UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Jan Aare van Gent

Clustering Faces from the National Archives of Estonia

Bachelor's Thesis (9 ECTS)

Supervisor: Tambet Matiisen, MSc

Tartu 2018

Clustering Faces from the National Archives of Estonia

Abstract:
The National Archives of Estonia contains hundreds of thousands of images, where many of the people are unlabeled. To assist manual labeling, a label prediction system is proposed that detects faces from images and uses an artificial neural network to generate a representation for each face in the form of a feature vector. These feature vectors can be clustered using euclidean distances since similar faces produce feature vectors that are close to each other in feature space. In this thesis, a proof-of-concept of the pipeline for face clustering and label prediction was created. Clustering results were evaluated with different clustering parameters and stages of the pipeline were profiled in terms of time taken for execution and average memory consumption. A survey of different methods for face detection, shape prediction, face alignment, feature extraction, clustering, and labeling was also presented.

Keywords: clustering, machine learning, face images, face representation, artificial neural network, face detection, National Archives

CERCS: P175 Informatics, systems theory

Eesti Rahvusarhiivi digiarhiivi nägude klasterdamine

Lühikokkuvõte: Eesti Rahvusarhiivi kogud sisaldavad sadu tuhandeid digiteeritud pilte, kuid enamik isikuid nendel piltidel on märgendamata. Käsitsi märgendamise lihtsustamiseks pakutakse töös välja automaatse märgendamise süsteem, mis põhineb piltidelt tuvastatud nägudel. Iga leitud näo jaoks leitakse tehisnärvivõrkude abil tunnusvektor ning need tunnusvektorid klasterdatakse eukleidilise kauguse alusel. Töös on realiseeritud süsteemi prototüüp, mis sisaldab näotuvastust, klasterdamist ja märgendite automaatset määramist. Lisaks hinnati klasterdusalgoritmi parameetriks oleva raadiuse mõju tekkivate klastrite ühetaolisusele ja täielikkusele. Samuti hinnati protsessi erinevates etappides kasutatud aja ning mälu sõltuvust piltide arvust. Töö sisaldab ka ülevaadet näotuvastuse, näopunktide määramise, nägude joondamise, tunnuste eraldamise, klasterdamise ning automaatse märgendamise meetoditest.

Võtmesõnad: klasterdamine, masinõpe, nöo pildid, näo representatsioon, tehisnärvivõrk, näotuvastus, rahvusarhiiv

CERCS: P175 Informaatika, süsteemiteooria

Contents

1 Introduction

Face recognition has recently seen a surge of popularity due to the amount of face images uploaded to social media sites or captured by surveillance systems. This in turn incentivized researchers to develop new methods for face verification and classification [SOJ18]. The advent of the convolutional neural network (CNN) and the residual neural network for image classification and verification tasks has paved a new way for creating accurate face representations that are robust under unconstrained backgrounds, pose variation and different illuminations [HZRS15]. These new types of representations (also called feature vectors or embeddings) are able to pick up very high level facial features, which is not the case for the older hand crafted feature representations [SWT14].

Although facial representations based on CNN architecture have seen a lot of progress in recent years [SWT14] [SCWT14] [SLWT15] [SKP15], relatively little work has been done on clustering these representations based on identity [SOJ18]. One way identity based clustering can be leveraged would be for the use of an identity label prediction system, where the goal would be to suggest facial images that contain a specific individual or simply similar faces. This type of label prediction can come in handy when tasked with manually labeling a large data set of images, such as images from the National Archives of Estonia.

The goal of this thesis is to implement a proof-of-concept pipeline for face clustering and face label prediction that could be used for easing facial image labeling for the National Archives of Estonia. Most of the algorithms required will not be implemented from scratch in this thesis, since every stage or algorithm needed for the pipeline has well-tested open source implementations in various image processing and machine learning libraries, mainly Dlib and Scikit-learn.

The first chapter gives insight into each stage of the pipeline from a theoretical perspective and presents possible solutions for each problem including the solution selected for this thesis and the reasoning behind it. The second chapter describes the implementation of the pipeline created for this thesis. The third chapter evaluates the clustering results and the performance of the pipeline with different clustering parameters.

2 Pipeline

Connecting pictures of faces to individual identities can be divided into 4 main stages: face detection, face alignment, extraction of features, and clustering. A key feature of this pipeline is its modularity: any of the stages can be substituted, if the need arises, by other more effective methods. This makes the software solution easier to manage and provides future-proofing to a certain extent.

2.1 Face Detection

Face detection is the process of finding faces on an image and returning a rectangular bounding box for every face found in the image. The bounding box can simply be thought of as the 2D positions of the four corners of the detected face on the image. In addition to the algorithm used, the effectiveness of face detection is heavily dependent on lighting, image quality, and orientation of the face. Even though there is an abundance of face detection algorithms, only a small portion of them can work with unconstrained backgrounds and grayscale images (as is the case with the images from the National Archives) [Fri]. These include the very popular (but slightly dated) weak classifier cascade by Viola and Jones [VJ04], the Histograms of Oriented Gradients (HOG) classifier by Dalal and Triggs [DT05], and multiple deep neural network (DNN) based detectors. In this thesis, the Max-Margin Object Detection (MMOD) [Kin15] implemented with a CNN instead of the HOG feature extractor, is used [Kin16]. This was chosen mainly because of the good accuracy and availability of pretrained CNN weights for human face detection. It is important to note that choosing the best face detector is not the goal of this thesis, since most state-of-the-art face detectors find enough faces for later clustering.

2.2 Face Alignment

To actually cluster images of faces, they need to be transformed to feature vectors. For that, a pretrained DNN is used. However, faces with different orientations will look very different to the network, since proportions will be very different. That means the faces need to be aligned prior to feature extraction, which is done with the face landmark estimation algorithm.

The process of face alignment can be divided into two parts: identifying the geometric structure of the face and translating, scaling and rotating the face to standardize its position [Ros17]. Face alignment algorithms consist of 2D and 3D alignment methods. 3D alignment methods attempt to impose a 3D model of a face on to the image with the help of face landmarks, while 2D methods directly transform the image to align the landmarks. Although 2D methods are currently more popular, 3D methods are making a comeback, such as seen in the implementation of DeepFace [TYRW14] by Facebook. The 2D algorithms can be further subdivided into Active Appearance Model

(AAM) [ETC98], Constrained Local Model (CLM) [CC06] and regression methods, the latter of which are most commonly used. An ensemble of regression trees [KJ14] model using 5 facial landmarks (2 for each eye and one for the nose) was chosen for this thesis. The decisive factor behind the decision was that it has been trained with our previously chosen MMOD face detector. Since different face detectors position face bounding boxes differently, using alignment models with a detector it wasn't trained with will reduce accuracy [Kin].

2.3 Feature Extraction

Feature extraction refers to generating a feature vector (also called embedding) from the face, so that feature vectors of similar faces (i.e. the same person) are more similar than vectors of different faces. These vectors can be thought of as points in a space - the closer the points are, the more similar the faces and vice versa. A distance metric is used to quantify the similarity or dissimilarity of two feature vectors. It can be thought as a function f(x,y), where x and y are feature vectors, that results in the measure of distance or dissimilarity of the pair of vectors. The Euclidean distance metric is used most often for this purpose. It goes without saying that the best distance metric is dependent on the feature extraction algorithm.

The earliest forms of face representations are holistic by nature: features are found by processing the face as a whole, in contrast to localized features, which look at individual parts of a face separately. Holistic methods include the Principal Component Analysis (PCA) based on "Eigenfaces" [TP91], the more robust "Fisherface" method [BHK97], Linear Discriminant Analysis (LDA) [EC97], and the 2D PCA [YZFY04]. However, localized or component-based feature representations are more robust to pose variation, which makes their usage more attractive for unconstrained images of faces [HHWP03]. Component-based representation methods include the eigenfeatures method [PMS+94], Local Feature Analysis (LFA) [PA96], Elastic Bunch Graph Matching (EBGM) [WKKVDM97], Local Binary Pattern (LBP) [AHP06], Scale-Invariant Feature Transform (SIFT) [Low99], and the faster Speeded-Up Robust Features (SURF) [BETVG08]. Nevertheless, face descriptors learned using deep learning (also known as deep features) are quickly becoming the standard for face representation, since they are able to pick up higher lever features than any handcrafted face descriptor could [SWT14] [SCWT14].

One way a deep metric can be learned is by training a CNN to be an identity classifier on a very large data set such as the CASIA WebFace Database and then using the output of an intermediate network layer as the embedding for the face [SWT14]. This assumes that a CNN classifier trained with a large variety of faces to classify a very diverse group of people has learned an effective and compact representation of the face in one of its layers that precedes the final classification layer.

Another way would be to use face verification instead of identification. While face identification is the process of finding a certain individual from multiple faces, face

verification refers to deciding if two images contain the same identity or not. In this case, the network is trained on pairs of faces that are labeled either positive (same identity) or negative (different identities). If the pair is positive, the loss function discourages the embeddings to be further apart than a specified threshold, and if the pair is negative, the network is reconfigured such that the embeddings are further than a given threshold [SWT13]. The training could also be done using the triplet loss which, during training, takes in a triplet of pictures where two of the faces belong to the same person and the third to a different person, as was done in FaceNet by Google [SKP15]. The resulting network trained in this way will have learned non-linear transformations to transform the image into a feature space [HLT14].

A third type of approach used for DeepID2 and DeepID3 attempts to combine both identification and verification for metric learning [SCWT14] [SLWT15]. Even further improvements can be made by removing weak neuron connections from the network and retraining it, which improves verification accuracy [SWT16]. It is, however, very difficult to estimate, which CNN architecture or metric learning method is superior, since the amount, quality, and diversity of the training data are all important factors.

For this thesis, a deep metric which was learned using the verification method was chosen. It was trained with a structured metric loss that tries to project all the identities into non-overlapping balls of radius 0.6 [Kin17]. The network generates a 128-dimensional embedding from a face that is well suited for clustering purposes. It was chosen for its availability of pretrained weights and good results during initial testing.

2.4 Clustering

Once an embedding for each face has been generated, they are clustered based on their similarity. Ideally, every cluster would contain only all faces of one individual, but this is difficult to achieve, as it depends on the embeddings, the clustering algorithm used, and the amount of data to be clustered.

It could be that the DNN used for generating the embeddings has overfitted to training data such that the embedding might depend on factors like image quality, pose of the face, glasses worn, facial hair, emotions, the background, and the color space (gray-scale or colored) of the image. It could also be that the network was underfitted, so that the distance between embeddings of a father and his grown up son, or brothers in the same age range, are negligible.

Even if all these problems were addressed during training and the network generalizes well, facial features change over time as people age. Therefore a picture of an older person would generate an embedding that is more similar to other people of the same age than to a picture of the same person when they were younger. Assuming that pictures of a person in a certain narrow age range generate embeddings which form a spherical cluster (as they should in our case), and as the person ages, their face embeddings move in a generally similar direction in the feature space, then embeddings of that person

will, as their age varies, create an elongated cluster also in that direction. Since the direction the embeddings move during aging might change, the cluster can be concave or "banana-shaped". The only way we could cluster all embeddings of the same person from different age ranges is if we have enough data from all ages of that person to form an elongated cluster, which is not the case for the archive images. Therefore, one person may have multiple clusters correlated to their faces of differing ages.

Despite that the deep metric used in this thesis was trained by pushing faces of the same identity into non-overlapping balls of radius 0.6, there is still a chance that some or many clusters will end up being more or less concave due to insufficient data. If we imagine a person's face embeddings creating a convex spherical cluster, we can also imagine that we only have the data points for a small part of the surface of the sphere and as a result, we get a shell-shaped non-convex cluster. This means the clustering algorithm must be able to handle concave clusters in addition to simple convex ones.

Another consequence of insufficient data is that we might only have embeddings on the opposite sides of the theoretical spherical cluster so that embeddings on one side are closer to embeddings of an other identity cluster than to the opposite side of the same cluster. Since the underlying distribution of embeddings in unknown, we can not know which cluster parts belong to the same identity cluster. This introduces the precision-recall trade-off to our problem - while we could make the clusters so large that the cluster will contain all embeddings of an individual in a specific age gap, this will involuntarily include embeddings of other people into the identity cluster. A more concise description would be that the clusters have high recall but low precision.

The resulting clusters, in the case of high recall, would be quite disparate in size as many individuals are represented only once in the entire image data set, creating a lot of single element clusters, while others might be represented in hundreds of images. Having high recall would greatly amplify this effect and would also put a new constraint on the clustering algorithm, as it must be able to handle the very contrasting size differences of the clusters. Furthermore, this would mean that the proposed label prediction system would find large but impure clusters of a person, which the user then has to clean by removing false-positives (faces that are in the same cluster but belong to different individuals) before assigning the cluster to an individual manually.

A high precision and a low recall, on the other hand, would make the clusters very small and numerous, but the probability of all embeddings in a cluster belonging to the same individual would be high, making most of the clusters "pure". This has the added benefit that the user does not have to clean the clusters beforehand and just has to merge the suggested smaller clusters. Finding false-positives in these clusters would require only a quick glance, compared to maybe tens on hundreds of faces for high recall clusters. Having smaller clusters also does not constrain our algorithm to consider large size differences, both, in terms of number of embeddings and feature space volume. In fact, manual user feedback for Google Picasa, which also has a face label prediction feature,

indicates that people would rather merge pure (high precision) clusters than manually clean impure (high recall) ones [ZKM13]. In addition to Picasa, Apple iPhotos and Microsoft Photos also seem to work similarly. Since this trade-off primarily depends on the preferences of the user of the software, the choice of the cluster size, in the work of this thesis, will be left for the user to decide.

Choosing a clustering algorithm that is robust to differences in shapes, sizes and amount of clusters is important since they all depend on how accurate the embeddings are and how much detail is in each face to generate an accurate embedding. There are many clustering algorithms that do not need prior information relating to those details, however, the density based clustering algorithm known as the density-based spatial clustering of applications with noise (DBSCAN)[EKS+96] seemed to perform better than other algorithms in terms of both recall and precision. DBSCAN had the added benefit of having only one parameter that needed to be fine-tuned, which was the radius of the surrounding neighborhood of each data point. If the data point doesn't have a label assigned to it, it would get clumped in the same cluster as data points in it's radial vicinity and if it doesn't have any neighbors, it would get a new unique cluster label. This of course assumes that the second parameter of DBSCAN that describes the number of neighbors needed to be clumped into the cluster, is 0, since it should be possible for a cluster to be made up of only a single data point. The cluster propagation effect of DBSCAN also allows for clusters of any kind of shape, such as elongated and concave shapes. All of these advantages led to it being chosen as the clustering algorithm in this thesis. It is possible that a different clustering algorithm would work better if a different feature vector generator were to be used.

2.5 Labeling

The pipeline described so far simply clusters the faces. The main purpose of the pipeline is, however, label prediction. When all faces have been clustered, the question of how to implement identity labeling and make use of existing identity labels arises.

Since using labeled data in conjunction with unlabeled data is a problem of semi-supervised learning (SSL), it is possible to adapt a SSL clustering algorithms for label prediction. Sticking to the rule of making use of open source and well tested implementations of algorithms, the label propagation algorithm is promising: it takes in a similarity matrix and a list of labeled data labels and outputs the label probability distributions for each data point [ZG02].

The similarity matrix that is used by the label propagation algorithm can be thought of as a nearest neighbor graph of faces. Firstly, every node is initialized with a unique label. Then, at each time-step, each node is processed and is given a label that occurs the most frequently among its first degree neighbors. In the case of a tie, if one of the labels is a hard label (manually labeled), that is the label that propagates to the node being processed. If none of the neighboring nodes have a hard label and there is still a

tie, a random label from the neighboring labels is selected.

Making use of all the previously mentioned algorithms, a fairly simple label prediction system can be designed that can do two things: find similar faces to a labeled face and recommend labels for unlabeled faces based on nearby labeled faces. First of all, it is assumed that all faces have been clustered and each face has a cluster ID associated with it. If no faces have been labeled, the user can label a face within a cluster, preferably one that is a representative of the cluster in some way. Once a face in a cluster is labeled, the label propagation can be ran on that cluster, giving every member of the cluster, other than the labeled one, a label distribution probability, where the label with the highest probability for that point is its soft label. A soft label is simply a label that has not been manually assigned and can still change when processed by label propagation. Whenever another point in the same cluster is manually labeled differently (or given a different hard label), the label propagation is re-run in that cluster with two hard labels instead of one. If the cluster was, for example, evenly distributed and the hard labeled points are on the opposite sides of the cluster, the label propagation algorithm effectively splits the cluster in two by giving half of the points the soft label of the first hard label and the other half the other soft label. The reason for this kind of setup is that it contains the effect of label propagation to only nearby faces in a single cluster. If the user wishes to find unlabeled but similar faces to a labeled face, then faces that have the same soft label can be presented as candidates for hard labeling. However, if the user instead wishes to find a label for an unlabeled face, the nearest neighboring faces from some nearest-neighbor graph can be analyzed and the most common/closest labels (both hard and soft) are recommended. The nearest neighbor graph can either be radius based or k-nearest neighbors (k-NN) based.

3 Final implementation

The final pipeline implementation was created in Python using the libraries Scikit-learn 0.19.1, Dlib 19.10.0 and all of their dependencies (Numpy, Scipy, etc.). The pipeline was split into 3 separate independent programs: feature extractor, clustering program, and label propagator. The pipeline also makes use of SQLite3 to store all necessary information related to the faces so that the pipeline can easily be incorporated into any SQL database system such as MySQL, PostgreSQL, etc.

3.1 File structure

The first section of the pipeline is implemented in "serialize_features.py", which does face detection, shape prediction, face alignment, and feature extraction. The second section that is responsible for clustering is implemented in "cluster_features.py". Label propagation is implemented in the file "propagate_labels.py". The folder "dats" contains the pretrained Dlib models for the face detector, the shape predictor and the DNN that extracts the feature vector of the face. The files "face_models.py" and "linkages.py" are modules that contain utility functions and classes.

```
workspace/
├── serialize_features.py
├── cluster_features.py
├── propagate_labels.py
├── face_models.py
├── linkages.py
└── dats/
    ├── dlib_face_recognition_resnet_model_v1.dat
    ├── mmod_human_face_detector.dat
    └── shape_predictor_5_face_landmarks.dat
```

3.2 Face detection

The first stage of the pipeline involves finding all faces from all images that are in a specified folder, which is given as a parameter for the face detection program. Each ".jpg" file in the input folder is first converted into RGB format so that all images have 3 color channels. Then the image is processed using a face detection model, which returns a list of bounding boxes of the faces found in the image. The detector model is deserialized into the Dlib detector class via the Dlib function "cnn_face_detection_model_v1" with the default file path parameter "dats/mmod_human_face_detector.dat" which specifies the path to the object detector file. However, any Dlib face detection model file can be specified as the detector model if specified as a program argument.

3.3 Feature extraction

Each bounding box (in combination with the original image) is given to the shape predictor which returns the facial landmarks or face shape. The shape predictor model path can also be specified as an argument for the program just like the face detector. If a shape predictor is not specified, it defaults to using the model with the file path "dats/shape_predictor_5_face_landmarks.dat" which is the only currently available pretrained shape predictor model that works with Dlib's MMOD face detector. The landmarks, along with the original image, get fed into the deep feature DNN model described by Dlib's "face_recognition_model_v1" class with the default model file path being "dats/dlib_face_recognition_resnet_model_v1.dat". During processing, Dlib internally aligns the face using affine transformations before extracting the embedding from it.

Once the bounding boxes and embeddings have been found, an SQL table called "Faces" is created. Every face entry will get an auto generated ID. The first important attribute is the ID of the original image, which references and image entry. The reason for this is two-fold: it makes querying faces on every image easier and allows for saving the relative face dimensions in the entry as another attribute. The face chip, which is the part of the image that contains the face, is a rectangular upright shape, which means it can be completely described using just four numbers or two 2D points referencing the upper left part of the face and the lower right part respectively. The next attribute is the embedding, which is the 128-dimensional floating point array which will be used in the clustering stage.

3.4 Clustering

The clustering program is a separate program that can take a database file as input, but defaults to using the database created by the feature extractor program with a default name. The program then retrieves all embeddings and clusters them using the DBSCAN algorithm implemented in Scikit-learn. The new cluster labels are then stored in the corresponding face entry's "cluster_id" attribute.

3.5 Labeling

The label propagator program takes the face ID and new label as input, then assigns the face the new label in the "label_id" and performs label propagation on the cluster containing the given face. It also has the functionality of suggesting labels in a given radius for any given face ID. The label propagation used is implemented in Scikit-learn.

4 Results

In order to measure how well the pipeline performs, two aspects of the pipeline will be examined and evaluated: the clustering results and performance.

4.1 Evaluation of clustering

Since there are too many faces to individually label and then compare the results, only a small fraction of faces, which have been labeled beforehand, will actually be looked at to evaluate the precision and recall of the resulting clusters. Firstly, the faces included in the ground truth set will be processed and the embeddings of the faces stored in the database, along with a path to their original image file, the relative position of the bounding box on the image and the file path to a chip of the face. Then, the tested data set is processed similarly and appended to the database and the resulting embeddings will be clustered. Although images in the ground truth set are already contained in the tested data set, it is difficult to figure out which face in the real data set corresponds to which face found in the ground truth set. This also does not affect clustering results, since if the embeddings are exactly the same, they do not propagate the cluster further.

After clustering, the first section of the database, which contains only faces from the ground truth, is examined and its cluster labels compared to the ground truth labels. The main evaluation metrics used are the homogeneity and completeness scores which correspond to the precision and recall of the clusters respectively, with the main difference that completeness and homogeneity take cluster entropy into account [RH07]. The adjusted Rand index (ARI), adjusted mutual information (AMI), and V-measure, which is simply the harmonic mean of completeness and homogeneity [RH07], are also used to describe the general quality of the clusters to find a good balance between precision and recall. The ARI metric is the adjusted-for-chance (random permutations will give a score close to 0) version of the Rand index (RI) that is bounded between -1 and 1 where RI is simply the ratio of correctly assigned pairs over the total number of pairs or combinations of data points [VEB10]. Likewise, AMI is the adjusted-for-chance version of the Mutual Information index, which is a measure of dependence between two distributions that takes entropy into account [VEB10]. The parameter that affects clustering results directly, is the DBSCAN's neighborhood radius ε. Therefore three values of ε will be found: one that produces pure (100% precision) clusters with the highest recall possible, one that produces complete (100% recall) clusters with the highest precision possible, and one that maximizes the general quality of the clustering with the highest ARI, AMI, and V-measure scores.

Table 1. Clustering evaluation with different radii for DBSCAN with 2838 faces.

ε	**Completeness**	**Homogeneity**	**ARI**	**AMI**	**V-measure**
0.31	0.536	**1.0**	0.043	0.111	0.698
0.32	0.634	0.991	0.329	0.297	0.773
0.36	0.833	0.872	0.663	0.680	0.852
0.37	0.869	0.854	**0.697**	**0.726**	**0.862**
0.38	0.911	0.804	0.695	0.662	0.854
0.43	0.973	0.601	0.535	0.436	0.743
0.44	**1.0**	0.531	0.479	0.374	0.694

4.2 Evaluation of performance

Performance measurements will focus on CPU time and memory consumption required for each stage separately. The High Performance Computing (HPC) cluster of University of Tartu was selected as the system on which the profiling will take place, since all of its technical specifications are publicly available [HPC]. It also has a lot of computing power and memory, which speeds up the process of evaluation considerably. The hardware used while profiling includes 4 central processing unit (CPU) cores and 2 GB of random-access memory (RAM) from the "Rocket" cluster of the HPC. To measure the performance, the Python libraries "line_profiler" and "memory_profiler" will be used. Only deterministic profiling will be used where the effects of every line of code are measured, in contrast to statistical profiling, where the instruction pointer or memory use are randomly sampled during the lifetime of the program [Pyt]. The main reason for it is that the scripts themselves are relatively short and contain few lines of code, which means that the effects of each line of code are more pronounced and hence should be monitored closely.

Three image databases of different sizes are used to measure the performance and profile the code, the first one with 240 images and 269 detectable faces, the second one with 2491 images and 2832 detectable faces, and the third one with 9722 images and 5170 detectable faces. The pipeline stages profiled include Dlib's MMOD face detection, shape or landmark prediction, feature extraction (which also encompasses the face alignment stage), and clustering with DBSCAN. The information about each stage includes the hit count, which specifies how many times it was executed during the lifetime of the program, the average time the execution took in milliseconds, the total time spent executing the stage, and the average increase in RAM used by the Python interpreter after the execution of that stage.

The results in table 2 are run with access to 4 CPU cores, but without a graphics processing unit (GPU), since it is shown simply for reference. It is evident, that Dlib's MMOD face detector is quite slow without the support of a GPU, since it takes on average over a second to analyze a single image. It also requires the most RAM, over 15x more than all the other stages combined. However, this would not be a practical

Table 2. Profiling results with 240 images and 269 faces. The column "hits" refers to the number of times the stage was executed. The "avg. time" and "avg. memory usage" refer to the time and memory required to execute the stage once, on average. "Total time" is simply the product of the average time taken and the number of hits.

Stage	Hits	Avg. time	Total time	Avg. memory usage
Face detection	240	1232.6 ms	295.8 s	160.7 MiB
Shape prediction	269	2.7 ms	0.7 s	0.5 MiB
Feature extraction	269	29 ms	7.8 s	5.1 MiB
Clustering (269 faces)	1	31.9 ms	0.03 s	1.2 MiB

problem, since face detection and feature extraction would be preformed only once per image and the results stored in a database for later use.

Table 3. Profiling results with 240 images and 269 faces using GPU.

Stage	Hits	Avg. time	Total time	Avg. memory usage
Face detection	240	39 ms	9.3 s	561.4 MiB
Shape prediction	269	2 ms	0.5 s	4.6 MiB
Feature extraction	269	7.5 ms	2 s	157.3 MiB
Clustering (269 faces)	1	31.5 ms	0.03 s	1.0 MiB

The results in table 3 show that using a GPU increases the face detector's performance by a factor of around 30. The feature extraction stage has also gotten a performance boost from the GPU by almost a factor of 4. A noticeable difference can be seen in the average memory usage of face detection, shape prediction and feature extraction, while the memory usage of clustering seem unchanged. Since the clustering stage does not allocate memory on the GPU, it is possible that the difference stems from how memory is allocated on the GPU.

Table 4. Profiling results with 2491 images and 2832 faces using GPU.

Stage	Hits	Avg. time	Total time	Avg. memory usage
Face detection	2491	36.6 ms	91.2 s	617.5 MiB
Shape prediction	2832	2 ms	5.7 s	4.5 MiB
Feature extraction	2832	6.4 ms	18.3 s	158.4 MiB
Clustering (2832 faces)	1	2900 ms	2.9 s	1.5 MiB

In table 4 it can be seen that the average times of face detection, shape prediction and feature extraction have not really changed when processing a much larger data set, however, the same can not be said about the RAM usage of face detection. A possible

explanation is that the images in the bigger data set had more faces in them on average and their resolution was higher, contributing to a higher use of memory. The memory usage of DBSCAN surprisingly didn't increase a lot, but the processing time increased by a hundredfold compared to only a tenfold increase in input data.

Table 5. Profiling results with 9722 images and 5170 faces using GPU.

Stage	**Hits**	**Avg. time**	**Total time**	**Avg. memory usage**
Face detection	9722	37.4 ms	364.1 s	618.1 MiB
Shape prediction	5170	2 ms	10.6 s	4.5 MiB
Feature extraction	5170	6.5 ms	33.7 s	159.9 MiB
Clustering (5170 faces)	1	26155 ms	26.1 s	4.0 MiB

From table 5 it is apparent how the time and memory of the clustering stage have a quadratic complexity and will be the main bottleneck when clustering with hundreds of thousands of faces. One way of mitigating this problem would be to do clustering as seldom as possible, so that, for example, if new faces are added to the clustering, only clusters that are in the near vicinity of that image are updated. It could also be possible to divide up the job of clustering between multiple CPU cores for added speed.

5 Conclusion

The goal of this thesis was to create a proof-of-concept pipeline for face clustering and face label prediction with the main goal of easing facial image labeling for the National Archives of Estonia. The pipeline was written entirely in the programming language Python and the algorithms for doing facial image processing and clustering were obtained from popular open source libraries, such as Dlib, Scikit-Learn and their dependencies. The resulting pipeline's clustering quality was then evaluated by comparing the results to labeled data, also known as the ground truth. In addition, each stage of the pipeline's performance and memory consumption were profiled with two different data sets to find the main bottlenecks. All evaluations were done on the HPC platform of University of Tartu.

The resulting pipeline consisted of the following stages: face detection, shape prediction, face alignment, feature extraction, and feature clustering. The face detection stage uses a modified MMOD object detection algorithm with a pre-trained model implemented in Dlib. For shape prediction, Dlib's 5-point pre-trained shape predictor was used. The face alignment stage uses affine transformations to transform the facial image such that the 5 face landmarks detected are centered. The aligned face is then fed into a pre-trained CNN that generates a 128-dimensional feature vector, also called a deep feature or embedding. The DBSCAN clustering algorithm implemented in Scikit-learn was then used to cluster the embeddings.

The aim of the cluster quality evaluation was to find the best clustering parameters to get clustering results with

- 100% recall with the maximal precision
- 100% precision with the maximal recall
- The best balance between recall and precision using different metrics like V-score, the adjuster Rand index and the adjusted mutual information index.

A 100% recall means that the facial images of a particular individual are all in the same cluster and a 100% precision means that facial images of different individuals do not appear in the same cluster. Since the best performing clustering algorithm was DBSCAN, only the neighborhood area radius ε was varied. The evaluation suggested that the most optimal values of ε for this particular pipeline setup and image data are 0.30, 0.37 and 0.44 (for 100% precision, best balance and 100% recall respectively).

The results of profiling showed that the most memory consuming part of the pipeline was face detection with the MMOD face detector since it used up around 600 MiB of RAM on average. However, when clustering hundreds of thousands of images, clustering will most likely be the main bottleneck in terms of required CPU time.

Although the face clustering pipeline developed for this thesis is not production ready since it suffers from performance issues when processing hundreds of thousands

of images, it provides a starting point and an overview of what could be accomplished using well tested open source implementations for the various parts of the pipeline. The description of each stage of the pipeline in this thesis also provides a basic theoretical understanding of the problems relating to clustering feature vectors of human faces. Any future work on this topic should focus on how to make better face label predictions for the user of the program, using labeled image data.

References

[AHP06] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, Dec 2006.

[BETVG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[BHK97] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, Jul 1997.

[CC06] David Cristinacce and Timothy F Cootes. Feature detection and tracking with constrained local models. In *Bmvc*, volume 1, page 3, 2006.

[DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[EC97] Kamran Etemad and Rama Chellappa. Discriminant analysis for recognition of human face images (invited paper). In *Proceedings of the First International Conference on Audio- and Video-Based Biometric Person Authentication*, AVBPA '97, pages 127–142, London, UK, UK, 1997. Springer-Verlag.

[EKS+96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[ETC98] G. J. Edwards, C. J. Taylor, and T. F. Cootes. Interpreting face images using active appearance models. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 300–305, Apr 1998.

[Fri] Robert Frischholz. Face detection algorithms and techniques. `https://facedetection.com/algorithms/`. Accessed: 2018-04-06.

[HHWP03] Bernd Heisele, Purdy Ho, Jane Wu, and Tomaso Poggio. Face recognition: component-based versus global approaches. *Computer Vision*

and Image Understanding, 91(1):6 – 21, 2003. Special Issue on Face Recognition.

[HLT14] Junlin Hu, Jiwen Lu, and Yap-Peng Tan. Discriminative deep metric learning for face verification in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1875–1882, 2014.

[HPC] Rocket cluster - high performance computing center, university of tartu. `https://www.hpc.ut.ee/en_US/web/guest/rocket-cluster`. Accessed: 2018-05-06.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[Kin] Davis King. Which face detector works with which landmark detector? `https://sourceforge.net/p/dclib/discussion/442517/thread/b53c57ac/?limit=25`. Accessed: 2018-04-07.

[Kin15] Davis E. King. Max-margin object detection. *CoRR*, abs/1502.00046, 2015.

[Kin16] Davis King. Easily create high quality object detectors with deep learning. `http://blog.dlib.net/2016/10/easily-create-high-quality-object.html`, 2016. Accessed: 2018-04-06.

[Kin17] Davis King. High quality face recognition with deep metric learning. `http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html`, 2017. Accessed: 2018-04-09.

[KJ14] Vahid Kazemi and Sullivan Josephine. One millisecond face alignment with an ensemble of regression trees. In *27th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, United States, 23 June 2014 through 28 June 2014*, pages 1867–1874. IEEE Computer Society, 2014.

[Low99] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[PA96] Penio S Penev and Joseph J Atick. Local feature analysis: A general statistical theory for object representation. *Network: computation in neural systems*, 7(3):477–500, 1996.

[PMS+94] Alex Pentland, Baback Moghaddam, Thad Starner, et al. View-based and modular eigenspaces for face recognition. 1994.

[Pyt] 27.4. the python profilers — python 3.6.5 documentation. https://docs.python.org/3/library/profile.html. Accessed: 2018-05-06.

[RH07] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007.

[Ros17] Adrian Rosebrock. Face alignment with opencv and python. https://www.pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/, 2017. Accessed: 2018-04-07.

[SCWT14] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems*, pages 1988–1996, 2014.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[SLWT15] Yi Sun, Ding Liang, Xiaogang Wang, and Xiaoou Tang. Deepid3: Face recognition with very deep neural networks. *CoRR*, abs/1502.00873, 2015.

[SOJ18] Yichun Shi, Charles Otto, and Anil K Jain. Face clustering: Representation and pairwise constraints. *IEEE Transactions on Information Forensics and Security*, 13(7):1626–1640, 2018.

[SWT13] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Hybrid deep learning for face verification. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1489–1496. IEEE, 2013.

[SWT14] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1891–1898, 2014.

[SWT16] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Sparsifying neural network connections for face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4856–4864, 2016.

[TP91] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[TYRW14] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[VEB10] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854, 2010.

[VJ04] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004.

[WKKVDM97] Laurenz Wiskott, Norbert Krüger, N Kuiger, and Christoph Von Der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):775–779, 1997.

[YZFY04] Jian Yang, David Zhang, Alejandro F Frangi, and Jing-yu Yang. Two-dimensional pca: a new approach to appearance-based face representation and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 26(1):131–137, 2004.

[ZG02] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. 2002.

[ZKM13] Liyan Zhang, Dmitri V Kalashnikov, and Sharad Mehrotra. A unified framework for context assisted face clustering. In *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*, pages 9–16. ACM, 2013.

Appendix

I. Glossary

AAM Active Appearance Model. 5

AMI adjusted mutual information. 13, 14

ARI adjusted Rand index. 13, 14

CLM Constrained Local Model. 6

CNN convolutional neural network. 4–7, 17

CPU central processing unit. 14, 16, 17

DBSCAN density-based spatial clustering of applications with noise. 9, 12–14, 16, 17

DNN deep neural network. 5, 7, 11, 12

EBGM Elastic Bunch Graph Matching. 6

GPU graphics processing unit. 14–16

HOG Histograms of Oriented Gradients. 5

HPC High Performance Computing. 14, 17

k-NN k-nearest neighbors. 10

LBP Local Binary Pattern. 6

LDA Linear Discriminant Analysis. 6

LFA Local Feature Analysis. 6

MMOD Max-Margin Object Detection. 5, 6, 12, 14, 17

PCA Principal Component Analysis. 6

RAM random-access memory. 14, 15, 17

RI Rand index. 13

SIFT Scale-Invariant Feature Transform. 6

SSL semi-supervised learning. 9

SURF Speeded-Up Robust Features. 6

II. Source code

The source code can be found here: `https://bitbucket.org/GaugeAnomaly/face-clustering/`

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Jan Aare van Gent**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis

   Clustering Faces from the National Archive

   supervised by Tambet Matiisen

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 14.05.2018