.
- **Deadline is Sunday, 24 April 2022 at 23:59h**

Exercise 2

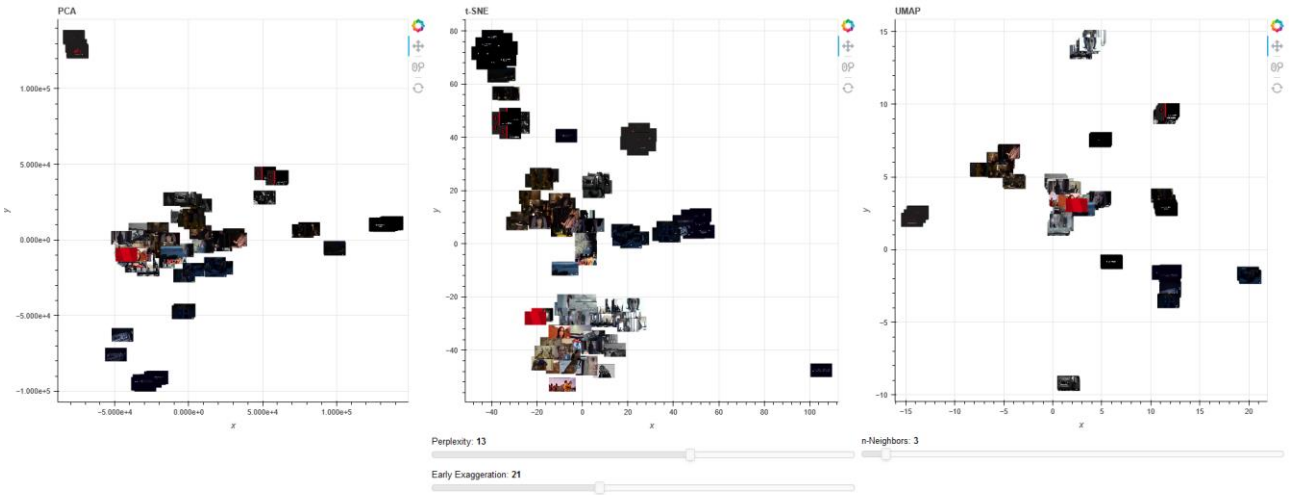
Dimensional reduction techniques are an important tool when working with datasets, which have more dimensions than we can convey using the coordinates, colors, shapes, or other visual cues. In such cases, a dimensional reduction technique may be applied to the high-dimensional dataset to project it into the low-dimensional visualization space (2D in our case since we have a two-dimensional coordinate system).

The aim of this exercise is to practice different dimensionality reduction processes and to learn how to plot them. Your task will be to read the color information from images, process and filter it and present the images based on their color content in a dashboard, using three different but connected plots. The arrangement of the widgets in your dashboard should look like the figures below. Exact tasks and hints are provided in the code skeleton.

Important: To run the application, you will need to deploy it as a *directory format* application, such that the images can be loaded by the visualization. In essence you must make sure that the python script is named **main.py** and you run in the directory format:

```
# bokeh serve --show .
# python -m bokeh serve --show .
```

Otherwise the images will not appear in your visualization. More information about the directory format of the bokeh server can be found here: https://docs.bokeh.org/en/latest/docs/user_guide/server.html#directory-format



Screenshot: How the resulting tool should look, please note that results in the plots may vary.

Task 1: Preprocessing:

First you need to compute a high-dimensional **color histogram**, which you will later input into the dimensional reduction method. Essentially, a color histogram is computed by splitting the complete RGB color space, which is a cube, into a number of smaller cubes (bins) and then counting for each bin, how many pixels of an image fall into it according to their color value. Luckily, NumPy provides such a multi-dimensional histogram functionality. A color histogram with 16 bins per edge would thus result in a high-dimensional feature vector with $16^3 = 4096$ bins. To place the histograms into the 2D visualization space, we will apply two different dimensional reduction techniques, namely t-SNE, UMAP and PCA, for which we will use the scikit-learn and umap libraries.

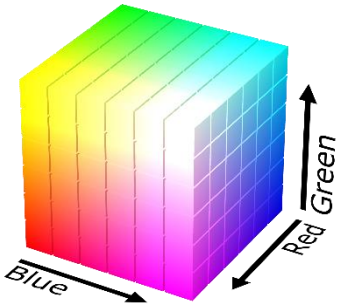


Figure 1: A color histogram splits the complete color space into a set of smaller cubes.

Task 2: Visualization

The visualization is comprised of three different plots for the results of PCA, t-SNE and UMAP respectively. Both, t-SNE and UMAP expose a variety of hyperparameters, which we allow the user to adjust “interactively” to tweak the final plot.

Important: All deliverables of this exercise must be submitted before the deadline. The absence of any required files will automatically lead to a **FAIL**.

Important: You must run the code with the bokeh folder application:

```
python -m bokeh serve --dev --show .
```

Remarks:

- Try to make good use of the hints and references provided in the skeleton code. (**very important**)
- Try to google first for any Python related issues/bugs.
- Due to the special situation, we don’t arrange in person meeting in this semester. Please contact me (Gaudenz Halter, halter@ifi.uzh.ch) for technical questions regarding the exercise only as a last resort.
- More than one day late submission will not be accepted and graded.
- If the submitted solution works as expected, full points will be rewarded, else, we follow the points indicated in the template
- **Online office hour** will be on **Friday, 8 April 16:00 – 17:00**. The zoom link will be communicated in that week, prior to the office hour, via email.