

Assignment 4

Dimensionality reduction and clustering.

IAML 2020

Note: Please read the whole assignment carefully before starting to solve it. This includes the introductory text which may be updated for each assignment!

This is the last assignment of the course and focuses on unsupervised machine learning techniques as well as some core mathematical concepts for the course. The assignment has three parts:

1. **Theoretical questions (week 1):** Answer several questions related to the mathematical aspects of the course.
2. **Dimensionality reduction (week 1+2):** Perform dimensionality reduction of face shape data using PCA.
3. **Clustering (week 3):** Implement the bag of visual words method for face images using various clustering techniques.

Due to the structure of the assignment, your report will be more divided between assignment sections than in the last assignment.

On average, we expect students to use 10 hours for solving the exercises. This time is in addition to time used on lectures, reading, and exercises. This is why it is important that you use the resources available to you to get the help you need. Don't hesitate to contact us by email, on Slack, or on LearnIt - we are here to help you.





Hand-in

You have to hand in a collection of the assignments at the end of the course. The following elements have to be included:

- **Code:** You have to hand in the complete source code used for the assignments **including any files provided by us**. Use a separate folder for each assignment. This makes it easy for us to test whether your implementations actually works. Upload the code as a **zip** archive (don't use rar).
- **Report:** You must combine your writings for each part into a single report document in **pdf** format. The reports for each part should be approximately 2-3 pages in length (normal pages of 2400 characters). Use clear formatting for assignment numbers. The report should be a single coherent text that is self-contained (no need to look at other files to understand the content) and reproducible (it must be possible for other students to redo what you did from your description).
- **Results:** Include all pictures, videos, and other files produced in a separate **zip** archive. Use a separate folder for each assignment.

Requirements

We specify a number of requirements for the assignment - these have to be completed to pass the exam. Completed here means attempted and described in the report. However, completing each item is not a guarantee for passing the exam. The final assessment is based on an overall evaluation of your code and report. To make the requirements as clear as possible, we mark tasks with one of the following symbols:

-  The task is required.
-  You have to implement at least one of the specified tasks.
-  The task is entirely optional. You should only focus on these after you completed the other tasks.
-  The task is required for master students but is of course allowed to be attempted by bachelor students as well.

Do not expect that you will always get optimal results. Neither should you focus on optimising your code for speed unless explicitly specified. The assignments are generally open ended and we encourage you to improve your solutions as much as possible.

For questions regarding the general course requirements and differences between bachelor and master students, refer to the official course descriptions.

Guidelines


We provide a few general guidelines here.

- **Be as precise as possible:** Be as precise as possible when explaining your approach, implementation, and results.
- **Reproducibility:** Your results should be reproducible from the report description alone.
- **Use the theory:** Relate your findings to theoretical concepts from the course when relevant. Don't repeat the theory unless specified, use references instead.
- **Remember that you are learning:** We don't expect you to be fluent in all the material. Solve the tasks to the best of your abilities and write what you did so others can reproduce your results from the descriptions.


Assignment 4.1*Theoretical questions*

The first section contains exercises that you have to solve by hand - using a computer won't be of much help anyway since the exercises require an understanding of theoretical concepts used in the course from linear algebra and calculus.


You may either write your answers on a computer or by hand. If you write by hand you have to scan the documents and include them in your report. For this section you may simply answer each question without trying to write a coherent piece of text. However, you still have to explain yourself - a correct answer without adequate explanation will detract from the overall assessment.

1.  Answer the following:

- (a) Given $A = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{bmatrix}$, $A^n = \begin{bmatrix} a_1^n & 0 & 0 \\ 0 & a_2^n & 0 \\ 0 & 0 & a_3^n \end{bmatrix}$. Derive a few steps of this result and explain why it keeps this simple form.
- (b) Calculate $A\mathbf{v}$ given $A = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$.

2.  Below are multiple sets of vectors in \mathbb{R}^3 . For each set, determine whether they (a) are linearly independent, (b) what their span is, and (c) whether they are a basis of \mathbb{R}^3 or not.

- (a) $\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$
- (b) $\left\{ \begin{bmatrix} 1.5 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 \\ 4 \\ 0 \end{bmatrix} \right\}$
- (c) $\left\{ \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$
- (d) $\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$

3.  Let $T = \begin{bmatrix} t_1 & t_2 \\ t_3 & t_4 \end{bmatrix}$ be an unknown 2×2 transformation matrix and $\mathbf{p} = \begin{bmatrix} x \\ y \end{bmatrix}$, $\mathbf{p}' = \begin{bmatrix} x' \\ y' \end{bmatrix}$ be two-dimensional points such that $T\mathbf{p} = \mathbf{p}'$. Answer the following questions:

- (a) What type of transformations can T represent?
- (b) How many points are needed to solve $T\mathbf{p} = \mathbf{p}'$ for T ?
- (c) Set up a system of linear equations (in matrix form) that can be used to solve $T\mathbf{p} = \mathbf{p}'$.



4. Below are pairs of image matrices f and convolution kernels k . Calculate the convolution $f \star k$ for each pair.

(a) $f = \begin{bmatrix} 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 2 & 0 & 1 & 0 \end{bmatrix}, k = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

(b) $f = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, k = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$

5. Find the derivative for the following functions wrt. x . Use the chain rule:



(a) $g(f(x))$

(b) $\frac{f(x)}{x^2}$

(c) $f(g(x), h(x))$

6. Find the derivative $\frac{df}{f x_i}$ of the function

$$f(x_1, \dots, x_n) = \prod_i x_i$$



7. For this exercise, you will use the chain rule to derive the gradient for a basic linear model. The model function is defined as $\hat{\mathbf{y}} = \mathbf{w}^T \mathbf{x} + b$ and the loss function is $\ell(\hat{\mathbf{y}}, \mathbf{y}) = (\hat{\mathbf{y}} - \mathbf{y})^2$. Find the partial derivative of $\ell(\hat{\mathbf{y}}, \mathbf{y})$ with respect to \mathbf{w} , i.e. find $\frac{\partial \ell}{\partial \mathbf{w}}$.



Assignment 4.2

Dimensionality reduction

In this exercise, you will use *principal component analysis* (PCA) to embed face shape points in a low dimensional space, referred to as the principal component space. You will use the principal components to generate new faces by manipulating vectors in the principal component space. You will also measure how much of the information is lost when transforming existing face points to and back from the principal component space.

We will work with the *IMM Frontal Face Database* created at DTU¹. It contains 120 facial images of 12 participants. For each image, a total of 73 facial landmarks have been manually annotated. A few examples are shown in Figure 1.

¹ [Link to website.](#)

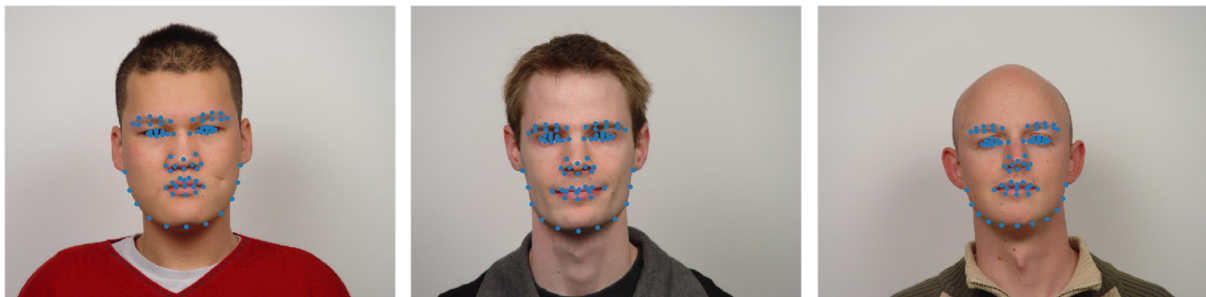


Figure 1: Samples from the IMM Database with annotated points visualized as blue dots.



1. **⚠ Download the dataset:** The dataset can be found on the IMM website or downloaded directly at the following [link](#). Unzip the data and move the folder to the same folder where the assignment files reside.
2. **📄 Get an overview of `util.py`:** Read the documentation for the functions in `util.py`. You will need to use some of these in the next part.

If you are interested, the webpage contains more information on how the data has been recorded and annotated. It has many interesting uses beyond this assignment.

Task 1: Implementing PCA


Your first task is to implement the PCA method as well as functions for transforming to and from the space defined by the principal components. But first, a quick recap of the terminology to minimize confusion.


Principal component analysis is about finding a linear transformation that reduces the number of dimensions used to represent samples, while destroying as little of the variation as possible. PCA is defined by W , an $C \times M$ matrix representing a linear transformation from vectors in M -dimensional space to C -dimensional space. We have the following transformations

$$x' = Wx, \quad (1)$$

$$x + \epsilon = W^T x', \quad (2)$$


where $x \in \mathbb{R}^N$ is the input vector and $x' \in \mathbb{R}^C$ is the embedded vector. As shown in equation 2, it is possible to reconstruct x with some amount of error ϵ . To find W , we use the *eigenvectors* of the covariance matrix of our data matrix X . To construct W you use an arbitrary number of the eigenvectors sorted by their associated eigenvalues.

1.  **Setup:** Create a script file or notebook for this exercise. In it, start by loading the face shapes and images using `util.face_shapes_data`.

2.  **Implement PCA:** Create a function that calculates and returns the principle components of the shapes dataset. Use the method described above where the eigenvectors of the covariance matrix is used. **Make sure to center the samples (subtract the mean before calculating the covariance matrix).**

Hint: The reading material for the PCA lecture contains an excellent tutorial on how this can be done ([link](#)), but remember that copying is not allowed!!

Hint: Some of the later tasks will be easier if you return all 146 principle components. You can then create another function for extracting n components to generate W .

3.  **Implement transformations:** Create two functions, one for transforming from feature space to principal component space (equation 1) and one for transforming from principal component space to feature space (equation 2). You have to subtract the μ vector when transforming to the principal component space and add it again when transforming back to feature space. You may use the following modified equations for reference:

$$x' = W(x - \mu) \quad (3)$$

$$x = W^T x' + \mu \quad (4)$$

Task 2: Evaluating precision

As described above, using PCA to transform a sample x to a principal component space and back again incurs some error ϵ . This error is called the *reconstruction error*. In this task you will implement a method for calculating this error and use it to test the effect of increasing or decreasing the number of principal components used to construct W .

When solving regression problems, the error is typically measured as the average distance error, otherwise known as root mean square error (RMSE). This is also used when calculating the construction error. For reference, the RMSE is

$$RMSE(x, \tilde{x}) = \sqrt{\sum_i (x_i - \tilde{x}_i)^2}, \quad (5)$$

where x, \tilde{x} are the original and transformed samples respectively.

Another method for evaluating PCA models is to look at the eigenvalues, where eigenvalue i is denoted $\lambda^{(i)}$. The eigenvalues explain the variance of each dimension when that data has been transformed by PCA. The sum of all eigenvalues $\lambda^{(1)} + \dots + \lambda^{(n)}$ is equal to the total variance of the data. By comparing all the eigenvalues we can calculate:

- (1) **Proportional variance:** What proportion of the total variance is explained by a single component. The following formula can be

used

$$\frac{\lambda^{(i)}}{\lambda^{(1)} + \dots + \lambda^{(n)}}$$

- (2) **Cumulative proportional variance:** What cumulative proportion of the total variance is explained by the first k components.

$$\frac{\lambda^{(1)} + \dots + \lambda^{(k)}}{\lambda^{(1)} + \dots + \lambda^{(n)}}$$



1. **Calculate reconstruction error:** Implement a function in your script that calculates the reconstruction error given a dataset X , principle components W , and a mean vector μ .



2. **Plot reconstruction error:** When constructing W you may use a single principal component or all of them. Plot the reconstruction error of W for all possible numbers of principle components. An example is shown in (REF).



3. **Calculate variance:** Create functions that calculate the *proportional* and *cumulative proportional* variance.



4. **Plot variance metrics:** Plot both the proportional and cumulative proportional variance in a single plot. An example is shown in Figure 2.

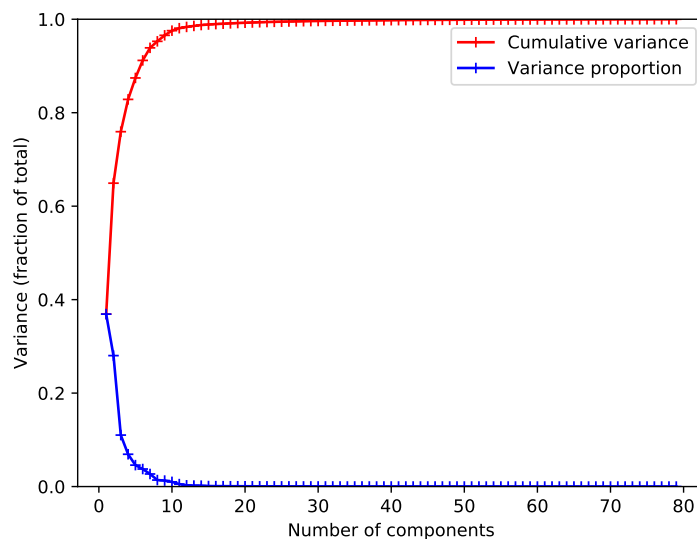


Figure 2: The expected result of the cumulative and individual variance proportion.

Task 3: Generative PCA

As shown in the lecture, PCA can be used for generative processes, where new samples similar to the training data X are desired. This requires choosing a x' instead of finding one by transforming a real sample. By varying the components of x' it is possible to explore

what information each principle component encodes. We provide an interface (shown in Figure 3) for easy exploration of the effect modifying x' has on the resulting face shape.

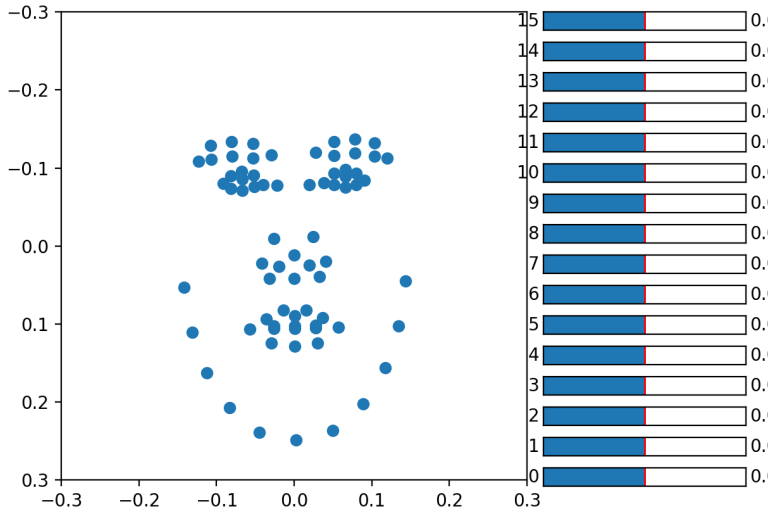


Figure 3: Preview of the interface used to explore generating face shapes. The sliders on the right control values for each component of the vector that is transformed to face-shape space using the inverse pca transform.

To get reasonable values, we start at $x' = \mathbf{0}$ which is the same as using the average shape of the dataset X since it is also $\mathbf{0}$. We use the variance in each dimension to define a reasonable range of values. Specifically, for each vector component x_i , a range of $[-3\sigma, 3\sigma]$ is used, where $\sigma = \sqrt{\lambda^{(i)}}$. If the data is normally distributed, $\pm 3\sigma$ covers 99.7% of all points. This is shown in Figure 4. We will not go into more detail on this choice – suffice it to say that the normal distribution is typically a good initial assumption in these circumstances.

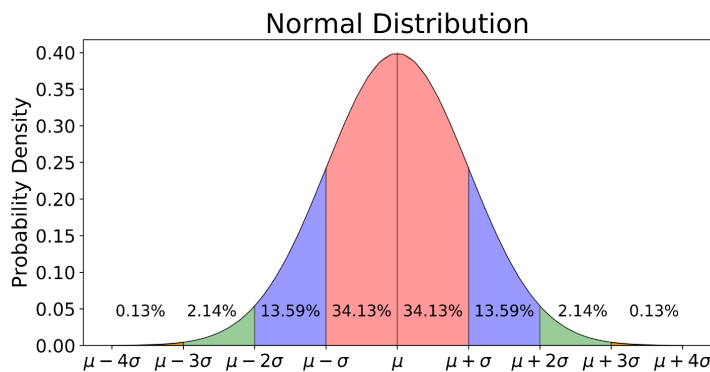









Figure 4: Illustration of a normal distribution. The shaded regions indicate the area of the distribution covered by 1, 2, 3, and 4, standard deviations (σ) respectively.

1. **Exploring the space of the principal components:** Read the documentation for `SliderPlot` in `util.py`. It is a helper class for setting up and updating a Matplotlib window that allows you

to adjust the components of a vector in the principal component space. Create an instance as described in the documentation and call `plt.show()` after instantiation to show the window.

2.  **Varying the number of components:** Experiment with varying the number of components used when creating W ? Try at least four configurations. Relate this to the previous task.

Task 4: Report

1.  **Approach/method:** Describe in precise and academic language how you approached the problem and implemented the solution. Don't include your code but remember that your method should be reproducible by reading your report.
2.  **Results/evaluation:** Describe and reflect on the results gathered from various parts of the assignment. Specifically, answer the following questions (and back up your arguments with results or theory):
 -  • For the task about generative PCA, how did changing component values change the generated face points? Does this match your expectations?
 -  • What did you think of the results from Task 3? Explain why so few principal components are needed to reproduce the samples quite accurately.
 -  • Relate the results from Task 3 to the visual results from Task 2.
 -  • Why does the reconstruction error approach 0 much more gradually than the cumulative proportional variance approaches 1?

Checklist

- ☐ A script file or Jupyter notebook with your implementation as described above.
- ☐ Plots of variance and reconstruction errors.
- ☐ Report section as specified.

Assignment 4.3

Clustering

In this exercise you will use the bag-of-visual-words technique to identify people. An diagram visualizing the main components of the technique is shown in Figure 5.

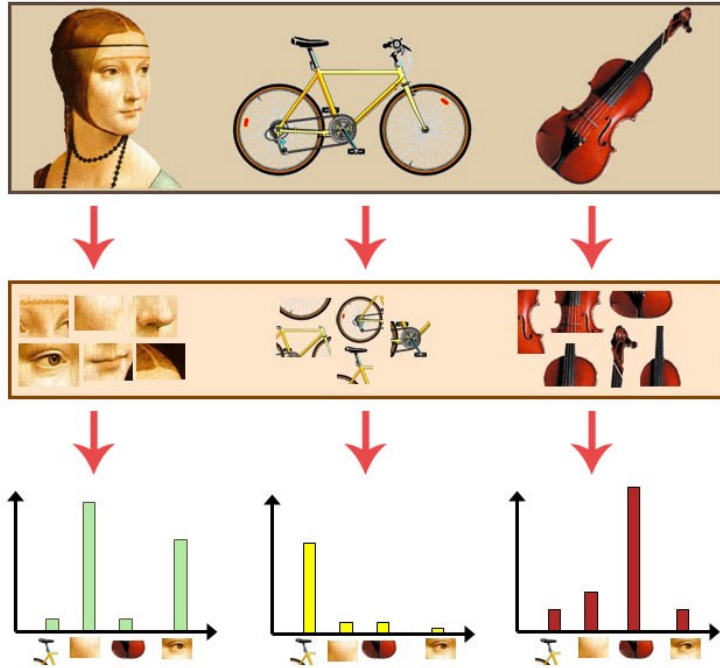


Figure 5: High-level diagram of the bag-of-visual-words technique. First, clusters are defined

The technique is divided into the following operations:

1. Decide on format of visual words. In this assignment, we use small and evenly spaced windows of image-patches.
2. Learn a set of visual words by applying a clustering algorithm to all image patches of the training dataset.
3. For a new image, classify its image-patches according to the identified visual words. The image is then characterised by its distribution of visual words.
4. This new representation is a vector in the space of possible distributions. By examining the distance between images it is possible to determine how similar they are. In this assignment, you will use this to find similar images but it has also been used effectively in object recognition tasks.

Task 1: Clustering setup

In the first task, you will learn a set of visual words using clustering as mentioned.


1.  **Create a new script:** Create a script or notebook for this exercise.



Figure 6: Sample extracted clusters for 10×10 windows and 5×5 stride using 0.1 for the `scale` parameter of `read_image_files`.



2. **⚠ Load data and generate windows:** Load the face images using the `read_image_files` function in `dset.py` - use the `scale` setting to adjust the scale for faster computation. Use `image_windows` from `util.py` to create a list of windows for each image. Finally, merge all windows into a single data matrix.

Hint: Use large stride settings for testing, as it makes the clustering process much faster



3. **⚠ Train clustering models:** Implement code for fitting a Scikit-learn `sklearn.cluster.KMeans` model to the window data.

Hint: Use `np.vstack`.



4. **⚠ Plot image results:** Use the function `plot_image_windows` from `util.py` to plot the cluster centers found by the k-means model. They are accessed as `<model_obj>.cluster_centers_`.



5. **⚠ Try other clustering methods:** Try training on at least `sklearn.cluster.MeanShift` and `sklearn.cluster.AgglomerativeClustering` as well and compare the output clusters.

Task 2: Calculating the distribution for an image

In this task, you will use visual words to find images that are similar to each other. First, you will use the visual word clusters found in the last task to create a word histogram for each image. Similar images can be found by finding histograms that are geometrically closest to the source.

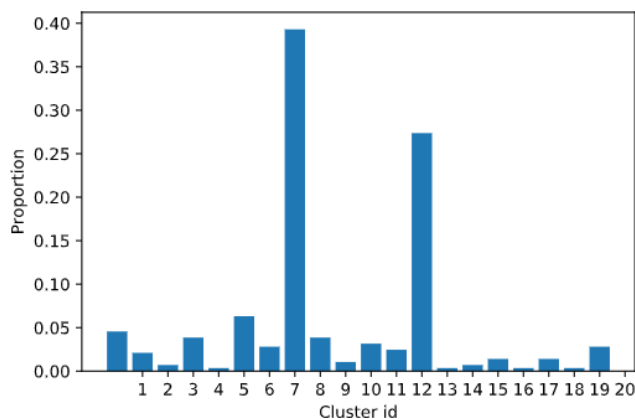






Figure 7: Distribution of clusters for a single image in the dataset.








1. **⚠ Predict clusters for each image:** For each image in the dataset, use the trained model to predict (`<model_obj>.predict`) a cluster for each window. The result is an array of cluster indices. Use this array to create a histogram for each image, representing the distribution of window clusters in that image. Figure 7 shows an example. Create a list with all the histograms.



2. **⚠ Set up neighbor search:** Scikit-learn provides the class `sklearn.neighbors.NearestNeighbors` for finding nearest neighbors easily. Create an instance of the model and use its `fit` method to initialise it to the list of histograms, i.e. call `<model>.fit(<histograms>)`.

3.  **Finding closest neighbors:** Use the `NearestNeighbors` method `kneighbors(x, k)` to find the k nearest neighbors to the histogram x .

4.  **Experimentation and visualisation:** Experiment with using different images for x . **Visualize the resulting neighbors and their histograms.** Compare results for the different clustering methods and different window sizes and window strides.


Task 3: Report

1.  **Approach/method:** Describe in precise and academic language how you approached the problem and implemented the solution. Don't include your code but remember that your method should be reproducible by reading your report.

2.  **Results/evaluation:** Describe and reflect on the results gathered from various parts of the assignment. Specifically, answer the following questions (and back up your arguments with results or theory):
 -  • How does the cluster results compare visually across algorithms? Is there any reason to believe any of them are superior at this stage? Use visualizations (plots of histograms and sample nearest neighbor outputs) in your report.
 -  • How accurate is the visual word method for finding images of the same person? If other people are found, are they similar in appearance to the target? Compare differences between the three clustering algorithms and various window sizes and stride combinations (you only have to compare clustering methods on one size, then experiment with sizes only for a single method).

Checklist

- ☐ A script file or Jupyter notebook with your implementation as described above.
- ☐ Figures showing samples of images and their nearest neighbors.
- ☐ Sample plots of distributions.
- ☐ Report content as specified.