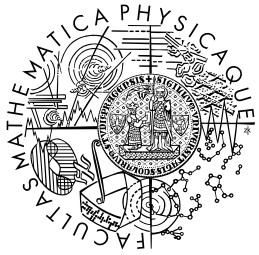


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Gregor Kovalčík

Comparison of signature-based and semantic similarity models

Department of Software Engineering

Supervisor of the bachelor thesis: RNDr. Jakub Lokoč, Ph.D.
Study programme: Computer Science
Study branch: Programming

Prague 2017

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In Prague date July 21, 2017 signature of the author

Title: Comparison of signature-based
and semantic similarity models

Author: Gregor Kovalčík

Department: Department of Software Engineering

Supervisor: RNDr. Jakub Lokoč, Ph.D., Department of Software Engineering

Abstract: Content-based image retrieval and similarity search has been investigated for several decades with many different approaches proposed. This thesis focuses on a comparison of two orthogonal similarity models on two different image retrieval tasks. More specifically, traditional image representation models based on feature signatures are compared with models based on state-of-the-art deep convolutional neural networks. Query-by-example benchmarking and target browsing tasks were selected for the comparison. In a thorough experimental evaluation, we confirm that models based on deep convolutional neural networks outperform the traditional models. However, in the target browsing scenario, we show that the traditional models could still represent an effective option. We have also implemented a feature signature extractor into the OpenCV library in order to make the source codes available for the image retrieval and computer vision community.

Keywords: similarity model, semantic model, image feature signature, deep neural network

I would like to thank my supervisor RNDr. Jakub Lokoč, Ph.D. for his advice, support and grammatical corrections during this project.

Contents

Introduction

Recent decades witness an enormous development of multimedia technologies, ranging from devices/sensors to sharing online services and large-scale multimedia management/retrieval systems. This thesis focuses on several content-based image retrieval (CBIR) tasks, representing a popular alternative to traditional keyword-based retrieval systems. The CBIR systems usually employ a similarity model, where images are transformed into descriptors (usually vectors in R^n), while their similarity is modeled by a distance measure defined over the descriptor universe. For several decades, many descriptors were designed as various aggregations of specific low-level features extracted from the images (Datta et al. [2008]).

Recently, deep learning started a revolution in the field of content-based similarity retrieval. The state-of-the-art deep convolutional neural network architectures have demonstrated that they can be employed as non-linear functions transforming images Img to highly effective image descriptors ($f_{DCNN-transform} : Img \rightarrow R^n$). In the text, we denote the descriptors as DCNN descriptors. Despite the fact that the networks were trained just on a limited set of semantic classes, the DCNN descriptors capture enough semantic information for retrieval in a much larger number of classes (Zhou et al. [2014]). This interesting level of generalization often enables effective applications of existing networks for different datasets, even without network adaptations/retraining. This trend is supported by available deep learning platforms (Caffe, TensorFlow), enabling to easily use the existing networks as black-box algorithms.

In this work, we focus on the comparison of traditional descriptors based on low-level features and the DCNN descriptors obtained from several deep convolutional neural networks. More specifically, we compare color-based feature signatures (Rubner and Tomasi [2001]) and DCNN descriptors (Wan et al. [2014]) on two image retrieval tasks. The first task is traditional query-by-example benchmarking, where the database of images is sorted with respect to a given query object and the mean average precision Beitzel et al. [2009] is evaluated for the ordered set (note that the used benchmark dataset contains class tags for each object). The second task is the target browsing scenario, where users cannot explicitly formulate their search intents, however, the users know the searched target image (e.g., the image was previously seen). Hence, users try to localize the target image by selecting suitable candidate objects (i.e., similar to the target) from the actual display.

The main contributions of the thesis are

- Comparison of several similarity models on different tasks.
- Implementation of color-based feature signatures into the OpenCV library.
- Implementation of SW components necessary for the experimental evaluations.

The thesis is organized as follows. Section 1 presents preliminaries for the formalisms used in the thesis. Section 2 provides details about the utilized models. Section 3 describes methodology of our experiments. We present and discuss

those experiment in Section 4. Finally, we provide a high level overview of our implementation details in Section 5. Section 5.3.2 concludes the thesis.

1. Preliminaries

This section presents the formal background for both considered image retrieval tasks – query-by-example search and target browsing.

1.1 Query-by-example Similarity Search

The query-by-example approach is a popular method to query image databases. As the users are interested in similar objects to the query example, a similarity model has to be utilized. The similarity is usually modeled by a pair (U, δ) (Zezula et al. [2005]). The set U represents a descriptor universe accommodating the set of all considered images I transformed by a suitable feature extraction function $f_e : I \to U$. The distance function $\delta : U \times U \to R_0^+$ models the similarity of two images by the distance between their descriptors. A lower distance represents higher similarity, and vice versa. Examples of the similarity models considered are summarized in the following section. The similarity search enables the sorting of the dataset with respect to the query example object.

Effectiveness evaluation

In order to compare various similarity models, effectiveness evaluation measures in connection with annotated benchmark datasets are considered. The similarity model is usually considered effective for a particular benchmark, if the most similar objects to the query have the same class label. The popular measure of effectiveness is the average precision, defined as the area under the precision-recall curve for a given query object (Zhang and Zhang [2009]). The mean average precision (MAP) (Beitzel et al. [2009]) is often automatically evaluated for a set of queries.

1.2 Similarity Browsing for Target Search

In order to find a searched target image, image retrieval systems often provide an interface for similarity browsing (finding similar objects to the already displayed ones). Given a similarity model (U, δ) and a large dataset $S \subset U$, the target browsing approach then results in a series of presented displays $D_0, \cdots, D_T \subset S$, where D_0 represents the initial display and D_T is a display with the searched image (Ferecatu and Geman [2009]). Each display D_i is further connected to a set of images M_i selected as mediators, so the browsing history can be formalized as a set $\{(D_i, M_i)\}_{i=0}^{T}$. The optimization objective of a browsing system is to minimize the number of displays T the user needs to localize the target image. The number of iterations T to find a target image in a browsing system corresponds to a search time and can be treated as a random variable. The goal is to find such settings of the system that maximize the growth rate of the cumulative distribution $F(t) = P(T \leq t)$ (in an ideal system $F(1) = 1$). In the experiments, $F(t)$ of an investigated system is estimated using m simulated search sessions of different target images.

During the last decades, many different approaches for effective target browsing were designed and investigated (Heesch [2008], Ferecatu and Geman [2009]). The evaluation of all the possible ways to approach target browsing is out of the scope of this thesis. In order to compare two orthogonal similarity models, we have selected a target browsing system incorporating coarse to fine browsing using similarity search. We consider k nearest neighbor query $kNN(q,S) = \{X \subset S; |X| = k \wedge \forall x_i \in X, \forall y \in S - X : \delta(q,x_i) \leq \delta(q,y)\}$, where $q \in U$ is a query object and $S \subset U$ is the dataset inspected. The browsing approaches and problems considered in this paper are summarized in the following subsections, with highlighted methods used in the experiments.

Effectiveness evaluation

As user's decisions are the essential part of the target browsing, the evaluation and optimization of target browsing systems is particularly difficult and highly time-consuming. User studies and evaluation campaigns have to be organized to compare different approaches and settings. As the number of all the possible combinations to investigate is too high, simulations of the browsing users can be designed to provide a first insight into the performance of the particular settings. In order to thoroughly simulate a browsing user, a complex model considering user perception, memory and decision making would be necessary. A simplified simulator can be designed assuming a level of coherence between the user's notion of similarity and the similarity model (U, δ) employed by the system. In Ferecatu and Geman [2009], two user types were considered as simulated baselines in the experiments.

- **Ideal user** – assumes no model usability gap. The user selects the most similar object (with respect to (U, δ)) from the actual display D_i to the target image.
- **Random user** – assumes no coherence. The user can select any image from the display with the same probability.

According to the results reported in [ref], the performance of the real user was somewhere between ideal and random users, closer to the ideal user. Hence, we also utilize a generalized variant of the artificial user, defined as:

- **k-coherent user** CU_k – assumes limited coherence. The user selects any image from the k most similar objects from the actual display D_i to the target image.

The ideal user corresponds to CU_1, while the random user is represented as $CU_{|D_i|}$. For $k \in (1, |D_i|)$, some levels of coherence are assumed. Note that in our simplified model each of the k images can be selected with the same probability. For more realistic simulations, a sophisticated parametric probabilistic model of image selection would be necessary, including thorough user studies for estimations of the parameters.

2. Compared Models

For both considered image retrieval tasks, similarity models based on DCNN features and feature signatures (representing classical low-level descriptors) are considered.

2.1 Feature Signatures

Beecks [2013] have introduced a formal framework, enabling a flexible representation of images. Given a feature space $\mathbb{F}$ (e.g., RGB color space), each image can be modeled as a finite set of representatives with corresponding weights. More formally:

Definition 1 (Feature Signature)**.** *Given a feature space $\mathbb{F}$, the feature signature S^o of a multimedia object o is defined as a finite set of tuples $\{\langle r_i^o, w_i^o \rangle\}_{i=1}^n$ from $\mathbb{F} \times \mathbb{R}^+$, consisting of representatives $r_i^o \in \mathbb{F}$ and weights $w_i^o \in \mathbb{R}^+$*

In our work, we consider two types of feature signatures. For target browsing simulations, we consider just image thumbnails (joint position-color feature space $\mathbb{F}$) enabling efficient comparison with the Euclidean distance. For the query-by-example benchmark, we consider a more flexible feature signature extraction algorithm employing a joint feature space comprising position, color and texture information extracted from sampled image pixels. We obtained the algorithm from Kruliš et al. [2016]. Note that one of the thesis objectives was the implementation of the CPU version in OpenCV library (Bradski [2000]).

The algorithm extracts low level features from a set of sampling points. Then a clusterization of the sampled features is performed and resulting cluster centroids, together with their weights, form the extracted feature signature.

Further details on the settings of the utilized signature extractor can be found in section 3.2.

We use Signature Quadratic Form Distance (SQFD, Beecks et al. [2010]) to compare two feature signatures in the query-by-example benchmark.

Definition 2 (Signature Quadratic Form Distance)**.** *Given two feature signatures $S^o = \{\langle r_i^o, w_i^o \rangle\}_{i=1}^n$ and $S^p = \{\langle r_i^p, w_i^p \rangle\}_{i=1}^m$ and a similarity function $f_s : \mathbb{F} \times \mathbb{F} \to \mathbb{R}$ over a feature space $\mathbb{F}$, the signature quadratic form distance SQFD_{f_s} between S^o and S^p is defined as:*

$$\mathrm{SQFD}_{f_s}(S^o, S^p) = \sqrt{(w_o \mid -w_p) \cdot A_{f_s} \cdot (w_o \mid -w_p)^T},$$

where $A_{f_s} \in \mathbb{R}^{(n+m)\times(n+m)}$ is the similarity matrix arising from applying the similarity function f_s to the corresponding feature representatives, i.e., $a_{ij} = f_s(r_i, r_j)$. Furthermore, $w_o = (w_1^o, \ldots, w_n^o)$ and $w_p = (w_1^p, \ldots, w_m^p)$ form weight vectors, and $(w_o \mid -w_p) = (w_1^o, \ldots, w_n^o, -w_1^p, \ldots, -w_m^p)$ denotes the concatenation of weight vectors w_o and $-w_p$.

We use the Gaussian similarity function $f_s = e^{-\alpha \cdot d^2(c_i, c_j)}$ in our experiments.

2.2 Deep Neural Network Layer Features

DNN are composed of layers, which in turn are composed of neurons.

Neuron has one or more inputs and sums them to produce output.

Formally, output of a neuron is defined as $y = f(\sum_i w_i x_i + b)$, where:

- x_i are inputs of the k-th neuron
- w_i are weights of the inputs
- b is bias of the inputs
- f is the activation function

Layers of neurons are connected in such a way that outputs of the previous layer are the inputs of the following one.

Layers at the beginning of the network are detecting activation of the lower layer features, like edges and textures, while layers closer to the end of the network are detecting higher level features, like objects and object parts (Zhou et al. [2014]).

Training the neural network is a resource consuming task. They are trained using millions of images and it takes weeks to train them on the GPU (Graphics Processing Unit). As an example, the AlexNet network was trained for 6 days on two high end GPUs (Krizhevsky et al. [2012]). We will not be training our own deep neural networks. Rather, we will use a pretrained model.

2.2.1 Pretrained Neural Network Models

The neural networks selected were a state-of-the-art of their time. Their creators used them to participate, and most of them won, the object classification task of the ILSVRC (ImageNet Large-Scale Visual Recognition Challenge) competition (Russakovsky et al. [2015]).

AlexNet

Named by one of its authors, Alex Krizhevsky et al. [2012], it became famous by winning the ILSVRC 2012. Year 2012 was the first year where a deep convolution network was used. It won the competition with a top 5 test error rate of 15.4%. The next best entry achieved an error of 26.2%, which was a great improvement.

The network is made up of 5 convolution layers, max-pooling layers, dropout layers, and 3 fully connected layers. It has 60 million parameters and 650,000 neurons.

We obtained a replication of the trained model from the official Caffe BVLC (Berkeley Vision and Learning Center) GitHub web page.[1]

[1] `https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet`

CaffeNet

This is a slightly modified version of AlexNet which has the norm and pool layers swapped (pooling is done before normalization).

A trained model can be downloaded from the official Caffe BVLC GitHub web page.[2]

VGG

The VGG team, Simonyan and Zisserman [2014] of the University of Oxford, participated with their network in the ILSVRC 2014 and secured the first and the second places in the localization and classification tasks respectively. The error rate achieved in the classification task was 7.3%.

The team opted for simplicity of the convolution layers and the resulting ability to increase network depth. They strictly used 3x3 filters and pushed the depth to 16 and 19 layers (we are using the 19 layers version). In comparison, AlexNet has 11x11 filters in the first layer. The team's reasoning is that the combination of two 3x3 convolution layers has an effective receptive field of 5x5. This in turn simulates a larger filter while keeping the benefits of smaller filter sizes.

The number of filters doubles after each maxpool layer. The resulting shrinking of the spatial dimensions, but growing depth, reinforces the notion that convolutional neural networks have to have a deep network of layers in order for the hierarchical representation of visual data to work.

The models were publicly released by the VGG team.[3]

GoogLeNet

This was the winner of the ILSVRC 2014 with a top-5 error rate of 6.7%, and one of the first DCNN architectures that strayed from the general approach of simply stacking convolution and pooling layers on top of each other in a sequential structure. The authors showed that a creative structuring of layers can lead to improved performance and computational efficiency.

The Google team, Szegedy et al. [2014], introduced an inception module. The idea behind it is that while designing a traditional convolutional network, at each layer you have to make a choice of whether to have a pooling operation or a convolution operation and what filter size to use. The inception module allows to perform all of these operations in parallel.

Another interesting thing is that GoogLeNet does not use fully connected layers. It simply does an average pooling of the final layers.

The GoogLeNet is built using 9 inception modules in the whole architecture, with over 100 layers in total. It uses 12x fewer parameters than AlexNet, while being significantly more accurate. As the authors put it:

> "The main hallmark of this architecture is the improved utilization of the computing resources inside the network. This was achieved by a carefully crafted design that allows for increasing the depth and width of the network while keeping the computational budget constant."

[2] `https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet`

[3] `http://www.robots.ox.ac.uk/~vgg/research/very_deep/`

A replication was obtained from the official Caffe BVLC GitHub web page.[4]

ResNet

Microsoft Research Asia (MSRA), He et al. [2015], won the ILSVRC 2015 with an incredible error rate of 3.6%. Their 152 layer network is 8x deeper than VGG but still having lower complexity.

The network was constructed using multiple residual blocks. The idea behind a residual block is that an input x goes through a series of Convolution-ReLU-Convolution layers. This will result in some $F(x)$, which is then added to the original input x. The authors hypothesize that:

> "It is easier to optimize the residual mapping than to optimize the original, unreferenced mapping."He et al. [2015]

We tested the 50, 101 an 152 layer architectures. The authors released the models on their GitHub webpage.[5]

Neural network comparison table

Table 2.1: Neural network comparison table

Network name	**Model size**	**Layer depth**	**ILSVRC top-5 error**
AlexNet	232MB	8	15.4%
CaffeNet	232MB	8	-
VGG	548MB	19	7.3%
GoogLeNet	51MB	22	6.7%
ResNet-50	98MB	50	-
ResNet-101	170MB	101	-
ResNet-152	230MB	152	3.6%

[4] `https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet`
[5] `https://github.com/KaimingHe/deep-residual-networks`

3. Evaluation Methodology

In the experiments, two different types of image retrieval tasks are compared based on a specific methodology summarized in the following sections. The benchmark datasets, methods used to fine-tune the models and approaches used for simulations of target browsing, are highlighted.

3.1 Benchmark Datasets

3.1.1 Thematic Web Images Collection (TWIC)

Thematic Web Images Collection (TWIC) Lokoč et al. [2012] is a dataset comprising 11,555 images divided into 200 classes. To create the dataset, keywords from various topics were selected and for each keyword the results found by Google image search engine were manually filtered. Query objects were selected manually - the authors have focused on objects that visually represent the majority in the class (the query object is not an outlier).

We use this annotated dataset to compute the Mean Average Precision (MAP) in our similarity search experiments. Its relatively small size enables us to perform a multitude of MAP evaluations.

3.1.2 ImageNet ILSVRC Subset

ImageNet (Deng et al. [2009]) is an image database organized according to the WordNet hierarchy (currently only the nouns). WordNet (Miller [1995]) is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. The synsets are interlinked by the means of conceptual-semantic and lexical relations.

Each node of the ImageNet hierarchy is depicted by hundreds and thousands of images. According to statistics from 2010[1], it contains around 14 million images, hierarchically organized in approximately 22 thousand synsets (sets of synonyms).

A set of 1000 categories were selected[2] and used in a classification task of the ILSVRC challenge 2012 (Russakovsky et al. [2015]). We have selected a subset of about 335,556 images. We chose this number to approximately match the TRECVid keyframes dataset size. From each of the 1000 classes a random subset of 336 images was selected. We included all images of a class with less than 336 images.

Some important properties of the dataset are:

- The image size and the aspect ratio are very variable. According to our observations, the image size varies between approximately 1280x1024 and 140x100. The images can be found both in landscape and portrait shapes.
- The ILSVRC dataset contains multiple classes of dogs of different breed. The reason is that the organizers of the ILSVRC replaced other classes

[1] `http://image-net.org/about-stats`

[2] `http://image-net.org/challenges/LSVRC/2012/browse-synsets`

during the ILSVRC 2012 challenge due to introduction of the fine-grained classification task that year.[3]

ImageNet images are available for non-commercial research or educational use only. We have used the dataset for target browsing simulations.

3.1.3 TRECVid Keyframes

The TRECVid keyframes dataset corresponds to 335,944 extracted and officially provided[4] keyframes from the TRECVid IACC.3 video dataset (Awad et al. [2016]). The IACC.3 dataset (Internet Archive videos with Creative Commons license) contains approximately 4600 videos (144 GB, 600 h) in MPEG-4/H.264 format with duration ranging from 6.5 min to 9.5 min and the mean duration of almost 7.8 min. The resolution of most videos (and thus extracted keyframes) is 320x240 pixels.

The boundaries of non-overlapping video shots were automatically detected in the videos, and one keyframe was extracted from each video shot.

The keyframe collection can be downloaded from the following TRECVid webpage.[4]

We selected TRECVid keyframes as another dataset for simulations of target browsing in images. Furthermore, the TRECVid keyframes dataset shows also the performance of browsing in a collection of video files. The ImageNet images have objects of interest positioned mostly in the center of the image (according to our observation) and the images are sharp and focused. In contrast, the TRECVid keyframe images often capture objects in motion, and the objects of interest are not necessarily in the center of the image.

One important property of this dataset is that it contains scenes that do not change very much (for example in a video of a reporter in a studio). These clusters of very similar images make searching in the dataset very challenging.

3.2 Similarity Model Fine-tuning

To obtain the best results from the models compared, we adjusted their parameters according to the TWIC benchmark dataset. For the DCNN descriptors, we compared activation features extracted from each layer of the deep convolutional neural network architectures that were considered. We have selected the best descriptors (for a given layer), considering various distance measures. Hence, the fine-tuning efforts were focused more on the signature-based models.

3.2.1 Fine-tuning of the Feature Signature Extraction

The effectiveness of the signature-based models depends on the alpha parameter of the SQFD (Beecks et al. [2010]) and several parameters of the feature signature extractor (Kruliš et al. [2016]).

The extractor algorithm is divided into two subroutines:

[3] `http://image-net.org/challenges/LSVRC/2012/browse-synsets`

[4] `http://www-nlpir.nist.gov/projects/tv2016/pastdata/iacc.3.master.shot.reference/`

- **Sampler** sampling low level features from the input image at a $[x_i, y_i]$ position. The sampled features are a CIELab color component $[L_i, a_i, b_i]$ and a texture component $[c_i, e_i]$ composed of contrast and entropy.
- **Clusterizer** joining these low level features into clusters. The resulting cluster centroids, together with their weights, create a feature signature.

Settings of both subroutines can be altered by multiple parameters. These parameters were thoroughly examined in Kruliš et al. [2016]. We used the recommended values of parameters as a compromise between precision and performance:

Sampling parameters

- 2000 sampling points
- 4 bits per pixel of the gray scale image used to sample texture
- Radius of the lookup window used to sample texture $r_{ce} = 3$
- Gaussian point distribution – the points were generated using the OpenCV random normal distribution number generator with standard deviation of 0.2. The points with distance from point [0, 0] greater than 0.5 were ignored and a new point was generated until all 2000 sampling points were created instead. The standard deviation value was not mentioned in the paper, so we selected the value 0.2 experimentally, so that the distribution most closely resembles the depicted image of the Gaussian distribution in Kruliš et al. [2016].

K-means parameters

- 400 initial seeds
- Cluster joining distance $d_{min} = 0.2$
- The maximal number of k-means iterations $I_{max} = 10$
- The minimal cluster size $C_{min} = 2$ (clusters smaller than $C_{min} \cdot I$ are pruned).
- L_2 distance metric

We experimentally found the optimal values for the additional parameters that were not mentioned or that are dependent on the dataset. After extracting feature signatures using a configuration of parameters, we tested this configuration using MAP (Mean Average Precision) evaluation of the annotated TWIC image dataset.

Weight of sampled features

The weight of each dimension of sampled features can affect the effectiveness, so we have considered additional parameters to fine-tune the resulting feature signatures.

In a sampled feature $S_i = [x_i, y_i, L_i, a_i, b_i, c_i, e_i]$ we define

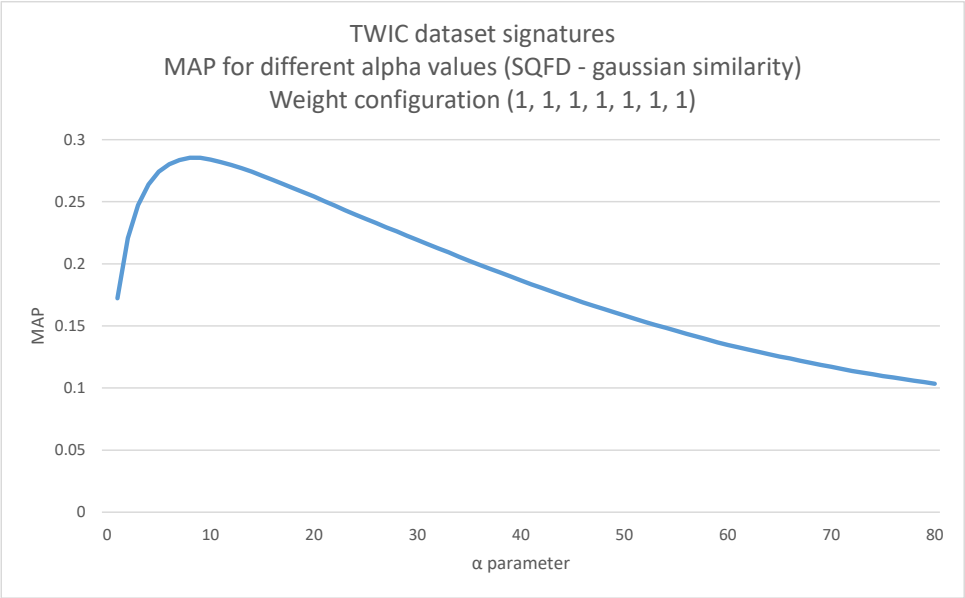
- $[x_i, y_i]$ as the position component

- $[L_i, a_i, b_i]$ as the color component
- $[c_i, e_i]$ as the texture component

To generate multiple configurations of feature dimension weights we introduce a set of component weights p, c and t. We restrict their values by equation $p + c + t = 1$ to be able to set values of two component weights and evaluate the third. This enables us to plot the configuration in a 3-dimensional graph. The sampled feature weight configuration is then generated as $[p, p, c, c, c, t, t]$

3.2.2 MAP Behavior Depending on the α Parameter

Figure 3.1: MAP behavior depending on the α parameter



The similarity function used in the SQFD measure introduces an α parameter that is dependent on the dataset (Beecks et al. [2010]). We tested MAP on the TWIC dataset for different values of α. We observed that the MAP function of α appears to be a continuous function on $(0, \infty)$ with one maximum at α_{max}. In order to find the α_{max} we implemented an algorithm that samples different α values and converges to the α_{max} value.

1. The algorithm starts with an ordered set of 3 α values:

   $(a, c, e) \in (\mathbb{R}^+)^3; a < c < e$

   The algorithm expects that $\alpha_{max} \in (a, c)$

2. The additional middle points of the intervals $[a, c]$ and $[c, e]$ are then computed:

   $b = \frac{a+c}{2}$

   $d = \frac{c+e}{2}$

3. MAP is evaluated for each of the 5 sampling α values a, b, c, d, e

4. Based on the sampled values, a new ordered set of 3 α values from the previous 5, is selected: the value generating the highest MAP and two of its neighbors.

5. The algorithm is repeated until a desired delta of the highest MAP values between iterations is achieved.

However, in our tests we found out that the α parameter is also dependent on the sampled feature dimension weights presented in the previous subsection. Therefore, we had to sample the best α value for each centroid weight configuration.

3.2.3 Selection of the Best Performing DCNN Layer

We extracted DCNN descriptors from all deep neural networks and their layers. For each layer descriptors we then evaluated MAP on TWIC dataset using Euclidean, Manhattan, Hamming and Cosine distance.

In layers where individual neurons produced more than one value we selected maximum of these values (the maximal activation).

3.3 Target Browsing

For the target browsing, we have again compared two similarity models. We have also selected a browsing approach enabling coarse to fine browsing.

3.3.1 Models Compared

Multimedia retrieval systems are usually developed for ordinary users without deep expert knowledge of system inner workings and without skills to control complex interfaces. From this perspective, similarity browsing constitutes an appropriate approach, where users only select a set of images and proceed to the next display in an intuitive interface. The usability gap is thus transferred to the proper usage of the similarity model employed. Therefore, the similarity model has to be intuitive enough for users, who have to clearly understand the effects of their actions, and thus control all the interactions. One option is semantic search, where users select as mediators objects from the same or highly related semantic category. Although models for semantic search still suffer from the semantic gap, recent breakthroughs in deep convolutional neural networks have significantly improved their effectiveness. Another intuitive option is color-based search, where users focus on the global color distribution in the memorized image, and from the actual display, select as mediators those images that resemble the memorized image, regardless the depicted object. The following representatives of both models are investigated in the experiments:

- **Image thumbnails** of size $w \times h$ represent approximations of the color distributions in the original images. Considering three color dimensions in chromatic images, each image o is then represented as a vector $x^o_{CD} \in R^{w \cdot h \cdot 3}$.

The color information is often transformed to a perceptually uniform color space (we use CIE Lab). Using position-color feature signatures obtained from an image resize operation enables comparison of two thumbnails by very efficient Euclidean distance (unlike the expensive SQFD).

- **Deep convolutional neural network activation features** $x^o_{NN} \in R^{4096}$ extracted from the last fully connected layer of the pretrained VGG network architecture (Simonyan and Zisserman [2014]). It has been shown that the features carry semantic information of a modeled image o and can be used for effective retrieval of objects from the same category using the Cosine or Euclidean distance.

Display size and organization

For similarity browsing, the number of images presented in each display D_i matters. The higher the number of images, the more options to select the most promising mediators. Assuming a standard notebook with full HD, one to two hundreds of images can be displayed as still readable thumbnails. Indeed, with higher number of images presented, users need more time to orient themselves in the display, and can overlook promising mediators or might even miss the target image. Hence, proper organization of images is necessary to benefit from larger displays. Various 2D/3D organization approaches have been presented and several studies have been evaluated (Schoeffmann and Ahlström [2012], Schoeffmann et al. [2014], Schaefer [2009]). The promising approach of color and/or semantic sorting in a 2D grid (Barthel et al. [2017]) has been successfully introduced (and soon adopted by other teams) at the Video browser showdown competition. The simulations in the experimental section assume that displays with up to 100 images do not slow down the process of the identification of proper mediators.

Page zero problem

The selection of the starting display D_0 significantly influences the effectiveness of the browsing and the overall number of necessary interactions. As failed query initialization is assumed, displays based on selected representatives are considered. A popular option to select a representative display D_0 is a clustering of the dataset in a preprocessing phase. As a result, a set of representative images is selected as a fixed D_0. The clustering is required for each similarity model considered separately. In this paper we consider a two-phase clustering. The first efficient phase employs hierarchical k-means clustering (Nister and Stewenius [2006]) to select a set of representative candidates. In the second phase, a hierarchical agglomerative clustering (Everitt et al. [2011]) is employed to select the desired number of objects for the display.

Display model

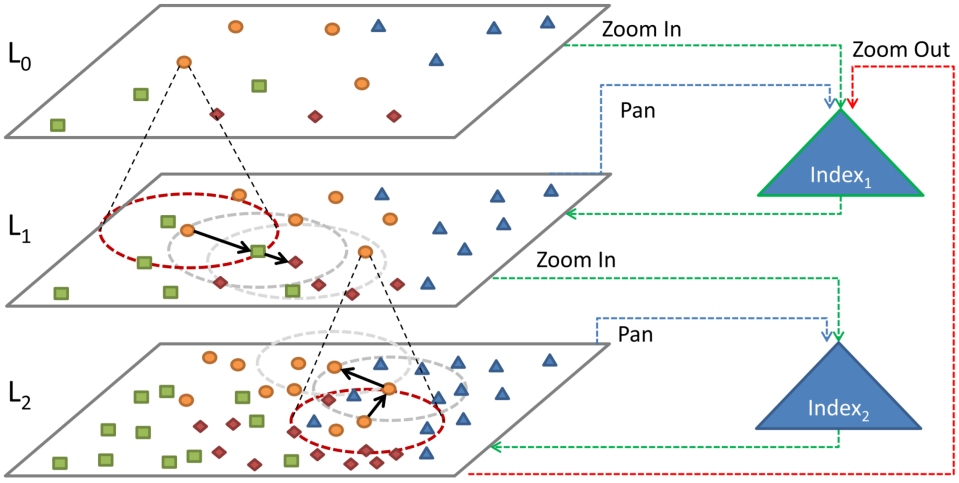
In order to select objects for the next display, multimedia browsing systems require a suitable organization of the collection. The structure should provide options for coarse to fine browsing and efficient interaction. Heesch [2008] distinguishes three classes of structures – static hierarchies, static networks and dynamic structures. In the experiments, the selection of new displays is ensured

by the combination of kNN queries (corresponding to a static network) and the following two approaches:

- **MLES structure** (Mosko et al. [2015]) organizes the dataset into a set of layers $L_0 \subset L_1 \subset \cdots \subset L_l$, where L_0 is the zero page and the last layer $L_l = S$. The employed method for the construction of the layers is described in the following subsection. The objects in the upper levels (closer to zero page) are supposed to serve as landmarks for coarse navigation, while the lower level layers are accessed once users track promising mediators (similar as navigation in maps). Formally, given a selected mediator m from a layer L_i, browsing operations $Pan(m, k, i) = kNN(m, L_i)$ and $ZoomIn(m, k, i) = kNN(m, L_{i+1})$ can be used to select objects for the next display (note that ZoomIn is not defined for the last layer L_l).

Figure 3.2: MLES with $Index_1$ for layer L_1 and $Index_2$ for layer L_2

Mosko et al. [2015]



- **Drop factor** can be introduced to prevent users from being stuck in a cluster of pairwise highly similar yet non-relevant objects in layer L_i. For an object o that already appeared in the browsing history for $d > 0$ times, its probability to be dropped from the new display is set to $1 - 1/d$. Note that the drop factor represents a heuristic to gradually investigate a larger neighborhood of the selected mediators.

MLES settings

For simulations of coarse to fine browsing, a three layer exploration structure (MLES) is employed. The layer $L_2 = S$, the layer L_1 (22500 objects) is selected using fast hierarchical k-means from L_2, while the layer L_0 is selected using a more expensive agglomerative clustering from L_1.

More specifically, the layer L_1 is selected using hierarchical k-means clustering as follows. First, 150 seeds are randomly selected and X iterations of k-means are evaluated. For each of the resulting cluster in the first level, again 150 seeds are randomly selected and k-means is performed. For each centroids c_i detected

in the second level of the clustering (i.e., for all 150^2 centroids), $1NN(c_i, S)$ is inserted to layer L_1.

A second method of hierarchical clustering that we tested selects a different amount of seeds for each cluster C_i in the second level. Based on the expected number of generated clusters (150^2), we set the seed count to $|C_i|/(|S|/150^2)$. We call this method an **adaptive clustering**. The adaptive clustering places more seeds in the bigger clusters. This gave us a greater browsing resolution in those clusters.

Figure 3.3: Visualization of non-adaptive and adaptive clustering

(a) Non-adaptive divisive clustering (b) Adaptive divisive clustering

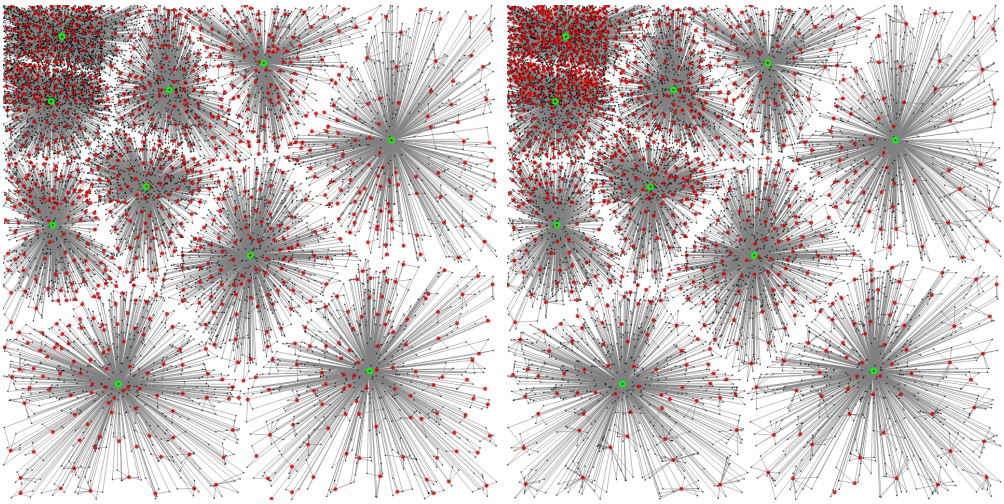


Figure 3.3 was generated using the two methods of hierarchical clustering on a dataset of 10,000 initial black points. Green points are centroids of clusters generated in the first level (10 seeds) while red points are centroids generated in the second (100 seeds per cluster). We can observe a higher density of red points in the bigger clusters of adaptive clustering.

The layer L_0 is selected from L_1 using agglomerative hierarchical clustering (Ward's method) with the given number of objects $|D_0|$ for the first display. Again, for the $|D_0|$ resulting centroids their closest objects from layer L_1 are used as representatives.

4. Experimental Results

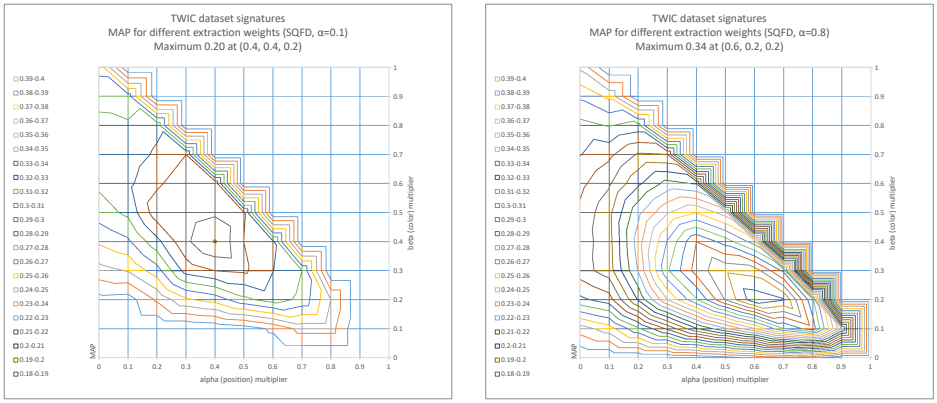
4.1 Query-by-example Benchmarking

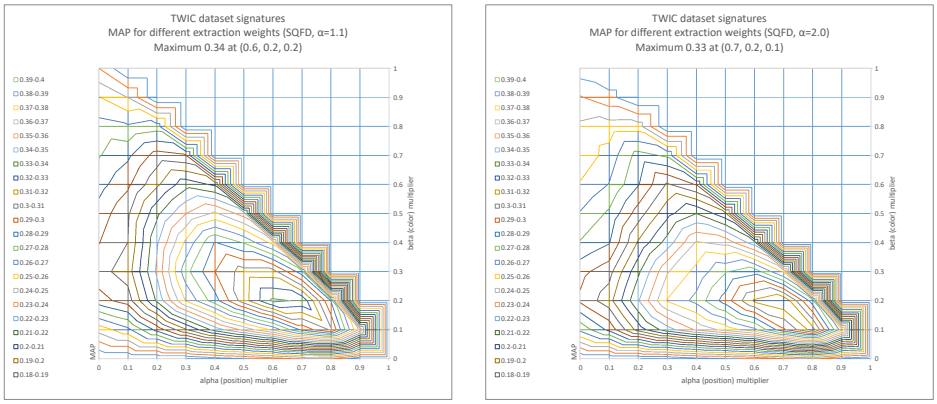
4.1.1 Signature-based Models

We extracted feature signatures of multiple configurations of sampled feature dimension weights. We evaluated MAP of each configuration on the TWIC dataset using SQFD with different values of parameter α. We can observe in Figure 4.1 that the optimal feature weight configuration is changing depending on the α parameter.

In the next step we sampled the best value of α parameter for each configuration individually. From the left graph in Figure 4.2 we deduce that the ideal feature weight component multipliers are 0.6× position, 0.2× color and 0.2× texture. This translates to feature weight vector (6, 6, 2, 2, 2, 2, 2) with $\alpha = 1.1$. Using this parameter configuration we were able to achieve MAP=0.338.

Figure 4.1: MAP of different weight configurations using SQFD with fixed alpha





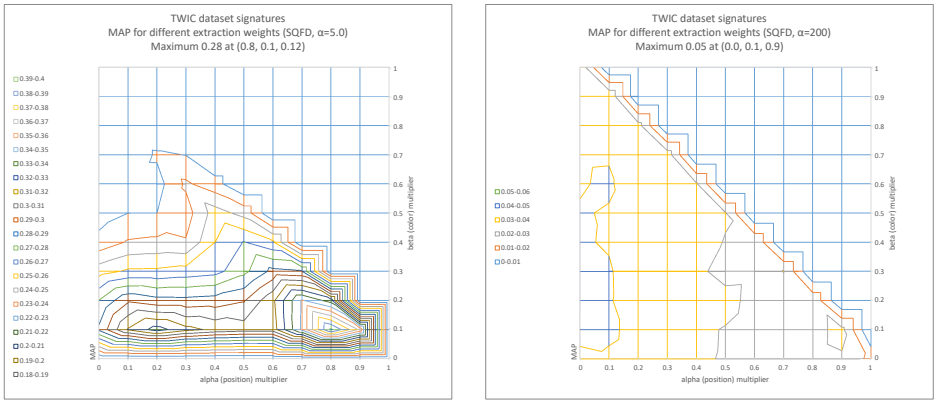
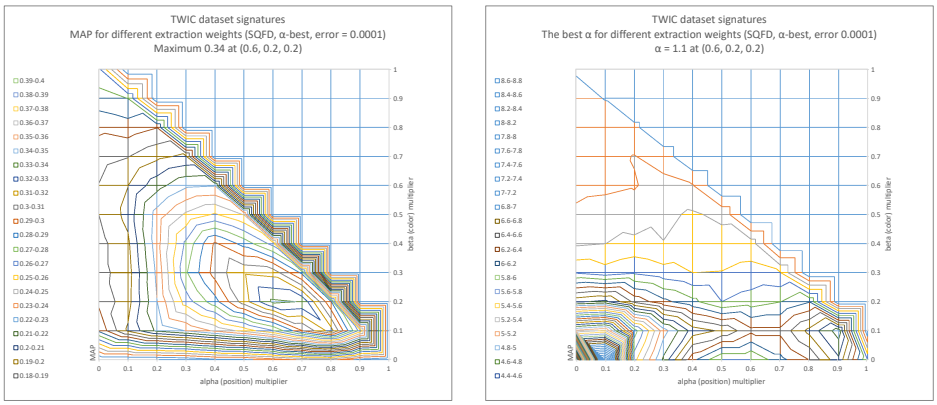


Figure 4.2: MAP of different weight configurations using SQFD with the ideal α



4.1.2 DCNN-based Models

For each pretrained DCNN model we extracted DCNN descriptors of the TWIC dataset from the output blob of each layer (some layers share the output blobs because the layer operations are performed in-place). We evaluated MAP of the descriptors using multiple distance metrics and created a graph for each network. The layers in each graph are sorted according to their depth in the network (the layers to the right are deeper).

We see in Figures 4.3 and 4.4 that the highest values of MAP are similar across all networks. The highest MAP ranges from 0.63 (VGG) to 0.72 (AlexNet).

Distance measures

Cosine distance shows the best results in all networks. This suggests that in the DCNN descriptor vector space, angle of two vectors is more important than distance.

We see a very little difference between L_1 and L_2 distances. Surprisingly, Hamming distance competes with or even surpasses the other metrics.

Layer depth

Looking at the results of each network, it is clear that DCNN descriptors from the deepest layers are the ones best performing in the similarity search task. We see a drop in MAP at the last layers, probably due to their reduced dimensionality.

We also observed in the Figure 4.4 that increased depth of the ResNet networks did not increase MAP of the deeper layers.

Figure 4.3: MAP of different network layers

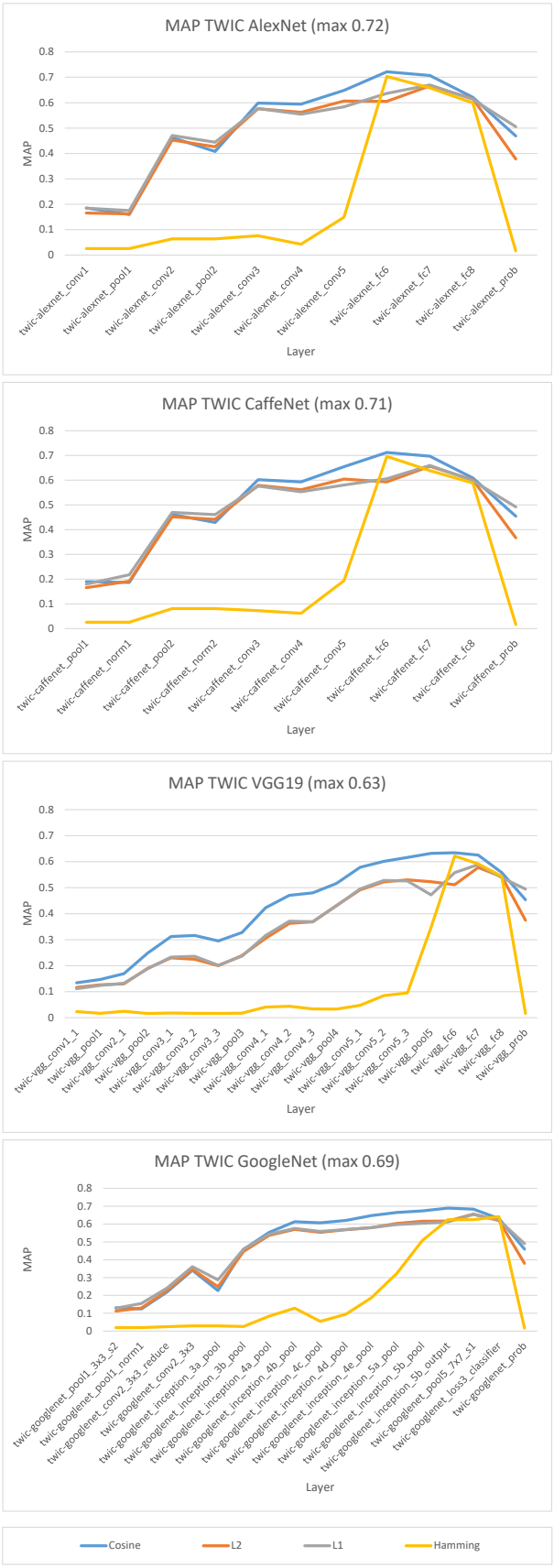
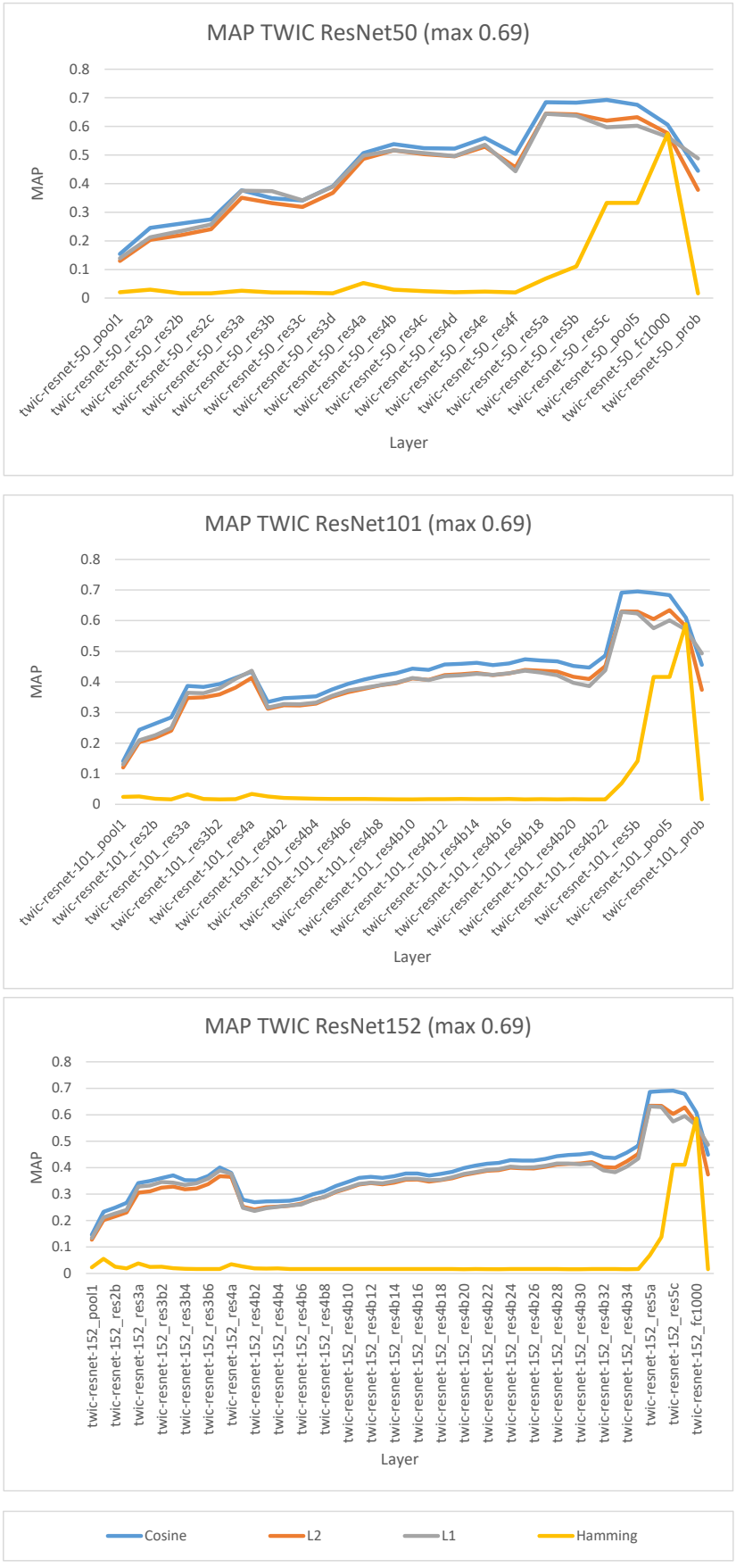


Figure 4.4: MAP of different network layers



4.1.3 Summary

By fine-tuning of the signature-based model we achieved the highest value of MAP = 0.338. This result correlates with MAP = 0.32 found by Lokoč et al. [2012] using the same benchmark dataset TWIC.

We observed a causality in signature-based model between sampled feature dimension weights and the α parameter of the SQFD measure. The model had to be fine-tuned with a consideration of this.

We found the ideal feature signature extractor weights $[6, 6, 2, 2, 2, 2, 2]$. Further testing on different datasets would be required to confirm that the weight configuration is consistent across all datasets and that they are not specific to our benchmark dataset TWIC.

DCNN-based models are more effective than signature-based models. All deep neural network models achieved overwhelmingly better results that the signature-based. The highest MAP value that we achieved was 0.72 using DCNN descriptors from the fc6 layer of the AlexNet network. This is is contrast with the precision of the classification task, where AlexNet was the least performing from all listed deep neural networks.

We selected the best performing configurations from both models. Then we compared the average precision for each query used in MAP evaluation. We counted how many times one model had a higher average precision than the other for the same query. We found that DCNN descriptors are better performing in 87% of queries for the TWIC benchmark dataset.

4.2 Target Browsing Simulations

Based on the methodology described in Section 3.3, we have evaluated four sets of experiments (four big Figures 4.5 4.6 4.7 4.8). More specifically, the sets are evaluated for two datasets (ILSVRC and TRECVid) and for each dataset two types of clustering are investigated for the layer L_1 (with and without adaptive clustering).

For each set of the experiments, we have considered nine graphs – three display sizes times three different numbers of simulated PAN operations in layer L_1 (denoted as ZOOM step 1-3). Note that ZOOM step 1 considers no additional simulated PAN interaction in layer L_1 and directly ZOOMs to L_2 in the second simulated interaction, while ZOOM step 3 considers two PAN operations in L_1 during the next two simulated interactions after the first ZOOM from L_0.

In each graph (of nine) in each big figure, four types of users are evaluated for both models denoted as CU_1 (ideal user), CU_{ALL} (random user), $CU_{0.125}$ and $CU_{0.375}$ (both coherent to some extent). The compared models are denoted as DNN (DCNN descriptors) and 8×8 (Image thumbnails with resolution 8×8). The X axis presents the number of visited displays (i.e., the number of interactions), while the Y axis presents the cumulative distribution $F(t) = P(T < t)$ (T is the number of displays to find a target image). The 1000 target images were selected randomly such that the images did not appear in the zero page. All the compared models/settings shared the same set of target images.

For all the figures, we may observe that

- As expected, the artificial users require a less number of interactions for a higher number of displayed items. Also the number of successful sessions during 20 interactions is higher for bigger displays.
- We observe just a small difference between the two types of used clustering methods.
- The TRECVid dataset (a video dataset) is more difficult to browse than the ImageNet dataset.
- Higher ZOOM step helps predominantly in browsing sessions with greater display size.
- Both models perform almost equally in the TRECVid dataset. The DCNN-model has approximately twice that good results in the ImageNet dataset than the signature-based model. This could be caused by the fact that most of the DCNN models were trained on the ImageNet dataset, hence the classes form compact clusters using the DCNN descriptors.

Figure 4.5: ILSVRC 2012 without adaptive clustering

Figure 4.6: TRECVid keyframes without adaptive clustering

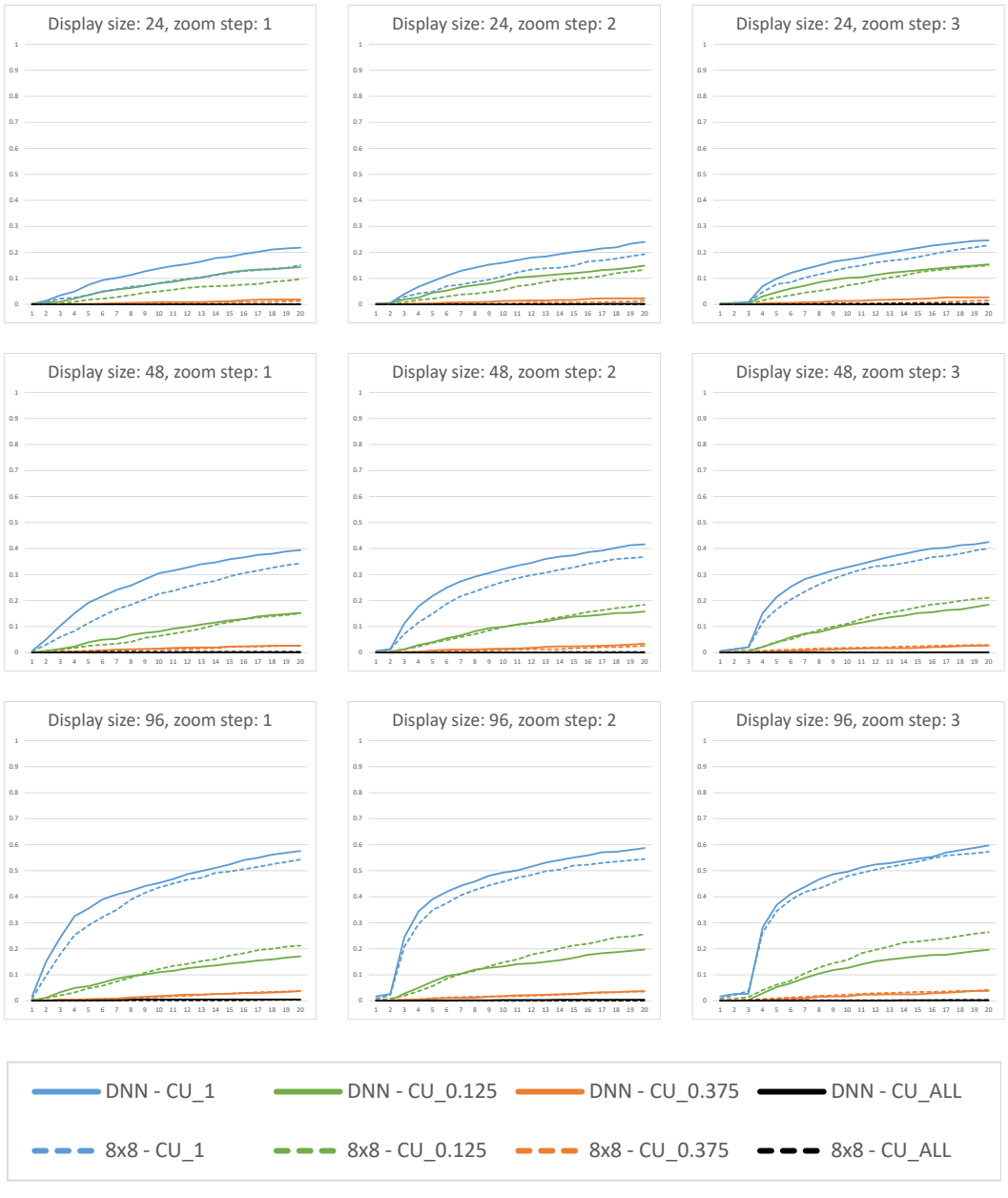
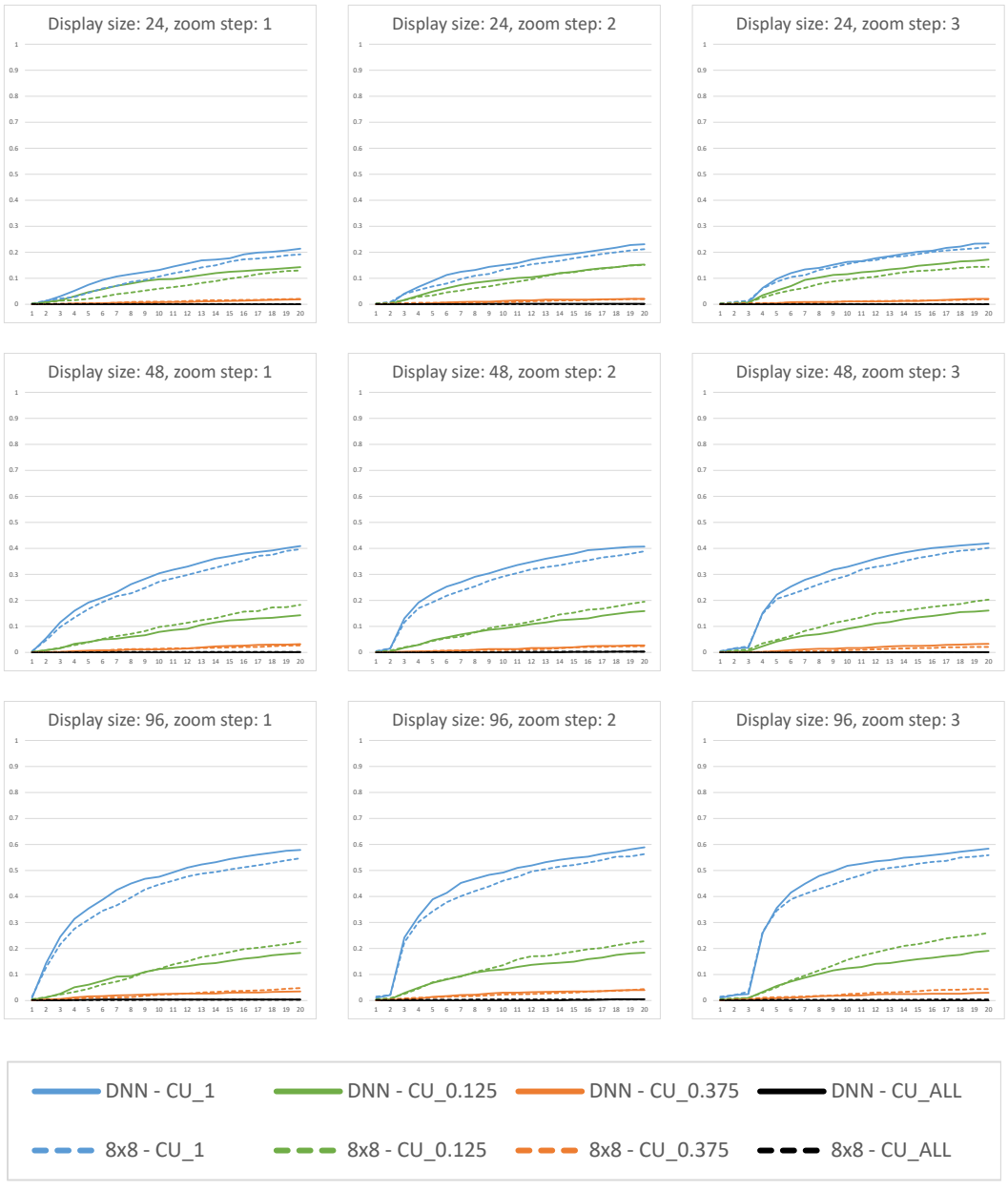


Figure 4.7: ILSVRC 2012 with adaptive clustering

Figure 4.8: TRECVid keyframes with adaptive clustering



4.2.1 Summary

The experiments reveal that the DCNN based features are highly competitive also in the target browsing scenario. However, the graphs show aggregated results for a high number of queries. In Tables 4.1 and 4.2, we present more detailed statistic regarding particular search sessions. The question is - are there target objects (i.e., sessions) where color feature signatures perform better? The Table presents the percentage of search sessions, where DCNN descriptors or feature signatures required a lesser number of simulated interactions to find the target image. The percentage of equal performances is presented in the middle column. From the results, we may observe that also traditional similarity models based on colors can be still competitive in the target browsing scenario. Especially for the more challenging TRECVid dataset.

Table 4.1: The proportion of faster browsing sessions of CU_1 user on ImageNet

Zoom step 1	**DCNN**	**Neither**	**8x8**
Display size 24	0.396	0.487	0.117
Display size 48	0.543	0.291	0.166
Display size 96	0.621	0.185	0.194
Zoom step 2	**DCNN**	**Neither**	**8x8**
Display size 24	0.407	0.473	0.12
Display size 48	0.559	0.277	0.164
Display size 96	0.597	0.203	0.2
Zoom step 3	**DCNN**	**Neither**	**8x8**
Display size 24	0.412	0.465	0.123
Display size 48	0.553	0.286	0.161
Display size 96	0.607	0.213	0.18

Table 4.2: The proportion of faster browsing sessions of CU_1 user on TRECVid

Zoom step 1	**DCNN**	**Neither**	**8x8**
Display size 24	0.195	0.692	0.113
Display size 48	0.314	0.455	0.231
Display size 96	0.389	0.279	0.332
Zoom step 2	**DCNN**	**Neither**	**8x8**
Display size 24	0.205	0.652	0.143
Display size 48	0.329	0.424	0.247
Display size 96	0.379	0.303	0.318
Zoom step 3	**DCNN**	**Neither**	**8x8**
Display size 24	0.209	0.623	0.168
Display size 48	0.319	0.417	0.264
Display size 96	0.37	0.306	0.324

5. Implementation Description

5.1 Descriptor Extraction

5.1.1 Feature Signature Extractor

We obtained a C++ source code of the extractor from Kruliš et al. [2016]. We analyzed the code and reimplemented it into the OpenCV (Open Computer Vision) library. The reimplementation process involved changing custom structures and templates to match the OpenCV interface. This included an analysis of the OpenCV interface from the existing source code, since there was little documentation of the OpenCV internals.

Algorithm analysis and implementation decisions

The extractor algorithm is divided into two subroutines:

- **Sampler** sampling low level features from the input image at a $[x_i, y_i]$ position. The sampled features are a CIELab color component $[L_i, a_i, b_i]$ and a texture component $[c_i, e_i]$ composed of contrast and entropy.
- **Clusterizer** joining these low level features into clusters of higher level features. The resulting centroids, together with their weights, create a feature signature.

The settings of both subroutines can be altered by multiple parameters. These parameters are described in the algorithm documentation.

OpenCV code contribution process

The OpenCV library is composed of three open source repositories:

- **opencv** containing core modules
- **opencv_contrib** for new experimental code
- **opencv_extra** containing extra data

As we were adding new experimental code, we were required to contribute into the opencv_contrib repository. We decided to contribute our code into the **xfeatures2d** module since our algorithm is extracting 2D features.

When designing an extension to the OpenCV library, we had to adhere to its coding style guide. This includes naming conventions, the file structure, the code layout and many others.[1]

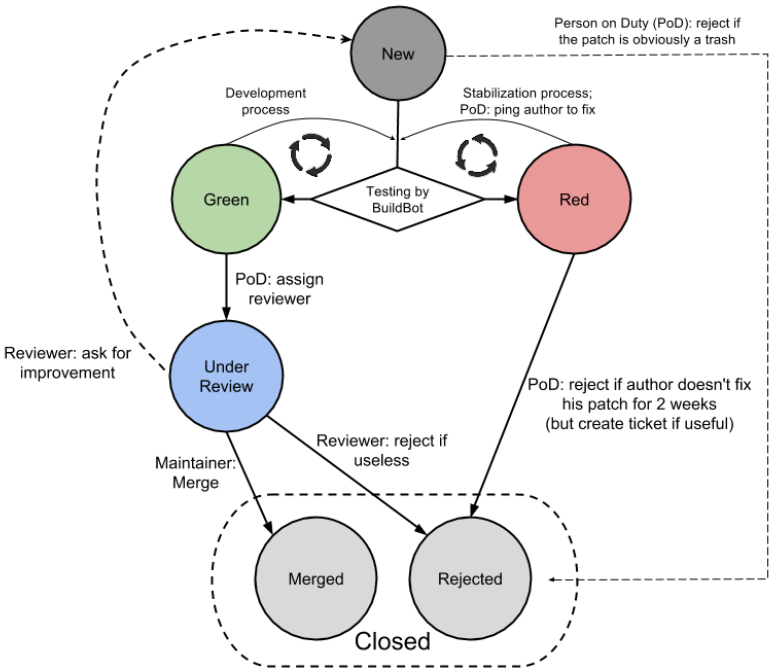
The actual contribution is managed through the GitHub webpage[2]. We had to fork the original opencv_contrib repository and perform the source code changes required in our copy. To start the process of contribution, as depicted in Figure

[1] `https://github.com/opencv/opencv/wiki/Coding_Style_Guide`
[2] `https://github.com/opencv/opencv_contrib`

5.1, we had to create a pull request. A pull request is a request to merge our copy of the repository with the original one.

The first phase of the contribution process is automatic. A build bot downloads our copy and runs build tests. If the tests are successful, the code is later reviewed by the repository maintainers. Although the build bot phase is quick, finishing the tests in a few minutes, the code review phase can last several weeks. Our review period lasted about three weeks.

Figure 5.1: OpenCV contribution process (source: `http://opencv.org`)



.NET wrapper EmguCV

Since the OpenCV library is programmed in C++ language and we wanted to use it in our .NET projects, we used the EmguCV .NET wrapper of OpenCV. When building EmguCV from source code, it downloads the latest version of OpenCV together with our feature signature extractor in it.

5.1.2 DCNN Descriptor Extractor

To implement the DCNN descriptor extractor we used Caffe framework Jia et al. [2014]. It was a challenge to build the Caffe source code on a Windows platform since it was not supported at first. We had to resort to unofficial builds. Enthusiasts, like the user initialneil [3], managed to rewrite parts of the internal code and compile Windows versions of the third party libraries that Caffe used. Later, the Caffe repository maintainers included a version of the source code able to be compiled on a Windows platform. They provided a preconfigured Visual Studio

[3] `https://github.com/initialneil`

solution that was easy to set up and extend. It also included a static library build of Caffe. This Visual Studio build was later deprecated and removed. A new build system that was introduced was based on CMAKE. Since we had little experience with extending the CMAKE version, we reverted to the latest Caffe version containing a working Visual Studio build.

Caffe framework description and analysis

A DCNN network is represented by the class `caffe:Net<T>()`. The class contains a set of network layers, represented by the class `caffe:Layer<T>()`. Layers communicate between each other by using `caffe:Blob<T>()`, which is simply a wrapper over the N-dimensional array. Blob also provides the synchronization capability between CPU and GPU. They conceal the underlying mixed CPU/GPU operation by synchronizing from the CPU host to the GPU device as needed.

The training of the Caffe DCNN model works by loading a preprocessed input image into the input blob of the first layer and then doing a forward pass through the network to produce an output. An error from the expected value is computed at the last layer and the error is then back-propagated using the backward pass. Back-propagation trains the network by altering the weights of the layer neurons. Since we are not interested in training a Caffe network, only in using a pretrained one, we ignore the backward pass capability.

Extracting descriptors from a network layer

We based our approach on an example C++ project provided by Caffe. We start the extraction by preprocessing the input image first. The image is resized and normalized from pixel values range [0-255] to [0-1]. A mean image is then subtracted from it. A mean image is the mean image of the dataset the model was trained on. After loading the image data to the input blob a forward pass is performed. The DCNN descriptor is then extracted from the selected layer output blob.

DCNN extractor C++/CLI wrapper

To be able to use the unmanaged C++ extractor in a managed .NET environment, we created a C++/CLI wrapper. The wrapper simply creates an unmanaged extractor instance and transfers data between the managed and unmanaged memories. Since it is often desired to extract DCNN descriptors from multiple features, we provided a way to forward pass the input image once, and extract the descriptors stored in layer output blobs multiple times.

5.2 MAP Evaluation

The MAP evaluator algorithm computes MAP on a dataset using an annotation file. We implemented different versions of the MAP evaluator for feature signatures and DCNN descriptors. Since MAP evaluation of feature signatures using the SQFD function was very resource demanding, we also implemented the GPU

version that uses nVidia CUDA capable devices to speed up the evaluation. For this CUDA GPU implementation we used AleaGPU library[4].

5.3 Target Browsing

5.3.1 Generating a MLES Structure

We used the following clustering implementations in creation of the MLES.

- **Simple clustering** - a simple version of basic kNN clustering. It cycles assignment and update operations for a number of iterations.
- **Divisive clustering** - it uses a top-down approach where a set of objects is split into smaller clusters. The process is repeated with the smaller clusters. We use this clustering to generate an initial set of clusters in a large dataset that is very resource demanding.
- **Agglomerative clustering** using Ward's method. A bottom up approach that joins two closest clusters in each iteration. The distance is measured using Ward's method as $d_w(x_i, x_j) = L_2(x_i, x_j)^2 \cdot (w_i \cdot w_j)/(w_i + w_j)$. We use this method to generate a zero page display for the target browser.

5.3.2 Browsing Simulator

The browsing simulator is a core component that manages creation and behavior of browsing sessions. A browsing session simulates one target search. It stores various status and logging variables. Based on the status variables either the ZOOM or PAN operation is performed.

The selection of display items is managed by the MLES component. It manages kNN search in the database, as well as search in the MLES structure. Many queries are repeated multiple times. To speed up the simulation we added a caching dictionary (a hash-table) that is saving a sorted list of 1000 nearest objects. The dictionary is saved to a file at the program shutdown.

[4] `http://www.aleagpu.com`

Conclusions

In this thesis, we analyzed two similarity models. We simulated a real world scenarios where these models were compared. As expected, we found out that the Deep Neural Network models outperform the signature based models in many tasks. However, in some scenarios, like for example browsing in the TRECVid dataset, both models perform similarly.

We also analyzed internal components of the OpenCV and Caffe libraries. We explored the highly organized contribution process when plenty of contributors have to be organized to create and maintain a stable library. We used this knowledge to extend these frameworks and we were listed as contributors on their official webpage. We are looking forward to participating in more open source projects such as these.

In future, we would like to focus our research on deep neural networks and their applications for multimedia exploration and target browsing.

Bibliography

George Awad, Jonathan Fiscus, Martial Michel, David Joy, Wessel Kraaij, Alan F. Smeaton, Georges Quénot, Maria Eskevich, Robin Aly, Gareth J. F. Jones, Roeland Ordelman, Benoit Huet, and Martha Larson. Trecvid 2016: Evaluating video search, video event detection, localization, and hyperlinking. In Proceedings of TRECVID 2016. NIST, USA, 2016.

Kai Uwe Barthel, Nico Hezel, and Klaus Jung. Visually browsing millions of images using image graphs. In Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, ICMR '17, pages 475–479, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4701-3. doi: 10.1145/3078971.3079016. URL http://doi.acm.org/10.1145/3078971.3079016.

Christian Beecks. Distance-based similarity models for content-based multimedia retrieval. In Dissertation, Fakultät für Mathematik, Informatik und Naturwissenschaften, RWTH Aachen University., 2013.

Christian Beecks, Merih Seran Uysal, and Thomas Seidl. Signature quadratic form distance. In Proceedings of the ACM International Conference on Image and Video Retrieval, CIVR '10, pages 438–445, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0117-6. doi: 10.1145/1816041.1816105. URL http://doi.acm.org/10.1145/1816041.1816105.

Steven M. Beitzel, Eric C. Jensen, and Ophir Frieder. MAP, pages 1691–1692. Springer US, Boston, MA, 2009. ISBN 978-0-387-39940-9. doi: 10.1007/978-0-387-39940-9_492. URL https://doi.org/10.1007/978-0-387-39940-9_492.

G. Bradski. The OpenCV Library. Dr. Dobb's Journal of Software Tools, 2000.

Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang. Image retrieval: Ideas, influences, and trends of the new age. ACM Computing Surveys (Csur), 40(2): 5, 2008.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09, 2009.

Brian Everitt, Sabine Landau, Morven Leese, and Daniel Stahl. Cluster analysis. Wiley, 5th edition, 2011. ISBN 978-0-470-74991-3.

Marin Ferecatu and Donald Geman. A statistical framework for image category search from a mental picture. IEEE Trans. Pattern Anal. Mach. Intell., 31(6): 1087–1101, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Daniel Heesch. A survey of browsing models for content based image retrieval. Multimedia Tools Appl., 40(2):261–284, 2008.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Martin Kruliš, Jakub Lokoč, and Tomáš Skopal. Efficient extraction of clustering-based feature signatures using gpu architectures. Multimedia Tools and Applications, 75(13):8071–8103, 2016. ISSN 1573-7721. doi: 10.1007/s11042-015-2726-y. URL http://dx.doi.org/10.1007/s11042-015-2726-y.

Jakub Lokoč, David Novák, Michal Batko, and Tomáš Skopal. Visual Image Search: Feature Signatures or/and Global Descriptors, pages 177–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-32153-5. doi: 10.1007/978-3-642-32153-5_13. URL http://dx.doi.org/10.1007/978-3-642-32153-5_13.

George A. Miller. Wordnet: A lexical database for english. Commun. ACM, 38(11):39–41, November 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL http://doi.acm.org/10.1145/219717.219748.

Juraj Mosko, Jakub Lokoc, Tomas Grosup, Premysl Cech, Tomas Skopal, and Jan Lansky. MLES: multilayer exploration structure for multimedia exploration. In New Trends in Databases and Information Systems - ADBIS 2015 Short Papers and Workshops, BigDap, DCSA, GID, MEBIS, OAIS, SW4CH, WISARD, Poitiers, France, September 8-11, 2015. Proceedings, pages 135–144, 2015.

David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2, CVPR '06, pages 2161–2168, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2597-0. doi: 10.1109/CVPR.2006.264. URL http://dx.doi.org/10.1109/CVPR.2006.264.

Yossi Rubner and Carlo Tomasi. Perceptual Metrics for Image Database Navigation. Kluwer Academic Publishers, Norwell, MA, USA, 2001. ISBN 0792372190.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Gerald Schaefer. A next generation browsing environment for large image repositories. Multimedia Tools and Applications, 47(1):105–120, 2009. ISSN 1573-7721. doi: 10.1007/s11042-009-0409-2. URL http://dx.doi.org/10.1007/s11042-009-0409-2.

Klaus Schoeffmann and David Ahlström. An evaluation of color sorting for image browsing. Int. J. Multimed. Data Eng. Manag., 3(1):49–62, January 2012. ISSN 1947-8534. doi: 10.4018/jmdem.2012010104. URL http://dx.doi.org/10.4018/jmdem.2012010104.

Klaus Schoeffmann, David Ahlström, and Marco Andrea Hudelist. 3-d interfaces to improve the performance of visual known-item search. Multimedia, IEEE Transactions on, 16(7):10, dec 2014. ISSN 1520-9210. doi: 10.1109/TMM.2014.2333666.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. CoRR, abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.

Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In Proceedings of the 22Nd ACM International Conference on Multimedia, MM '14, pages 157–166, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3063-3. doi: 10.1145/2647868.2654948. URL http://doi.acm.org/10.1145/2647868.2654948.

Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. Similarity Search: The Metric Space Approach (Advances in Database Systems). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. ISBN 0387291466.

Ethan Zhang and Yi Zhang. Average Precision, pages 192–193. Springer US, Boston, MA, 2009. ISBN 978-0-387-39940-9. doi: 10.1007/978-0-387-39940-9_482. URL https://doi.org/10.1007/978-0-387-39940-9_482.

Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. CoRR, abs/1412.6856, 2014. URL http://arxiv.org/abs/1412.6856.

List of Figures

List of Tables

Attachments

1.) DVD containing copy of this thesis, source codes and an example data.