Computer Vision Project
WiSe 2023/24
Florian Kordon, Mathias Seuret,
Vincent Christlein, Thomas Köhler, Mathias Zinnen

Sheet 3, starting from Nov 20th, 2023, due Dec 11th, 2023

Introduction and General Comments: We will provide you with

- (1) label files for the training and test set (`icdar17_labels_train.txt` and `icdar17_labels_test.txt`) which contain for each image a class index (i.e., the ground truth writer label), and (2) for each image a `.pkl.gz` file which contains already the local descriptors of each file. Please download the files from (8.7GB):

  `https://faubox.rrze.uni-erlangen.de/dl/fiMbHQuNMw76y8muZPMBjZ/icdar17_local_features.zip`

  Note: in case of quota issues: use your personal ciptmp-folder, i. e. `/proj/ciptmp/<yourlogin>`, it offers you 10GB (no backup). If the folder for your login doesn't exist, create it on your own. (If you cannot see the ciptmp folder, you have to enter it first — it is a shared network drive).
- We also provide in StudOn a code skeleton ('skeleton.py'), which we recommend to use.
- Test your algorithms with different parameters and try to get a deeper understanding regarding their behaviors. Show your implementation and your results to one of the advisors.

Exercise 3.1: Writer Identification

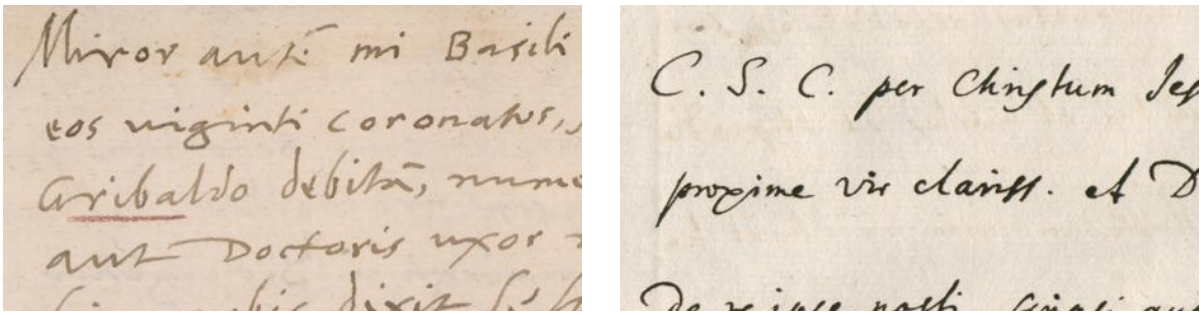

Abbildung 1: Image excerpts of the ICDAR2017 Historical WI Dataset: 21-IMG_MAX_966414 (left), 21-IMG_MAX_1006529 (right).

In this exercise, we will implement a small writer identification system. It is going to be evaluated using the ICDAR17 Historical WI benchmark dataset. The dataset is divided into a training set (a. k. a. experiment set) and a test set (a. k. a. benchmark set). Please download the dataset and the code skeleton.

The *bag of (visual) words* model for image recognition/retrieval is going to be used, where each image is represented by a global vector representation, which is formed by encoding the local feature descriptors. The global descriptor is eventually used for the classification/retrieval.

We will use VLAD encoding, which later gets extended by means of *generalized max pooling*. Given T local descriptors $\mathcal{X} = \{\boldsymbol{x}_t \in \mathbb{R}^D | t = 1, \ldots, T\}$ and a dictionary (or codebook) $\mathcal{D} = \{\boldsymbol{\mu}_k \in \mathbb{R}^D | k = 1, \ldots, K\}$, the VLAD encoding (i.e. VLAD embedding + sum pooling, see lecture) for one cluster is: $\boldsymbol{\xi}_k(\mathcal{X}) = \sum_{t=1}^{T} \alpha_k(\boldsymbol{x}_t)(\boldsymbol{x}_t - \boldsymbol{\mu}_k)$. The overall encoding is: $\boldsymbol{\xi}(\mathcal{X}) = (\boldsymbol{\xi}_1^T, \ldots, \boldsymbol{\xi}_K^T)^T \in \mathbb{R}^{DK}$.

a) Codebook generation

For computing the residuals ($\boldsymbol{x}_t - \boldsymbol{\mu}_k$), we need a codebook to generate the dictionary $\mathcal{D}$. To do so, a representative number of descriptors needs to be selected. Therefore, use a random selection of about 500 000 descriptors from the *training* features.

Compute a codebook (a. k. a. vocabulary/background model) with the help of k-means ($k = 100$). For this operation you can use sklearn's `MiniBatchKMeans`.

b) VLAD encoding

With the help of the computed dictionary, we can now compute the residuals (see above). They are multiplied with the indicator/association function $\alpha_k(\boldsymbol{x}_i)$, which returns 1 for the nearest cluster center and 0 otherwise. Therefore, you can use `cv2.BFMatcher` and `knnMatch`.

Suggestion: compute first the full association matrix (i.e. a vector for each descriptor. Each element of the vector represents the association to a cluster. That means, set all elements of the vector to 0 except the nearest cluster center). This can then be used to obtain for each cluster only those descriptors that have this cluster assigned. Then, compute the residuals with these descriptors and aggregate them to one global descriptor.

Compute the pairwise distances between all encodings and evaluate the performance in terms of mean average precision (mAP).

c) VLAD normalization

In order to improve the representational power of VLAD encoding, we should account for visual burstiness, i.e. frequent local descriptors that dominate the overall similarity. This can be achieved by using *power normalization*: $\xi_i' = \text{sign}(\xi_i)\sqrt{|\xi_i|},\ \forall\{\xi_i\} \in \boldsymbol{\xi}$, where sign is $+1$ if $\xi_i \geq 0$ and -1 otherwise. Don't forget to ℓ^2 normalize, i. e., divide by the ℓ^2 norm, the representation again afterwards. Compute the pairwise distances between all normalized encodings and evaluate the performance in terms of mean average precision (mAP) and compare with the result before.

d) Exemplar classification

The last step of the pipeline is the proposed exemplar classification. Therefore, you have to compute an individual SVM for each global representation (encoding) of the *test* set using this encoding as positive and all the encodings of the training set as negatives. (sklearn's `LinearSVC`), a good C value is 1000 (actually this value should be cross-validated with a different validation set), also set `class_weight` to `balanced`. Afterwards, ℓ^2 normalize the weight vector (`coef_` parameter of the SVM) and use it as your new global descriptor.

Compute the pairwise distances between all newly created descriptors and evaluate the performance in terms of mean average precision (mAP) and compare with the result before.

e) Bonus for 10 ECTS: Extract your own features

We want now to compare the pre-computed descriptors with traditional SIFT descriptors. Therefore, create a `computeDescs` method, which takes a filename as input, loads the filename and computes SIFT descriptors at SIFT keypoints. As improvements, we will modify (1) the keypoints: set all keypoints' angles to 0; (2) SIFT descriptors: apply a Hellinger normalization (this is in essence similar to a power normalization). However, because the SIFT descriptors are already ℓ^2-normalized, we have to first apply an ℓ^1 normalization and afterwards take the sign square root of each element of the vector (no ℓ^2 normalization afterwards required).

Afterwards, modify the `loadRandomDescriptors` and `vlad` methods.

You also need now to use the original images of the dataset, which can be found here:

`https://zenodo.org/record/1324999`

Compare your results with the given descriptors.

f) Bonus for 10 ECTS: Generalized max pooling

To further improve the result, we will now exchange the sum-pooling part of VLAD encoding with generalized max pooling. Therefore, you can use sklearn's ridge regression (`Ridge`) with the conjugate gradient solver (`sparse_cg`, note: `fit_intercept` can be set to False, `max_iter` to 500, you can try out different regularization values, standard is $\gamma = 1$). You can apply the ridge regression component-wise for each cluster to safe some computation time (since the residuals are independent from each other). The encoding lies then in the `coef_` parameter of the ridge regression classifier.

Compute the effect of GMP with and without the exemplar classifiers.

g) Bonus for 10 ECTS: PCA whitening and multi-VLAD

Another trick to improve the results is to counter the random effects of k-means. Therefore, run the clustering five times with different seeds (best also to use a different subset of descriptors each time) to get 5 different codebooks. For training set (size T): compute five VLAD encodings using the five different codebooks and concatenate them along the 1st (not 0th) axis. You should now have $T \times (5 \cdot 128 \cdot 100)$ large matrices, this is unfortunately quite big, so let's use principal component analysis to reduce the number of components to 1000 (max would be 1182), also use whitening! Train the PCA on the training set and apply it then on the individual files. In this way, you apply a joint dimensionality reduction and de-correlation procedure.

Compare the effect of multi-VLAD with VLAD. Which feature benefits the most of it? What works better for you? E-SVM or multi-VLAD or a combination of both?

Submission Please submit your code `exercise3.py` and a `exercise3.txt` file in which you report the results for each of the experiments.