Group assignment 1:
Crafting and learning features

Computer Vision (H02A5a)

The goal of this assignment is to explore more advanced techniques for constructing features that better describe objects of interest and to perform face recognition using these features. This assignment will be delivered in groups of 5 (either composed by you or randomly assigned by your TA's). Start from the provided *template-notebook.ipynb* on Toledo. The notebook you submit for grading is the last notebook you submitted in the kaggle competition prior to the deadline on **Wednesday 10 April 23:59**. Make sure that this notebook is well documented. Good luck and have fun!

1 Overview

In this assignment you are a group of computer vision experts that have been invited to ECCV 2024 to do a tutorial about "Feature representations, then and now". To prepare the tutorial you are asked to participate in a kaggle competition and to release a notebook that can be easily studied by the tutorial participants. Your target audience is: (master) students who want to get a first hands-on introduction to the techniques that you apply.

In the first part of this assignment, you will start from a **small** dataset and construct/learn features that describe the data. We discriminate two types of features, handcrafted features and features learned from data. Handcrafted features are properties that can be algorithmically extracted from the image (similar to previous assignment). In this assignment you will experiment with Histogram of Oriented Gradients (HOG) or Scale Invariant Feature Transform (SIFT) feature descriptors. The features learned from data are features that are learned by finding patterns in multiple images. You will explore Principal Component Analysis (PCA) in the context of face detection, so-called eigenfaces. In the second part of the assignment you will use the constructed features for face recognition using your favorite classifier. In the third part of this assignment you will participate in a kaggle competition and attempt to improve your classification pipeline in any way you please. Lastly, you will reflect on the entire assignment and write a final discussion, this is on top of the discussion and comments that you already provided in the previous parts. In summary, the assignment is broken down into four main parts:

1. build feature representations using handcrafted and non-handcrafted techniques (Sect. 4);
2. use and compare the feature representations in context of classification of faces (Sect. 5);
3. attempt to improve your model and participate in the kaggle competition (Sect. 6);

4. discussion (sect. 7).

2 Preparation: learning to use a machine learning library (TensorFlow/Keras)

If you choose to build a deep learning network in section 5 or 6, we recommend using the TensorFlow/Keras [Ten; Cho+15] library. Keras is a high-level neural networks API written in Python. Since the TensorFlow 2.0 release, Keras has been integrated more tightly into TensorFlow. While there are many good libraries available for building deep learning models, we recommend Keras as it is easy to use and still offers flexibility to make more complicated models at a later stage. Familiarize yourself with TensorFlow/Keras using for instance this tutorial, created by François Chollet.

3 A faces dataset

In this assignment we use a small subset of the VGG Face Dataset [PVZ15]. The subset contains images of four celebrities, more specifically:

- 30 images of Jesse Eisenberg;
- 30 images of Mila Kunis;
- 10 images of Michael Cera (who arguably looks similar to Jesse Eisenberg);
- 10 images of Sarah Hyland (who arguably looks similar to Mila Kunis).

Use OpenCV to detect the faces in the images and extract the faces (this code is already provided in the template notebook but feel free to improve on it).

Hint: split this dataset (hold-out or cross-validation), being methodologically correct is in your best interest!

4 Build feature representations

Once you know what defines an object or what allows you to differentiate between two objects, it becomes much easier to detect an object in an image. Learning a robust (same features are extracted from the same object in different conditions) and discriminative (different image objects can be easily separated from each other in feature space) feature representation is key to perform automated object detection effectively. Two big classes of feature representation methods can be identified:

- Handcrafted features (Sect. 4.1)
- Features learned from data (Sect. 4.2)

You will try building both types of features on the faces dataset and use them as baseline.

4.1 Handcrafted features

Handcrafted feature descriptors are extracted from the image itself, using a specific deterministic algorithm that might differ from task to task. To assess if a handcrafted feature is discriminative we compute it for a few images, we then compare the match score (e.g. euclidean distance between the feature representations) and observe if this matches our intuition/domain knowledge. Ideally, a (local) feature descriptor holds the robust and discriminative property and will additionally be invariant to e.g. illumination, pose, scale. This allows you to recognise objects of interest in virtually any image using that descriptor. Handcrafted features were commonly used in traditional machine learning approaches but we still see them today in complicated machine learning tasks e.g. doing machine learning on a manifold mesh.

Compute features from pixel intensities in order to learn a representation of local structures present in the images. You can choose between one of these two state-of-the-art handcrafted image feature descriptors (feel free to use e.g. OpenCV for this):

- Histogram of Oriented Gradients (more details can be found here)
- Scale Invariant Feature Transform (see course book SB 4.1.2)

Carefully choose the hyperparameters. Keep in mind: how local should the descriptors be for the object that you want to detect; how to make your descriptor behave well in different circumstances (e.g. image with different lighting)? Visualize your feature representations to gain more insight. To do this, use the high dimensional data visualisation tool t-SNE[1] [MH08]), an example is shown in figure 1. Reflect on the parameters you've chosen. How does this feature compare to your previous grabbing task in the individual assignment? Did you need specific pre-processing steps before computing these feature descriptors on your images (which ones and why)? Did the visualisation show good discriminative and robustness properties?

4.2 Learning features from data: PCA

An alternative approach to handcrafting features that capture desired object properties is to learn these features from training data. In this case, the algorithm looks for patterns in the data that are informative.

Apply Principal Components Analysis (PCA) to find the different components of variation present in your faces training set and use the scores to form a feature vector. Think about this problem mathematically:

- How to convert my image (you can work in color if you like) dataset to a 2D matrix?
- Can you exploit the dimensionality of this data matrix to make your computations more effective?
- Mean subtraction or not?

[1] Scikit-learn has an implementation.

- Shall we use eigenvalue or singular value decomposition?
- How many non-zero eigenvalues/singular values should we have?

Attempt to choose an *optimal* number p of principal components such that the dimensionality of the feature space is reduced but still informative (e.g. based on the average reconstruction loss of your training images). Project all images onto these same p-components. Visualize the reconstructions of one face using gradually more eigenfaces and visualize the faces on the feature space using the first two principal components, an example is shown in figure 2. Reflect on your choice of p and how this might have influenced the performance of the features that you learned. Did you need specific pre-processing steps before computing these feature descriptors on your images (which ones and why)? How many non-zero eigenvalues did you have, why is this? Did the visualisation show good discriminative and robustness properties?

5 Classification

Now it's time to use the feature representations that you have constructed and start to classify images. In this exercise you will train a classifier that can recognise person Jesse Eisenberg (class 1) and Mila Kunis (class 2) or recognise that neither of them are present in the image (class 3). This can be achieved using 2 binary-classifiers (a Jesse recogniser and a Mila recogniser) or you can use one multi-classifer, the choice is yours! Feel free to use your favorite classification algorithm, you can use existing functions from open source Python libraries (e.g. scikit-learn) or implement them yourself. Evaluate the features built in Sect. 4 and report the mean classification accuracy on Jesse/Mila test images. How did you build this classifier for each of your feature representations? Do the models hold up when you predict on images of Michael Cera and Sarah Hyland?

Hint: Don't forget to apply the same pre-processing pipeline for each feature representation as you did before!

6 Improve performance

Now that you are familiar with the basics of computer vision and classification scenarios and have dipped your feet in traditional techniques it is time to experiment with the state of the art.

Improve on the parts of your pipeline that will have the biggest impact on your performance (face detection, feature extraction and/or classification model) and participate in the kaggle competition. There are no rules for this section, you can try anything you please, just keep in mind that your data is limited. Good places to start: transfer learning, data augmentation, loss functions, incorporate prior knowledge, ... Clearly document your zero-to-hero journey in the submitted notebook: Which iterations did you go through? Do you notice improvements? Please ensure that the TAs get a clear understanding of what you are doing from the notebook as this will be our basis for evaluation your work: no documentation ⇒

lower grade.

Note: This competition is still in educational context, therefore the grades earned in this section do not only depend on the final performance achieved in kaggle but also on your ingenuity, methodological correctness and insight. The kaggle ranks are mainly intended for you to track the progress of your improvements and get an idea of what results others achieve on this task.

7 Discussion

Summarize what you have learned in this assignment. Discuss qualitative differences between the different feature representations that you constructed. Why/when would one work better than the other? Why is a performances sub-optimal? What would you do better/more if you would have plenty of time to spend on each step?

8 Practicalities

Make sure that your notebook is **self-contained** and **fully documented**. Walk us through all steps of your code. Treat your notebook as a tutorial for students who need to get a first hands-on introduction to the techniques that you apply. Provide strong arguments for the design choices that you made and what insights you got from your experiments. Make use of the *Group assignment* forum/discussion board on Toledo if you have any questions. Assignment submission and participation are performed though kaggle.

To get started: **CLOSELY follow the** instructions written by your TAs on how to join and participate in the kaggle competition. The final deadline is **Wednesday 10 April 23:59**, any submissions after the deadline will be disregarded.

References

[MH08] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[Cho+15] François Chollet et al. *Keras*. 2015.

[PVZ15] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. "Deep Face Recognition". In: Section 3 (2015), pp. 41.1–41.12. DOI: 10.5244/c.29.41.

[Ten] Google Tensorflow. *Tensorflow*. URL: www.tensorflow.org (visited on 05/22/2016).

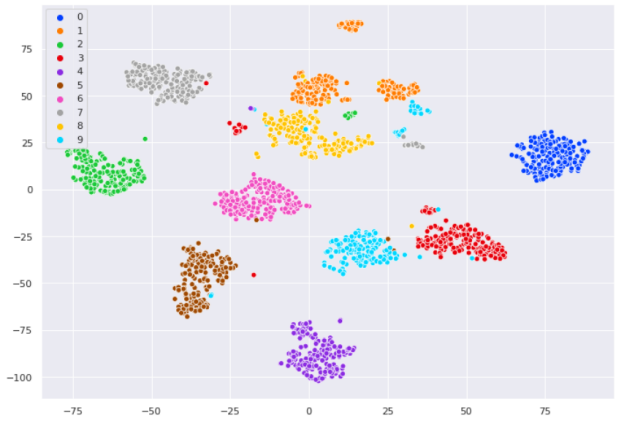

Figure 1: T-SNE plot (example from https://towardsdatascience.com/t-sne-python-example-1ded9953f26.

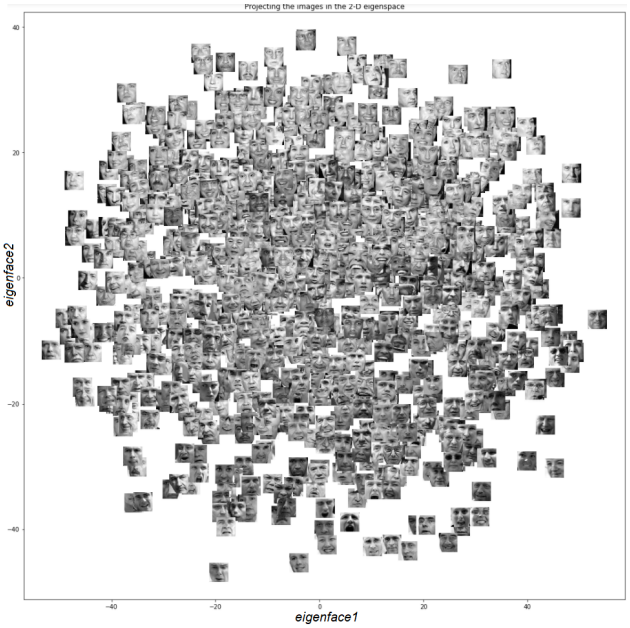

Figure 2: The face-feature plot (example from https://sandipanweb.wordpress.com/2018/01/06/eigenfaces-and-a-simple-face-detector-with-pca-svd-in-python/.