Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Bachelor's Thesis

Photo Culling - Selecting a Representative Set of Photographs

Lukáš Bartůněk
Cybernetics and Robotics

Supervisor: prof. Dr. Ing. Jan Kybic
January 2024

Acknowledgements

I would like to extend my sincere thanks to my supervisor prof. Dr. Ing. Jan Kybic as his guidance has helped me countless times. Thanks should also go to every school employee I had the pleasure of working with and learning from.

I am also grateful to my classmates and friends for the quality time spent discussing various problems that we encountered over the years. I would be remiss in not mentioning my family, especially my parents and my girlfriend for their never-ending support throughout my studies.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 1. January 2024

.....

Abstract

The objective of this thesis was to offer a solution to the time-consuming management of photo albums in the form of a summarization tool. We laid down few common methods and approaches to image data extraction, as well as reported on summarization tools and software that have been already developed. We then focused on proposing ways of combining extracted image information using conditional statements, a logical approximation model and a shallow neural network. A graphical interface was later developed showcasing all the proposed solutions. Moreover, we added learning capabilities for two of those solutions which are capable of closer approximation of user's preferences. We conducted a survey comparing selections made by our proposed solutions and a selection made by human. The results of the survey show that the proposed methods create selections able to compete with the human-made selection.

Keywords: photo summarization, automatic photo album management, photo quality, duplicity elimination

Supervisor: prof. Dr. Ing. Jan Kybic

Abstrakt

Cílem této téze bylo nabídnout řešení pro časově náročnou správu fotoalba ve formě nástroje vytvářející shrnutí. Stanovili jsme několik metod a přístupů k extrakci informací z fotografii a podali jsme zprávu o nástrojích a softwaru, které se problémem již zabývají. Poté jsme se zaměřili na navrhnutí způsobů kombinace extrahovaných informací za pomoci použití podmínkových výrazů, logické aproximace a mělké neaurální sítě. Nástroj pro uživatelskou interakci v grafickém prostředí byl vyvinut s ukázkou navrhnutích řešení. Dále jsem pro dvě z těchto řešení přidali možnost učení, které je schopno přiblížit výsledky blíže k uživatelským preferencím. Provedli jsme průzkum porovnávající výběry vytvořené našimi navrhnutími řešeními a výběrem vytvořeným člověkem. Výsledky veřejného průzkumu ukazují, že naše metody dokáží vytvořit výběry konkurující lidským výběrům.

Klíčová slova: shrnutí fotografii, automatická správa fotoalba, kvalita fotografii, eliminace duplikátů

Překlad názvu: Výběr reprezentativního souboru fotografií

Contents

Figures

Tables

Chapter 1

Introduction

With the wide spread of mobile phones and cameras at the beginning of this century, a common human experience is to take pictures and recordings of everything that is going on around us. There is probably nothing more common than to go on a holiday or visit an interesting place and to take a vast amount of photos to remember your experience or share it with family and friends. This can often lead to hundreds of photos that a person subsequently has to go through and delete all the unwanted ones.

We are focusing on this modern problem of managing your personal image collection that is both time draining and often very inefficient. We aim to create a tool that can help us filter a long sequence of photos and generate smaller subset that does not contain duplicates or near duplicates, blurred or otherwise damaged photos and prioritize the selection of the ones with high aesthetic quality.

Our tool should take a sequence of photos then analyze it to extract all the useful information. The information is then used to select a subset that represents the whole sequence. This can be done by different methods with different settings based on user's preferences.

1.1 Problem definition

To achieve our goal of creating a selection generating software, it is necessary to break down the problem into parts and define them.

Our input is a sequence of images with no additional information.

$$X \text{ - sequence of images}, \quad x_i \text{ - image with index i}$$

Firstly, we need to define what we mean by image quality, there are two aspects to focus on. The technical quality of an image, which is mostly objective.

The other aspect is the image aesthetics, which is mostly subjective. The qualities are represented by a number between 0 and 100, and it signifies the rating that would be given by a human.

$$Q_i^t \text{ - technical quality of i-th image,} \quad Q_i^t \in [0; 100]$$

$$Q_i^a \text{ - aesthetic quality of i-th image,} \quad Q_i^a \in [0; 100]$$

Secondly, we aim to filter redundant copies, and so we need to define a metric of similarity between images. We focus on two of the metrics. First is how structurally similar the images are. If two images have the same building and the building angle is identical, the images are structurally similar. We calculate this by comparing pixel to pixel. This metric is represented by percentage value between 0 and 100; the lower the number the lower the similarity.

$$S_{i,j}^s \text{ - structural similarity between i-th and j-th image}$$

Second metric is how similar is the content of the images, which means what themes are present in the image. If two images have different buildings that look nothing alike but are still recognizably buildings, the images have high content similarity.

We define the content as normalized vector of values representing how well each label fits the image.

$$\boldsymbol{C_i} \text{ - content vector of i-th image,}$$

$$\boldsymbol{C_i} = [c_1, ..., c_l, ..., c_m], \quad ||\boldsymbol{C_i}|| = 1$$

$$c_l \text{ - value of l-th label,} \quad m \text{ - number of labels}$$

We calculate content similarity by using a vector distance of vectors C_i. The range of values is scaled, so the numbers are between 0 and 100; the lower the number the lower the similarity.

$$S_{i,j}^c \text{ - content similarity between i-th and j-th image}$$

Our approach to generating a selection is to use the information extracted from the image and decide if that image is to be selected (value of 1) or culled (value of 0).

$$z_i \text{ - binary value of i-th image representing whether image is selected}$$

$$\zeta = \{z_i\}, \quad z_i \in \{0, 1\}, \quad i \in \{1, 2, ..., n\}$$

The output of our tool is a subset of image sequence X. We are going to call this final subset a selection and use the symbol O.

$$O = \{x_i | z_i = 1\}, \quad i \in \{1, 2, ..., n\}$$

1.2 Dataset used to train and test solutions

For the purpose of testing and training we will be using a dataset that was created from holiday images. This dataset contains 5 502 images. Examples of images from the dataset can be seen on Figure 1.1.

Figure 1.1: Example of images from the dataset

We created the dataset by taking photos from a long holiday and manually selecting nice images to create labels.

Since our dataset is unbalanced and much more images are not selected it will be useful to use class weights to avoid biased predictions. To calculate class weights we need to know how many images there are in our dataset and how many of them are labeled to be selected.

$l = 5502$ - number of images in the training dataset

$l_s = 750$ - number of images labeled to be selected

With this, we can calculate the class weight for selected images (w_s) and not selected images (w_n).

$$\begin{aligned} w_s &= \frac{l}{n_c \cdot l_s} = 3.668, \\ w_n &= \frac{l}{n_c \cdot (l - l_s)} = 1.158 \end{aligned} \tag{1.1}$$

where n_c is the number of classes; 2 in our case.

The training on this dataset is done by splitting it into training and validation dataset. The testing is done by cross-validation, where we test the performance on the whole dataset. Validation dataset is composed of a smaller sequence from our dataset and contains 438 images (8% of the dataset). The rest of the dataset is used for training.

Chapter 2

Existing software and solutions for image selection

As the first step in our development we researched what other software and solutions there are available. There are two groups of tools to focus on: a software that is either open source or commercially available and solutions described in research papers.

For our software testing we had two requirements. The software needed to be available for Window or Linux and the software had to have free automatic photo culling feature or a free trial for that feature.

For commercially available software we had particularly limited information about the principle of the culling, and so we only describe the functions based on our testing. As tools and solutions presented in scientific research papers are focused on the inner workings, we are able to describe those solutions on a deeper level.

2.1 Approaches described in research papers

2.1.1 Time and color histogram clustering

This approach is described in the paper: Automatic Summarization for Personal Digital Photos [1]

It partitions a sequence into smaller clusters of photos that are taken at similar time and those clusters are then partitioned based upon the content of the photos. The time when the photo was taken is extracted from the metadata. It is then necessary for the user to set a threshold to determine how close said photos need to be taken to be clustered together. The content partition is created by comparing the pictures in a time-based photo cluster based on their color histograms.

The key photo of each cluster is then decided and all key photos of the whole sequence form the final selection. The paper acknowledges the relation between image similarity detection and key photo selection, but opts not to use image similarity. It proposes the option of various selection criterions for key photo selection. An example from the paper is face criterion; which gives higher importance to photos including face. Another example presented is time criterion; which gives higher importance to images that were taken in very short succession as this might mean that the photographer wanted to make sure to capture the object well.

The paper provides a simple graph (shown on Figure 2.1) showing the process of the selection creation, with the option of browsing the selection or sharing it through a communication network.

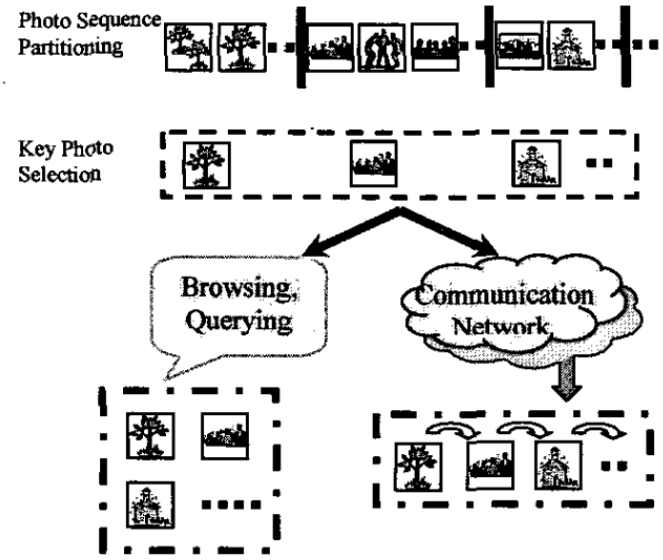


Figure 2.1: Summarization of framework; source: [1]

2.1.2 Query-based selection creation with extensive clustering

This approach is described in the paper: Personal Photo Album Summarization [2]

First step described in this paper is extraction of features from all the images. It extracts time, location and optical context data (eq. exposure, flash etc) from the metadata. It then extracts color and edge histograms. Lastly it creates GIST features [3] for each image which are used as representation of that image. The process is enhanced by face recognition. The paper does not go into detail how the color and edge histograms are extracted or how the faces are detected.

The paper then describes three variants of selection creation based on time interval of the sequence; a yearbook, week events and day events. The first two variants use macro level partition. Macro level partitioning is based on time, location and directory structure. It should allow for more inclusive representation by separating the images into macro clusters for large sequences.

Clusters and exemplars are created using exemplar selection algorithm which is described in a different research paper [4]. The idea of the algorithm is to create clusters of images that are structurally similar (pixel relation) and taken at similar time.

The selection is created based upon many factors including uniqueness (how many pictures were taken at that place in that time), presence of faces and others based on ranking logic. The ranking logic is derived from a query-retrieval procedure and can change the importance of each ranking factor.

The paper also includes a graphic (shown on Figure 2.2) of the whole process. It also shows the fact that the paper's tool is split into an offline and online stage. Although this only seems to be for convenience's sake, the paper does not mention any reason for this.

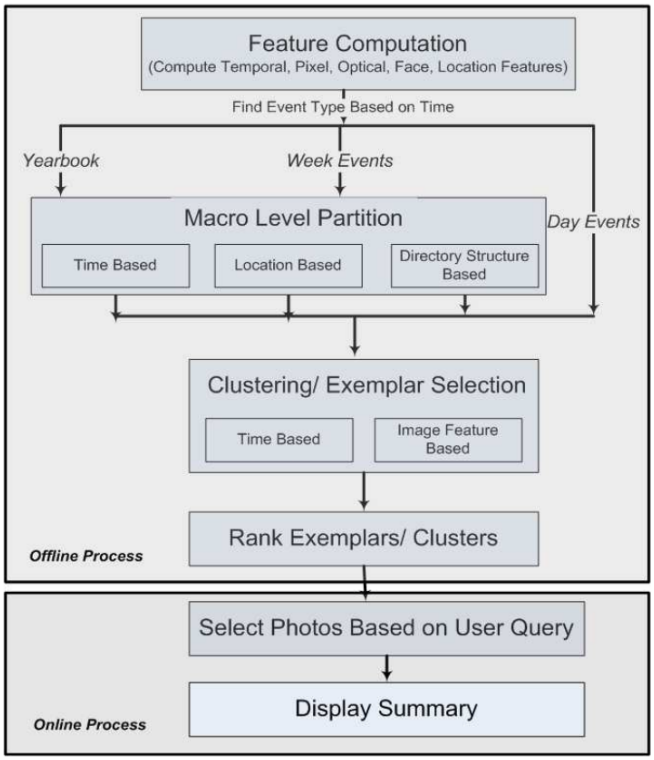


Figure 2.2: Summarization of software pipeline; source: [2]

2.1.3 Selection creation with preset setting options

This approach is based on the paper: Photo Summary: Automated Selection of Representative Photos from a Digital Collection [5].

Like the previous papers, this paper extracts the image metadata that it uses to cluster photos into subsets aiming to isolate events. Their tool however has more user settings that can be set. User can set a number of images of the final selection, people weight, importance weight and appeal weight. The paper does not go into details what those weights influence. It does also offer a three preset for those settings; appeal preset, people preset and balanced preset.

The paper states that the algorithm decides how important the event is and chooses the appropriate number of photos from it, it however does not state how the importance of event is determined. It also filters duplicates and near-duplicates, but it fails to mention how. The photos are rated based on their technical quality, which is determined by detecting exposure errors and blurriness. The paper does not mention what algorithm is used for this. Their tool also detects faces and based on the user setting gives higher importance to those photos. The aesthetics of the photo are assessed based upon simplicity, composition and colorfulness. The paper again does not mention how this is determined. The final selection is sorted chronologically.

We were unable to access the final program as it is protected by company license.

2.1.4 Cluster and annotate

This approach is based on: CLAN Photo Presenter: Multi-modal Summarization Tool for Image Collections [6].

The name of this tool gives insight in what it does, as CLAN stands for cluster and annotate. As in all previous papers the first part of the tool creates clusters. This paper however assumes that the images are in form of unordered set without any metadata. Clusters are created based on image similarity. Image similarity is assessed based on salient visual features extracted from the images during preprocessing (algorithm for preprocessing is not specified). The clustering can be influenced by the user as there are three clustering algorithms (bisecting K-means, K-means and distinct kNN) and other parameters (the paper mentions number of clusters as example).

The second part of the tool creates keyword annotations for each cluster. The annotations are completed by annotating representatives (how the representatives are selected is not stated, possibly randomly) and then their descriptions are aggregated. The number of representatives for each cluster can be changed by user. The selection is presented in a web interface.

The paper provides a scheme of the workings (shown on Figure 2.3).

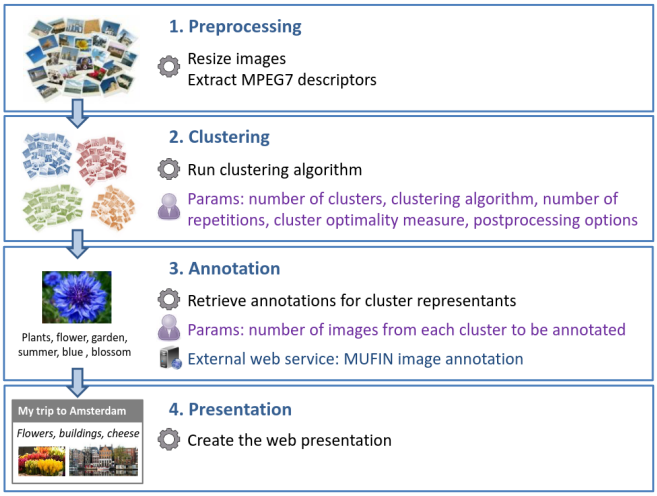


Figure 2.3: Scheme of algorithm process; source: [6]

This tool is open-source. It is however outdated, and we were unable to run it on our system.

2.2 Tested software

2.2.1 Aftershoot

Aftershoot is commercially available software that offers automatic culling as part of their paid features. It also offers 14-days trial, and thus we were able to test this feature. The feature itself costs $15/month or $25/month in version that includes their editing features as well. [1]

From our trial usage we can conclude that it offers image similarity assessment and similar image grouping. Our opinion is that the similarity algorithm is focusing on structural similarity. User can influence how strict the culling of duplicates is.

The software also has image quality assessment, which seems to be mainly focused on sharpness and closed eye detection. This implies usage of the Blind IQA algorithm and possibly face recognition.

The star rating of each image is given automatically based on conditions such as eye focus or presence of better duplicates. It also offers option to rate the images manually and filter the images based on the rating. The automatic

[1] Software can be found on: https://aftershoot.com/

rating does not take into account the relative quality of the images. The rating seemingly gives 5 stars to every image that does not feature closed eyes or is very blurry.

It does not appear to use metadata for culling process and does not write into image metadata.

Aftershoot states that the culling process is trainable but in our few experiments we have not been able to prove that it does (this does not however mean that training is not present).

Overall Aftershoot offers a lot of features that improve the experience of working with a collection of photos. However, we think that the culling process lacks the option of more advanced settings; e.g. even stricter selection or the option to influence the automatic rating. There should also be more options for aesthetic quality of images.

2.2.2 FilterPixel

FilterPixel is another commercially available software that we looked into. It offers a free version, which does not include automatic culling and a paid version, which does include automatic culling. The paid version offers a 14-days trial, so we were able to test this software. Normally the software is priced at $19.99/month. [2]

The software includes a similar assessment and similarity grouping. The similar images are seemingly determined by structural similarity. The strictness of the similarity grouping and culling can be set by the user, it does not seem to make much of a difference. Overall the similarity does not seem to be very strict and only near identical images are affected.

Quality assessment is in the form of eye focus and focus quality (sharpness). Focus quality and eye focus are presented in ambiguous percentages and an overall rating of the image is based upon the mean of these values.

FilterPixel does not include any sort of rating. It does not work with metadata and does not feature any sort of training capabilities.

We were deeply dissatisfied with FilterPixel's culling. The software only takes two metrics into account with no way to influence the ratio of their influence on the outcome of the process. Even on the strictest setting that it offers it filters about 10% of the photos. The only positive of the software is the speed of the process.

[2] Software can be found on: https://filterpixel.com/

2.2.3 PhotoRefine

PhotoRefine is part of Zenfolio subscription and offers automatic photo culling. Zenfolio offers 14-days trial, and henceforth we were able to try their automatic photo culling feature. The Zenfolio subscription that includes this software costs $40/month. [3]

It features a similarity assessment with similarity grouping. The software also offers grouping based on time at which the image was taken (information extracted from metadata). We could not figure out on which basis the similarity assessment is completed, however it appears to be grouping structurally similar pictures.

Quality assessment includes sharpness rating, closed eye detection, face focus and a face happiness rating. This implies facial detection. It does not state how the facial happiness is determined, however we presume that it uses CNN. A user can influence the importance of each of these four factors for overall image rating.

It automatically adds rating in the form of 5-star scale to images and offers an option for the user to edit the rating manually.

As we mentioned previously, it does read metadata for additional information for improved culling. It also writes the image rating into metadata.

It does not offer any training features.

The software has a variety of features and is very easy to use with a lot of preset settings. The user can influence the rating factors and the software features grouping based on time the photo was taken (this requires the images to have metadata information about this). The only inconvenience we found is the lack of easy import of photos to be culled; the software does not allow the user to select a folder with photos and all photos need to be imported separately.

2.2.4 Software feature comparison

For easier software comparison, we created a table of available features and additional information. This acts as sort of quick summary of the software and does not represent all the features of the individual software.

We included the presence of similarity assessment, technical quality assessment (e.q. blurriness detection) and aesthetic quality assessment (e.q. eye or face focus detection).

Furthermore, we looked if the program has grouping of similar images, the

[3] Software can be found on: https://zenfolio.com/features/photorefine-photo-culling/

options to manually rate images, read and write metadata and the trainability of user interaction.

As additional information we looked at availability, the platform it can be used on and the time to complete the photo culling (all software was tested on same computer).

Results of comparisons can be seen in Table 2.1 and Table 2.2

	Presence of features			
	Similarity	Technical Q.	Aesthetic Q.	Grouping
Aftershoot	YES	YES	Closed eye det.	YES
FilterPixel	YES	YES	Closed eye det.	YES
PhotoRefine	YES	YES	YES	YES

	Presence of features			
	Man. Rate	Meta. Read	Meta. Write	Trainable
Aftershoot	YES	NO	NO	Stated
FilterPixel	YES	NO	NO	NO
PhotoRefine	YES	YES	YES	NO

Table 2.1: Feature comparison

	Additional information			
	Time	Availability	Platform	Price [$/month]
Aftershoot	0:46:25	14-days trial	Windows/Mac	15
FilterPixel	0:11:09	14-days trial	Windows/Mac	19.99
PhotoRefine	1:21:51	14-days trial	Windows/Mac	40

Table 2.2: Comparison of additional software information

Chapter 3

Existing methods for extraction of useful image information

One of our sub-objectives was to familiarize ourselves with existing tools and approaches in domain of image data processing, suggest a suitable architecture for the photo selection tool and subsequently test out individual algorithms.

3.1 Image quality

For assessing image quality we have to consider two aspects. Technical and aesthetic quality of an image.

3.1.1 Aesthetic quality

The first method for aesthetic quality that we studied is based on a deep learning approach described in 'NIMA: Neural IMage Assessment by Hossein Talebi and Peyman Milanfar' [7]. This approach uses Convolutional Neural Networks (further as CNN) trained on rescaled images from AVA dataset. This dataset is created with images of amateur photographers and as such they are focused on the aesthetic quality of the images. The aesthetic quality assessment is trained on ratings of the public and therefore it is as close to an objective beauty rating as we can currently get. In our testing we used a NIMA proposed approach trained on VGG16 [8] model.

The output of the CNN is the distribution of ratings that is simulating a distribution of ratings that people might give. From this, we are then able to get a mean rating and the statistical deviation of the rating. We are not particularly interested in any statistical deviation and will only take the mean rating as our quality score.

Since the output of this is in the range [0;10] we scale it by multiplying it by 10 to get our desired Q_i^a range [0;100].

The second explored method is based on the paper 'Exploring CLIP for Assessing the Look and Feel of Images' [9]. This approach uses the CLIP (Contrastive Language-Image Pre-training) model for evaluating the visual aesthetics of images. CLIP is neural network originally designed for joint understanding of text and images. This paper investigates its use for capturing subjective qualities of images.

3.1.2 Technical quality

Technical quality is less subjective, and thus there are multiple algorithms able to assess the quality of an image. There are a few approaches and metrics of Image Quality Assessment (further as IQA) that we looked into. We want a method that only uses the image itself as input of quality assessment. This is called Blind or Referenceless IQA. As the name implies, the methods do not have any reference of how a clear picture with no distortions should look like.

The first method that we studied is 'Blind/Referenceless Image Spatial Quality Evaluator' (BRISQUE)[10]. Firstly, BRISQUE extracts statistical features from an image. These features capture information about the spatial structure and distribution of pixel intensity. These features are then normalized and put together to create feature vectors. Featuring vectors with training labels are then used to train a Support Vector Machine (SVM)[1] to generate image quality score. For our purposes we are going to be using already trained BRISQUE.

The second method tested is described in NIMA paper[7]. This approach trains CNN on TID2013[11] dataset identically to how aesthetic quality is trained on dataset AVA (mentioned in previous section). However, we have used CNN trained on a newer dataset called KonIQ-10k [12] to test the proposed approach.

3.1.3 Testing our selected methods for image quality

For image quality testing we have used IQA toolbox specialized in Image Quality Assessment methods [13]. This toolbox implements all tested methods accordingly with the papers these methods are proposed in. It uses official test splits of datasets for testing. To compare approaches, the toolbox uses commonly used metrics: PLCC (Pearson's Linear Correlation Coefficient), SROCC (Spearman's Rank-order Correlation Coefficient) and KRCC (Kendall

[1]To read more about SVM `www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/`

rank correlation coefficient). For more information about these metrics see Appendix B.

Results for aesthetic image quality tested on AVA dataset can be seen in Table 3.1. These results are based on a comparison between the labels of images in AVA dataset and the guesses done by the tested methods.

Results for technical image quality tested on TID2013 dataset can be seen in Table 3.2. These results are based on a comparison between the labels of images in TID2013 dataset and the guesses done by the tested methods. All the tests were timed on the same machine.

	PLCC	SROCC	KRCC	speed [images/s]
NIMA(AVA)	0.6624	0.6570	0.4719	2.65
CLIP-IQA	0.3576	0.3383	0.2301	0.51

Table 3.1: Test results of selected methods for assessing of aesthetic IQA, tested on AVA dataset

From the results in Table 3.1, we can see that the method CLIP-IQA performed worse in all aspects. Henceforth we will be using approach described in paper NIMA.

	PLCC	SROCC	KRCC	speed [images/s]
BRISQUE	0.4317	0.3672	0.2574	18.40
NIMA (KonIQ-10k)	0.5783	0.4724	0.3277	2.04

Table 3.2: Test results of selected methods for assessing of technical IQA, tested on TID2013

3.2 Image content

Image content is essentially the classification of the images. For the extraction of image content we need an image classifier that will label each image with the image contents.

We have no prior information about the content of the images, which eliminates the use of algorithms like K-nearest neighbors. Without a prepared database it is not possible for us to label images with the use of the Bag of Visual Words technique either.

The best and most popular method is using CNN. For our purpose we used pre-trained EfficientNetV2B1[14] with weights trained on ImageNet[15].

3.3 Image similarity

3.3.1 Structural similarity

There are many methods focusing on structural similarity between two images. Some of them are focused on the structure of the entire image and some are focused more closely on similarity of its smaller parts.

Methods focusing on the entire image are for example Mean Square Error (MSE) and its derivative Peak Signal to Noise Ratio (PSNR). MSE computes mean of square difference between each pixel of two images (see Equation 3.1) and PSNR is essentially inverse logarithmic version of this (see Equation 3.2). The high value of MSE signifies low similarity while high value of PSNR signifies high similarity.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(K_i - L_i)^2, \tag{3.1}$$

where n is a number of pixels and K_i and L_i are pixel values of each image.

$$\mathrm{PSNR} = 10 \cdot \log_{10}\frac{p^2}{\mathrm{MSE}}, \tag{3.2}$$

where p is peak value of pixel (eq. 255 for 8-bit images).

Another example of a method is Structural Similarity Index Measure (SSIM), which takes into account the structural information and does not focus on pixel-wise difference. It uses combination of components luminance (L), contrast (C) and structure (S). This makes the method less sensitive to constant distortions such as color change.

$$\begin{aligned}
\mathrm{L}(x,y) &= \frac{2\cdot\mu_x\cdot\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1},\\
\mathrm{C}(x,y) &= \frac{2\cdot\sigma_x\cdot\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2},\\
\mathrm{S}(x,y) &= \frac{\sigma_{xy} + c_3}{\sigma_x\cdot\sigma_y + c},\\
\mathrm{SSIM}(x,y) &= \mathrm{L}(x,y)\cdot\mathrm{C}(x,y)\cdot\mathrm{S}(x,y),
\end{aligned} \tag{3.3}$$

where μ_x and μ_y are local means, σ_x and σ_y are the local standard deviations, σ_{xy} is covariance and c_1, c_2 and c_3 are constants to avoid division by zero.[2]

[2]To read into detail about SSIM see [16].

Deep Image Structure and Texture Similarity (DISTS)[17] is example of methods that use CNN to asses similarity between images. It focuses on evaluating image similarity by considering structural information (edges and shapes) and textural information (patterns and text) of images.

The last method that we describe is Scale Invariant Feature Transform (SIFT)[18], it works on multiple scales to identify features at different levels detail. It then uses Gaussian filter to detect key points and eliminates low quality ones (low contrast or low definition of edges). In next step it computes the dominant orientation to achieve invariance in image rotation.

It generates descriptors for each key point to store information about the key point. The descriptors are also normalized to ensure invariance to changes in illumination and contrast. And the last step is to match corresponding descriptors for a pair of images. The matching is done by using nearest-neighbor matching approach, where each descriptor is matched to its nearest neighbor in other image based on euclidean distance. Additional measures are applied to filter out ambiguous matches. The output of this method is the percentage of matching descriptors.

3.3.2 Content similarity

We measure content similarity by inverting the distance between two content vectors. The smaller the distance between the vectors the high similarity. There are multiple different ways of calculating the distance between two vectors. For this purpose we can consider two as the best contenders; euclidean distance and cosine distance (cosine similarity). Our content vectors are normalized henceforth the difference between these distances is minimal, and the two are linearly linked. We have arbitrary chosen to use cosine distance.

3.3.3 Testing our selected methods for structural image similarity

For the testing of structural similarity we have chosen to conduct a simple test, where we use methods on set of images and observe how each method calculates the similarity. Testing images can be seen in Figure 3.1. Each method is used to calculate similarity of Toucan 1 and each other image.

For structural similarity we ideally want the method to recognize similarity with Toucan 2 and Toucan 3 while assessing that similarity with Banana and Eagle is low.

(a) : Toucan 1 **(b) :** Toucan 2 **(c) :** Toucan 3

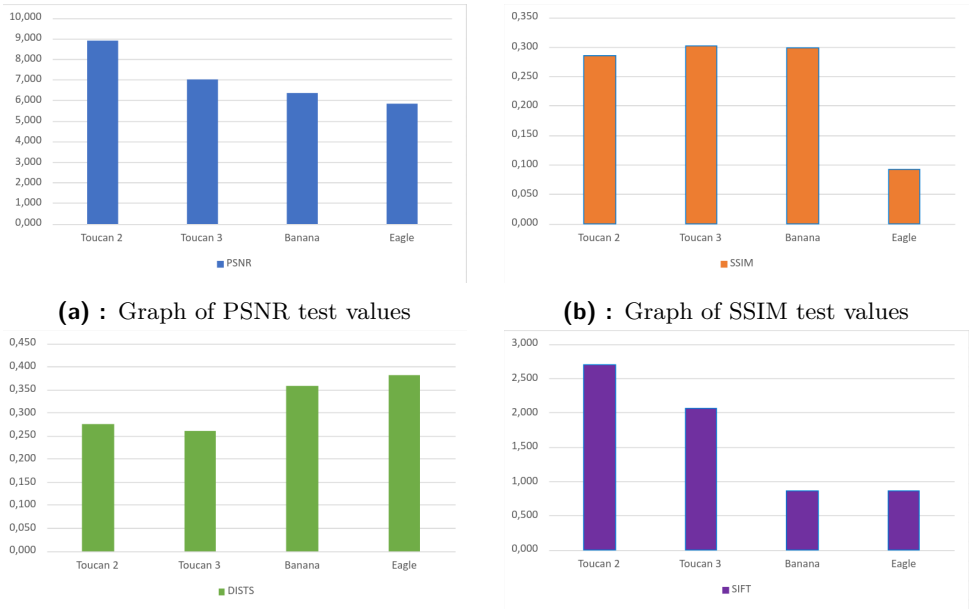
(d) : Banana **(e) :** Eagle

Figure 3.1: Images used to test image similarity methods

The results of our testing can be seen in Table 3.3. We added information about how long each method takes to add perspective. Each method has different scale and for better visualization we have created graph showing the values. Graphs can be seen in Figure 3.2

	Toucan 2	Toucan 3	Banana	Eagle	Time [s]
PSNR	8.890	6.992	6.370	5.836	0.128
SSIM	0.285	0.301	0.299	0.092	0.336
DISTS	0.276	0.260	0.359	0.382	9.174
SIFT	2.703	2.062	0.855	0.855	0.356

Table 3.3: Test results of PSNR for similarity assessment



(a) : Graph of PSNR test values

(b) : Graph of SSIM test values

(c) : Graph of DISTS test values*

(d) : Graph of SIFT test values

* The lower DISTS values represent higher similarity

Figure 3.2: Graphs showing the values from table 3.3

From the result graphs we can see that PSNR values are relatively close to each other and this could cause false identification of similarity.

We can further observe that SSIM does not detect different similarities for Toucan images and Banana. This is likely caused by similar colors and shapes of the Banana image.

The DISTS method has similar problem to PSNR as the values have low difference for similar and non-similar images. However, this problem is less significant for this method.

The SIFT method has high contrast between the similar and non-similar values.

Overall we choose to use the SIFT method as it has good distinction between similar and non-similar images. It also works relatively fast and so it won't increase the preprocessing time.

Chapter 4

Preprocessing of images

Before we start generating selections we need to extract information about image quality, image content and subsequently image similarity from images. This data is then used in a selection creation process. Data is saved for further use, this makes it easy to create different selection for already preprocessed sequence.

4.1 Image quality

For technical image quality Q_i^t we have decided to use BRISQUE. As we can see in the testing results in Table 3.2 BRISQUE does have slightly inferior results but is 9 times faster. We want our preprocessing to be done in reasonable times and slight drop in performance is acceptable trade off.

For aesthetic image quality Q_i^a we have decided to use NIMA approach.

The preprocessing of the images evaluates each image separately. The output is a list of all images of the sequence with values of Q_i^t and Q_i^a.

4.2 Image content

As we need to assess content similarity of two images we need to first extract the information about the image content. The preprocessing is done for each image separately and the output is a list of all images from the sequence with normalized vectors of image content.

4.3 Image similarity

Extracting information about similarity for every pair in the sequence leads to quadratic number of pairs in relation to the number of images in total. This would prove to be problematic for big sequences as preprocessing and then finding these similarities would require a lot of computing power. This needs to be avoided. The solution arose that we would only consider close neighboring images for similarity assessment. Number of neighbors for preprocessing is defined by parameter κ. For our purpose we set the parameter to 20.

neighbors - images that appear near the tested image in a sequence $\kappa = 20$ - number of tested neighbors,

For the structural similarity assessment we have chosen to use the SIFT approach as it detects the form of similarity that we are looking for. For content similarity we have decided to use cosine similarity of the content vectors. Cosine distance of vectors signifies the difference in vector orientation rather than just a pure relative distance that euclidean distance measures.

The preprocessing of image similarity consists of taking an image and calculating structural and content similarity with all of its neighbors separately. The output is a list of all pairs of all images with similarity values $S^s_{i,j}$ and $S^c_{i,j}$.

Chapter 5

Proposed methods for combining the extracted information

5.1 Selection parameters

From our preprocessing we extract values of qualities (Q_i^t, Q_i^a) and similarities ($S_{i,j}^s$, $S_{i,j}^c$). We have chosen to combine these values linearly into Q_i and $S_{i,j}$. $Q_i, S_{i,j} \in [0; 100]$.

The ratios in these linear combinations and the thresholds for quality and similarity influence the outcome in the methods that use them. We are going to call these the selection parameters and use the symbol θ_s to describe them. We will shortly introduce these parameters for easier understanding.

$$\theta_{\mathbf{s}} = (\mu_q, \tau_q, \mu_s, \tau_s) \text{ - selection parameters}$$

5.1.1 Technical and aesthetic quality ratio - μ_q

This parameter describes a percentage ratio in which the technical and aesthetic quality is combined to single score that defines the overall quality of the image. $\mu_q \in [0; 100]$.

$$Q_i = Q_i^t \cdot \frac{\mu_q}{100} + Q_i^a \cdot (1 - \frac{\mu_q}{100}) \tag{5.1}$$

5.1.2 Quality threshold - τ_q

Quality threshold is parameter describing the cutoff point for selection. Everything with quality score below this threshold will not be selected. $\tau_q \in [0; 100]$.

5.1.3 Feature and content similarity ratio - μ_s

This parameter describes a percentage ratio in which the feature and content similarity is combined to single score that defines the overall similarity between two images. $\mu_s \in [0; 100]$.

$$S_{i,j} = S^s_{i,j} \cdot \frac{\mu_s}{100} + S^c_{i,j} \cdot (1 - \frac{\mu_s}{100}) \tag{5.2}$$

5.1.4 Similarity threshold - τ_s

Parameters that defines limit of similarity. If similarity score between two images is higher than this number, those images are considered similar. $\tau_s \in [0; 100]$.

5.2 Method based on conditional statements

Main goal of this method is to allow user to select parameters θ_s. This allows user to influence the selection before it is created.

The method takes extracted image data and processes it with simple conditions that encode logic of selection. Selection settings can be set to output selection of specific size. If specific size is set the quality threshold τ_q is adapted accordingly.

The idea is that the algorithm selects images with a quality above the quality threshold τ_q. The image also needs to have the highest quality amongst all of its similar images to be selected. This can be represented by logic function.

$$z_i = Q_i > \tau_q \wedge \bigwedge_j (S_{i,j} < \tau_s \vee Q_i > Q_j) \tag{5.3}$$

The whole process can be summarized in these steps:

1. Combine technical and aesthetic qualities of all images in specified ratio defined by parameter μ_q to get Q_i.
2. Combine structural and content similarities of all pairs of images in specified ratio defined by parameter μ_s to get $S_{i,j}$.
3. Add unselected image with the highest quality Q_i into our selection.
4. For all potentially selected images that have similarity score $S_{i,j}$ with newly selected picture higher than τ_s remove them out of potentially selected. (These eliminated images are still used to determine the whether different image has the highest quality amongst its neighbors.)

5. Repeat steps **3.** and **4.** until all unselected images are bellow the quality threshold τ_q.

5.2.1 Finding optimal parameters

Method described in previous section requires user to input selection parameters θ_s. This can prove confusing and unintuitive at first, thus we generated pretrained input parameters which were found by optimizing F_1 score of training dataset (see section 1.2).

We decided to use F_1 score as the training dataset has unbalanced classes and this metric shows the balance between precision and recall.

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} \cdot \text{recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{5.4}$$

For training, we chose an approach that is inspired by grid search but is slightly optimized to avoid unnecessary computation. We start with defined vectors of possible values for each parameter within the range [0;100] with step size 10.

We make an initial guess of all values, then we choose a parameter and test values in the range of chosen parameter while the rest is set to initial guess. The parameter value with the highest F_1 score is saved as its best guess. The next step is to take a second parameter and do the same but the value of parameter that was already tested is set to best guess. This is repeated for all parameters until we end up with best guesses for all parameters.

We can create new smaller ranges for all parameters based on the previous best guess and reduce the step size to get even better guesses. This can be repeated any number of times to get more precise guesses.

With this technique we have found default parameters $\theta_\mathbf{s}$. The values optimizing F_1 score on our training dataset can be found in Table 5.1.

	μ_q	τ_q	μ_s	τ_s
Results	5	55	5	10

Table 5.1: Default parameters $\theta_\mathbf{s}$ found by brute forcing

Note: These default values will serve us as initial guess for method based on logistic regression

5.3 Method based on logical approximation

To create a logical approximation method, we take the logic function (equation 5.3) and approximate it with a function that is differentiable. To achieve that, we convert the logic function into purely conjunctions. We desire only conjunctives because it is easily translated into multiplication.

We approximate all inequalities by using sigmoid function. Sigmoid function $\sigma(z)$ is a function that maps any real-valued number to a value between 0 and 1. Input of positive numbers approaches output of 1 and negative input approaches output value of 1. Inputs close to zero approach output value of 0.5.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{5.5}$$

To compare two values we input their subtraction into sigmoid function and the output is approaching 1 if the first term of subtraction is bigger than the second term.

Our approximation looks as follows:

$$\hat{z}_i(\theta_\mathbf{s}) = \sigma(Q_i - \tau_q) \cdot \Pi_j(1 - \sigma(S_{i,j} - \tau_s) \cdot \sigma(Q_j - Q_i)) \tag{5.6}$$

Note: $\theta_\mathbf{s} = (\mu_q, \tau_q, \mu_s, \tau_s)$ *are selection parameters (see section 5.1)*

This function has gradient at every point, and so we can use gradient descent optimization to train the selection parameters $\theta_\mathbf{s}$.

We chose to optimize binary cross entropy (also known as log loss). More specifically its weighted form using class weights w_s and w_n (see Equations 1.1).

$$l(\theta_\mathbf{s}|\mathbf{y}) = -\frac{1}{N}\sum_{i=1}^{N} w_s \cdot y_i \cdot \log \hat{z}_i + w_n \cdot (1 - y_i) \cdot \log(1 - \hat{z}_i) \tag{5.7}$$

For optimization of parameters θ_s(see Section 5.1) we have decided to use the most commonly used optimization algorithm, gradient descent. More specifically we used Batch Gradient Descent (BGD) with update rule using Nesterov accelerated gradient algorithm [19].

For our BGD optimizer we used values in the Table 5.2

	Values
Learning rate η	0.001
Momentum γ	0.9

Table 5.2: Values of BGD optimizer

With all of this in place we trained our selection parameters $\theta_\mathbf{s}$ and for evaluating the quality of our training we used F_1 metric (see Equation 5.4).

Our training also requires initial guess of $\theta_{\mathbf{s}}$. For this we used values from the method based upon conditional statements (see Table 5.1). The usage of F_1 metric also helps us compare this method with the recommended parameters for our previous method. Our training losses can be seen in Figure 5.1 and close up of validation loss minimum in Figure 5.2 - this most likely signifies the start of overfitting. Learned parameters can be found in Table 5.3

	μ_q	τ_q	μ_s	τ_s	F_1
Initial guess	5	55	5	10	0.240
Results	5.066	54.811	1.630	16.140	0.291

Table 5.3: Results of logistic regression training

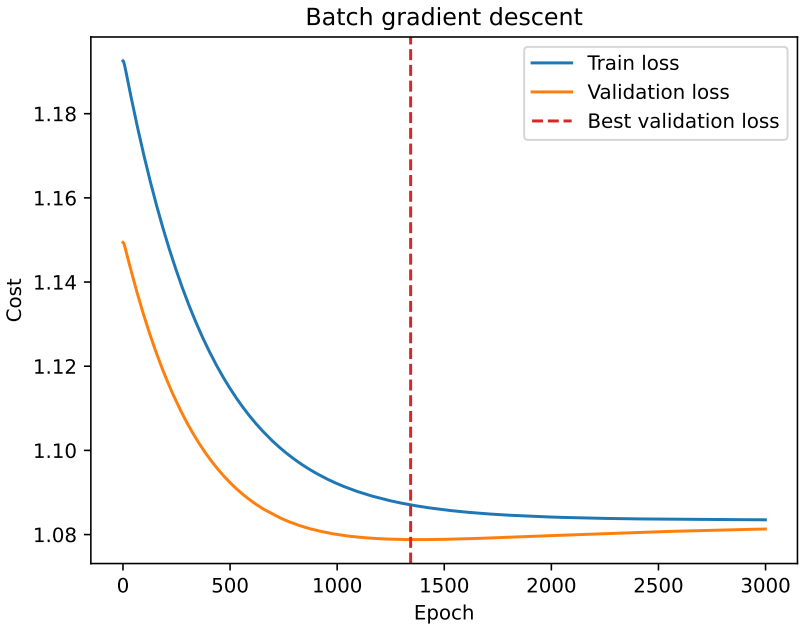


Figure 5.1: Optimizing of parameters for logical approximation

5.4 Neural network based method

5.4.1 Introduction to the method

With this method we hope to train our model of NN[1] to be able to process the extracted image data and to be able to decide whether an image is to be selected or not. The model should be able to learn a deciding logic that could be in theory very similar to logic used in previous methods (5.3). Although, there is the possibility that the model is going to be able to get more complex

[1]There are many books on topic of NN (e.q., "Deep Learning" by I. Goodfellow, Y. Bengio, and A. Courville or "Neural Networks and Deep Learning" by M. Nielsen)

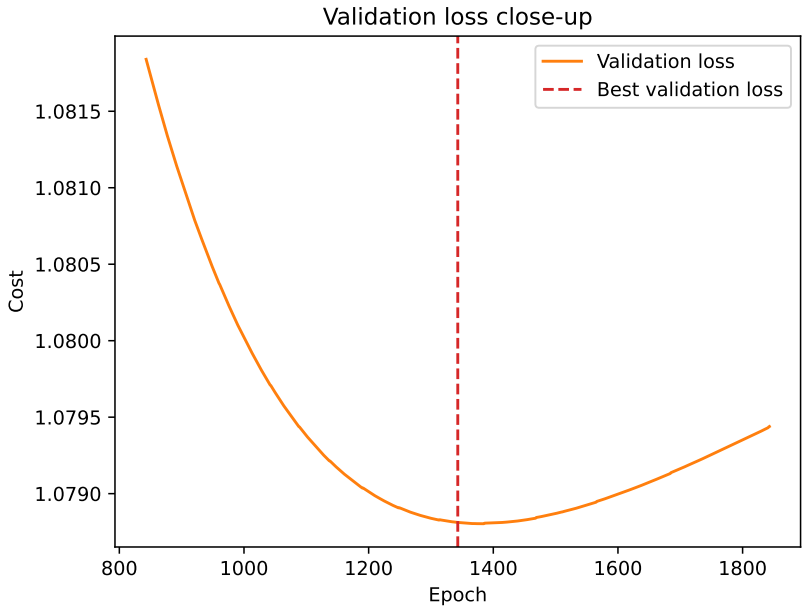


Figure 5.2: Close-up of validation loss at the lowest point

information from the extracted image data (see Chapter4) and will be able to guess the output with higher correlation to our testing data.

5.4.2 Model architecture and implementation

For our implementation we have to decide the architecture of our NN. The output layer is already determined because we aim to get binary output. The input layer is based on what data we want the model to have. It has to have representation of the judged image itself and representation of all potentially similar images. Judged image is represented by two quality values (see Chapter 3) and every potentially similar image is represented by two quality values and two similarity values. The maximum number of potentially similar images is double of κ (4.3), as we are testing images on both sides of the sequence. This gives us the shape of input layer.

$$2 \cdot \kappa \cdot 4 + 2 = 2 \cdot 20 \cdot 4 + 2 = 162$$

To decide the number and shape of hidden layer, based on complexity of our information we decided on a shallow network with only one hidden layer. The actual shape of the hidden layer was chosen arbitrary to 10 nodes. Visualization of the simple architecture on Figure 5.3 Our model has 172 trainable weights (input+hidden nodes).

We use gradient descent optimization for weights of our NN model, only this time we use a mini-batch gradient descent (MGD). This choice is made

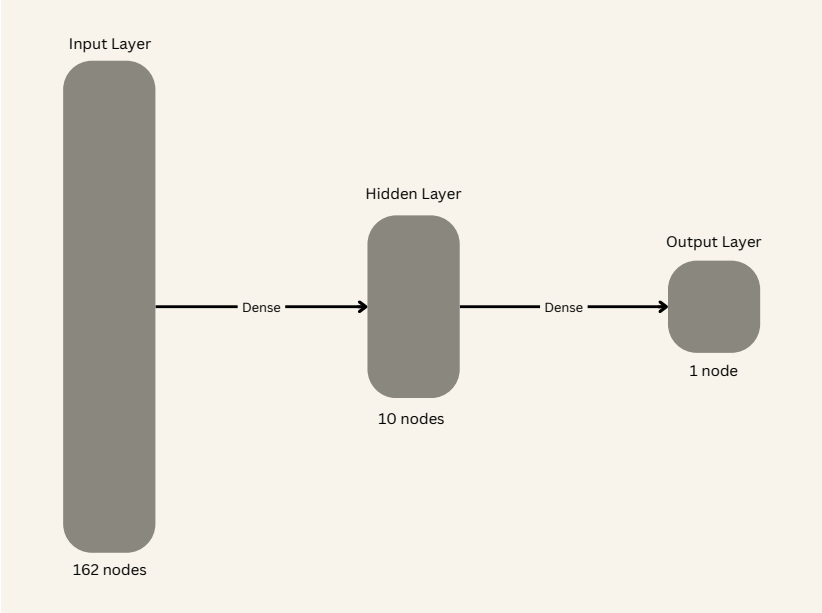


Figure 5.3: Architecture of our NN

with consideration that training of NN is more complex and can be slower. The same results should be however achieved with BGD as well. Our cost is the same as in previous method (Equation 5.7). As update rule we used RMSprop optimizer [2].

For our MGD optimizer we used values in the Table 5.4. The choice of learning rate is not as important as RMSprop optimizer is able to adapt it.

	Values
Learning rate η	0.00001
Momentum γ	0
Batch size	10

Table 5.4: Values of MGD optimizer

As in method based on logical approximation we used F_1 score to determine the quality of our model. Our training losses can be seen in Figure 5.4 and close up of losses can be seen in Figure 5.5.Comparison of cross validated F_1 metrics can be seen in Table 5.5

[2]You can read about RMSprop in its first proposition by Geoffrey Hinton in his lecture `www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`

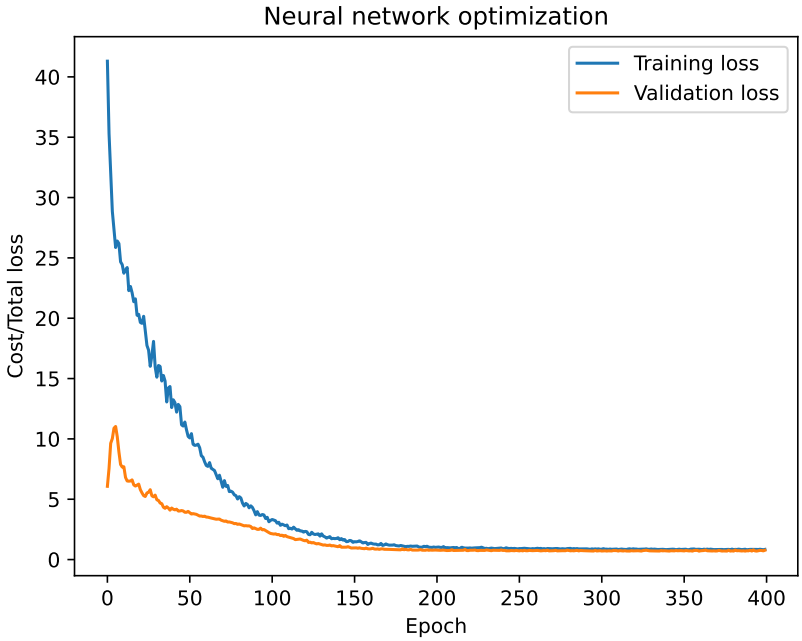


Figure 5.4: Training of parameters for NN

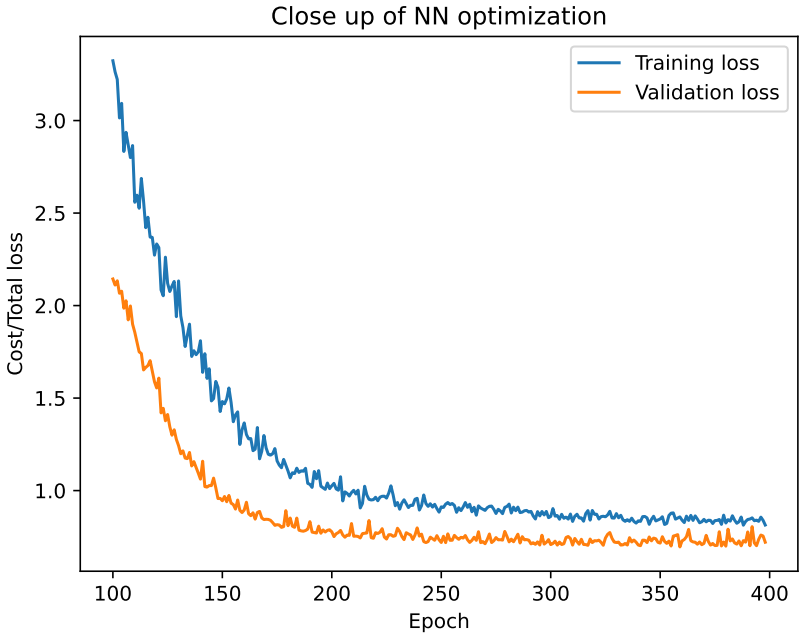


Figure 5.5: Close-up of losses in NN training

	F_1
Conditional statements	0.249
Logical approximation	0.291
Neural networks method	0.278

Table 5.5: Comparison of F_1 metrics of each of our method

Chapter 6

Additional functions and optional goals

6.1 Metadata

As one of our optional goals, we wanted to edit image metadata. This is useful for reasons such as easier sorting and searching through image collection or for compatibility with other platforms and programs.

Our software is working with JPEG images and there are a few metadata formats that are being used. Since we have decided to write information about image quality of each image, we used the metadata format EXIF which has metadata items rating (with five-star scale) and rating percent.

The image rating is derived from image quality Q_i. However, it would not be very useful if we took the image quality and directly wrote it into metadata, as the image quality is usually condensed in smaller portion of the scale. With this in mind, we have decided that for each selection we will limit the image quality information from lowest to highest rated image and rescale it to be in range from 0 to 100. We will call this rescaled image quality metadata rating and use symbol M_i.

$$M_i = \frac{Q_i - \min Q_i}{\max Q_i - \min Q_i} \cdot 100 \tag{6.1}$$

To achieve five-star scale we divide the metadata rating into five sections and give stars based upon the section the image ends up in.

6.2 Interactive selections and updating of our models

After any selection is generated by any of our methods, the user can choose to edit the selection by selecting unselected images or deselecting images already selected. This creates a new selection that is closer to user's preference.

The logical approximation model and neural network model allow for updating. The update is done by optimizing the model on a new selection created by the user. The process of updating can be done multiple times with different edited selections to achieve desirable models.

Chapter 7

Software and GUI

The software can be found on: `https://gitlab.fel.cvut.cz/bartulu2/photo_culling`.

This software was developed in Python and the graphical user interface for this software was developed with usage of PySimpleGUI library. As the usage of the software might be confusing at first we created a little manual with descriptions of the GUI to improve the user experience.

7.1 Manual GUI

To start the GUI version simply use *python gui_main.py*.

Our GUI is separated into three parts; settings, lists and image view with buttons. The three parts can be seen in Appendix A in Figure A.4 with detailed view in Figure A.1, A.2 and A.3.

To create a selection, the user needs to first select a folder with the input sequence X. The user has the option to use three of our proposed methods: conditional statements, logical approximation or neural network. Afterwards, the user needs to decide whether to specify an output size. The output size parameter sets the size to a percentage of the whole sequence.

For the method of conditional statements, the user can decide the parameters θ_s either by preset values or by using the advanced option of setting the parameters precisely. There is also the option of using recommended settings that was optimized for our training dataset. To start the process of generating, the user needs to press the generate button.

At the bottom of the settings part of our GUI there is a window informing about the selection process. Above the window there are two checkboxes. The first checkbox gives the user the option to write in to the metadata the

information about rescaled metadata rating (see Chapter 6.1). The second checkbox gives the user the option to force the process to run on CPU (priority is on GPU if available).

The process starts with the preprocessing of image information for sequence (see Chapter 4). After the preprocessing is finished, the culling process will start and shortly after generated selection will appear in the top listbox. The bottom listbox contains the images that were not selected. After clicking on any of the image names from the listbox, the image will appear. A selection can be then adjusted by two buttons by the listboxes. The user can also move through the lists by arrows near select/deselect buttons.

If the user wishes, they can update the models for automatic selection based on their adjusted selection. This will result in the automatic models more closely approximating user's preferences (this process might have to be repeated multiple times as the models will better approximate user's preference with more data).

Final selection can be exported in the form of a list of names or all of the selected images can be copied into a new folder based on the user's choice.

7.2 Manual console version

This software is mainly developed around the GUI version but can be also used in console version with few feature limitations. To use this version the user has to start the program with arguments. All usable arguments with short description are listed bellow. Console version is found in *console_main.py*.

-dir 'path/to/folder' - defines the folder that contains all the images

-man_log / -auto_reg / -auto_nn - chooses which method will be used

-recommended - uses recommended parameters for manual settings

-q_t 'value' - sets the parameter τ_q, default value - 50

-s_t 'value' - sets the parameter τ_s, default value - 10

-t_a_rattio 'value' - sets the parameter μ_q, default value - 50

-s_c_rattio 'value' - sets the parameter μ_s, default value - 50

-size_based - specifies that the user wishes to specify the output size

-size 'value' - sets the output size if selection is size based, default value - 10

-metadata - process will write into image metadata

-save - saves the selection list into the image folder

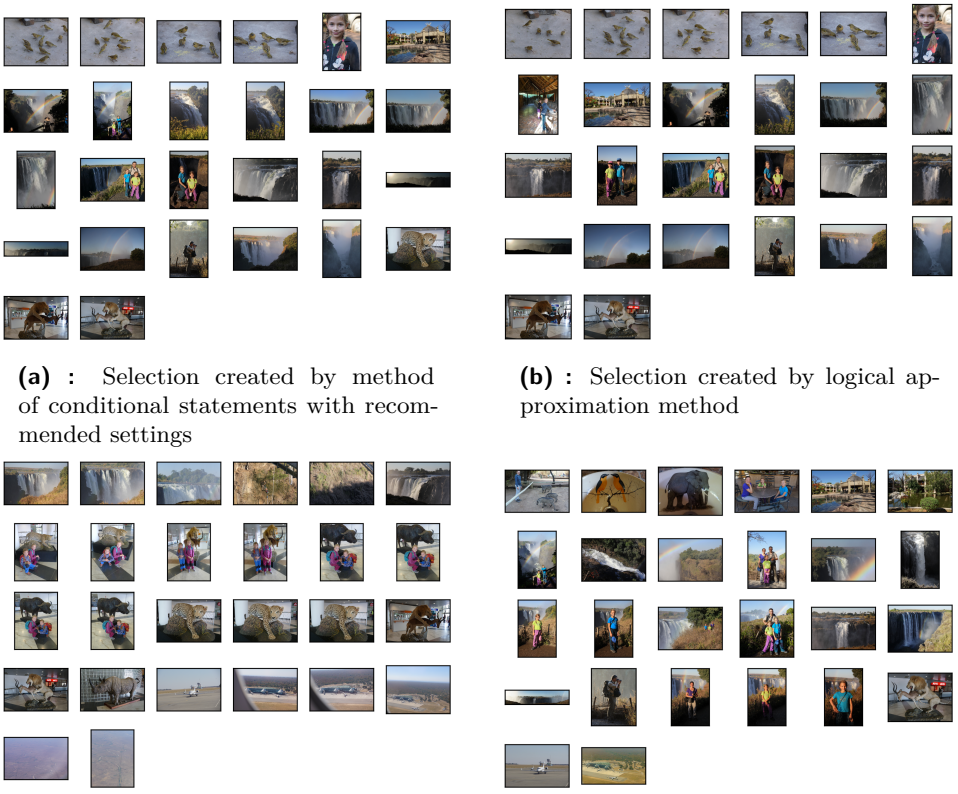
-copy - copies the selection into folder for selected images

Example of console version usage: *python3 console_main.py -dir images/test_images -auto_nn -size_based -size 10 -copy*

Chapter 8

Evaluating our software

For the evaluation of our software we have chosen to generate selections by all three of our proposed methods as well as creating one ourselves and then comparing these selections. We have done this for two sequences for more information.



(a) : Selection created by method of conditional statements with recommended settings

(b) : Selection created by logical approximation method

(c) : Selection created using neural network based method

(d) : Selection created by human

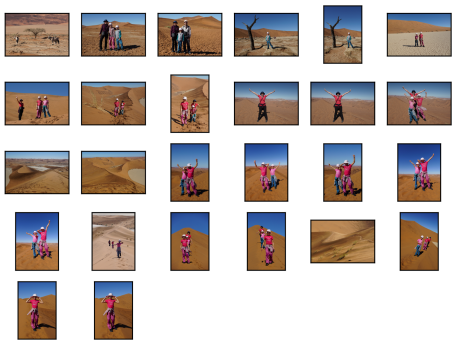
Figure 8.1: Victoria Falls - Showcase of selections

The first sequence included pictures from Victoria Falls. The original sequence

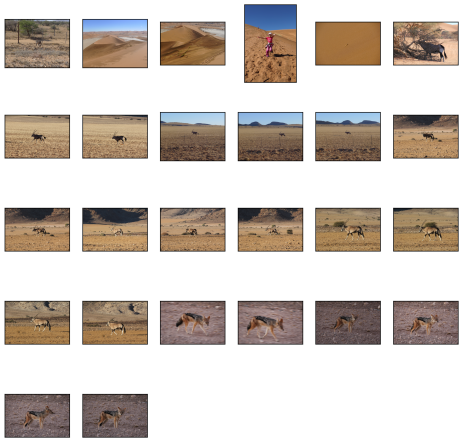
has 438 images and the selections were limited by size to 6 percent producing selection of 26 images. Three generated selections can be found on Figures 8.1a, 8.1b and 8.1c. Selection that was created by human can be seen on Figure 8.1d.

The second sequence included pictures from famous Namibian salt pan Sossusvlei. The original sequence has 265 images and the selections were limited by size to 10 percent producing selection of 26 images. Three generated selections can be found on Figures 8.2a, 8.2b and 8.2c. Selection that was created by human can be seen on Figure 8.2d.

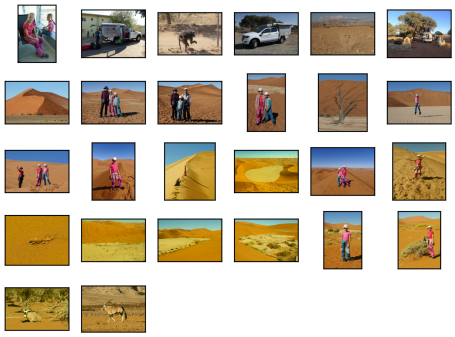
(a) : Selection created by method of conditional statements with recommended settings



(b) : Selection created by logical approximation method



(c) : Selection created using neural network based method



(d) : Selection created by human

Figure 8.2: Sossusvlei - Showcase of selections

8.1 Objective analysis

To analyze created selection we will look at how many common elements selections have with the human created selection. We will also look at created

selections and decide if the selections fulfill requirements that we have set for ourselves. The table of common elements can be seen in Table 8.1.

	CS*	LA**	NN
Victoria Falls	2	3	3
Sossusvlei	3	2	2

*CS = Conditional statements
**LA = logical approximation

Table 8.1: Table of common elements with the human made selection

As we can see the number of images that appear in both our generated selections and human made one is pretty low. This does not necessary mean that our selections are bad. It could simply show the difference between in preference.

If we look at each generated selection separately we can see that there are images that are similar to each other. This is something that we wanted to avoid. One possible explanation is that our training dataset was not high quality and that it included similar images. However, after inspection of the dataset we have not found many similar images in training selections.

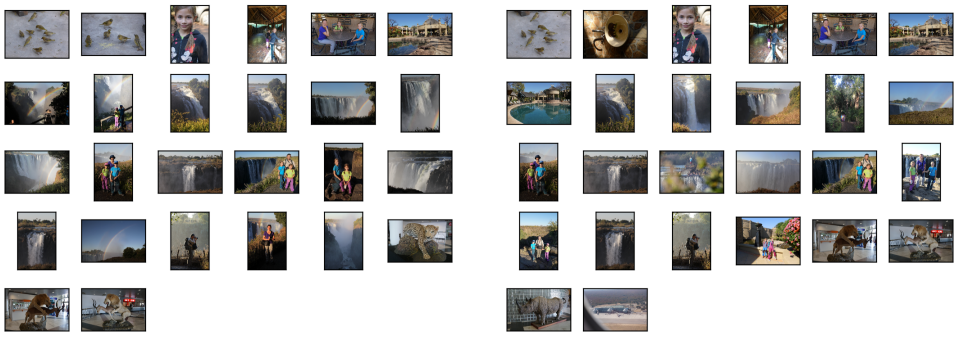
Another possible explanation is that our generated selections are more focused on eliminating content similarity and images that are structurally similar are overlooked. For this to be plausible explanation it would mean that our content labeling and henceforth content similarity is not working as intended.

8.1.1 Testing the possibility of faulty content similarity

We will test our hypothesis that the content similarity is not filtering similar pictures. Test will be conducted with Victoria Falls sequence as input as this sequence generates more similar images in its selections.

What happens if we manually change the parameters to only take structural similarity into consideration when generating selections. Since this involves changing of parameters, this test only applies to method of conditional statements and logical approximation. Selections generated with these changes can be seen in Figure 8.3 The number of common elements with human made selection can be seen in Table 8.2.

As we can see in there is improvement in how many similar images appear in the generated selections. This is especially noticeable in the selection generated by the method of logical approximation, which now seemingly includes no similar images.



(a) : Selection created by method of conditional statements with recommended settings and forced structural similarity

(b) : Selection created by logical approximation method with forced structural similarity

Figure 8.3: Selections created with forced structural similarity

	CS*	LA**
Victoria Falls	5	4

*CS = Conditional statements
**LA = logical approximation

Table 8.2: Table of common elements with the human made selection with forced structural similar

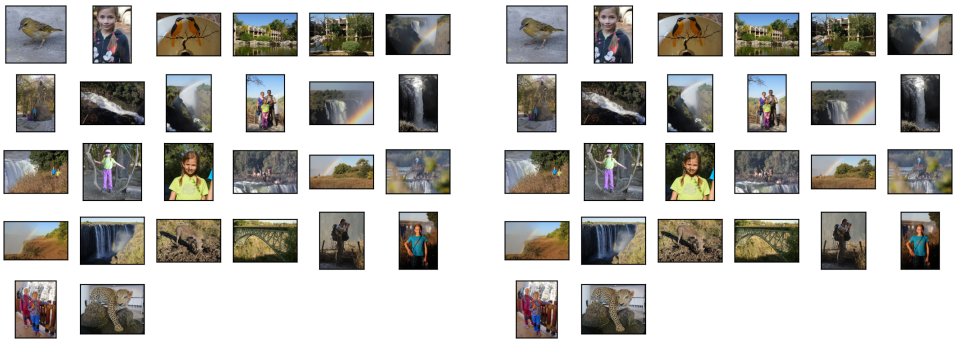
The Table of common elements also shows that these selections are closer to the selection that was human made.

8.1.2 Further comparisons

To further compare our selections, we have asked a volunteer to pick selection for each sequence. These selections can be seen in Figure 8.4. The table showing common elements with selections generated by our proposed methods, our handpicked selection and selection created by our volunteer can be seen in Table 8.3.

There is also option to compare our selections with selections created by tested software in Section 2.2. However, this would not be fruitful as the selections generated by tested software are much larger even when generated with the strictest software (the smallest selections were generated by Aftershoot, with size of roughly 50% of the whole sequence). The tested software does not offer any way to set the size of the output.

As we can see the selection created by our volunteer shares only a small amount or no images with any other selections. This proves that selection process is very subjective and complex.



(a) : Sossusvlei - selection created by our volunteer

(b) : VictoriaFalls - selection created by our volunteer

Figure 8.4: Selections created by our volunteer

	CS*	LA**	NN	Handpicked
Sossusvlei	0	0	2	15
Victoria Falls	2	2	1	9

*CS = Conditional statements
**LA = logical approximation

Table 8.3: Table of common elements with the selection created by our volunteer

8.2 Quality evaluation

Quality evaluation of our selections was conducted in the form of a survey, where we have given volunteers all four selections without any additional information and asked them to decide which one of these selection is the best in their opinion.

Our survey was concluded with 17 volunteers and the results can be seen on the graphs on Figures 8.5 and 8.6

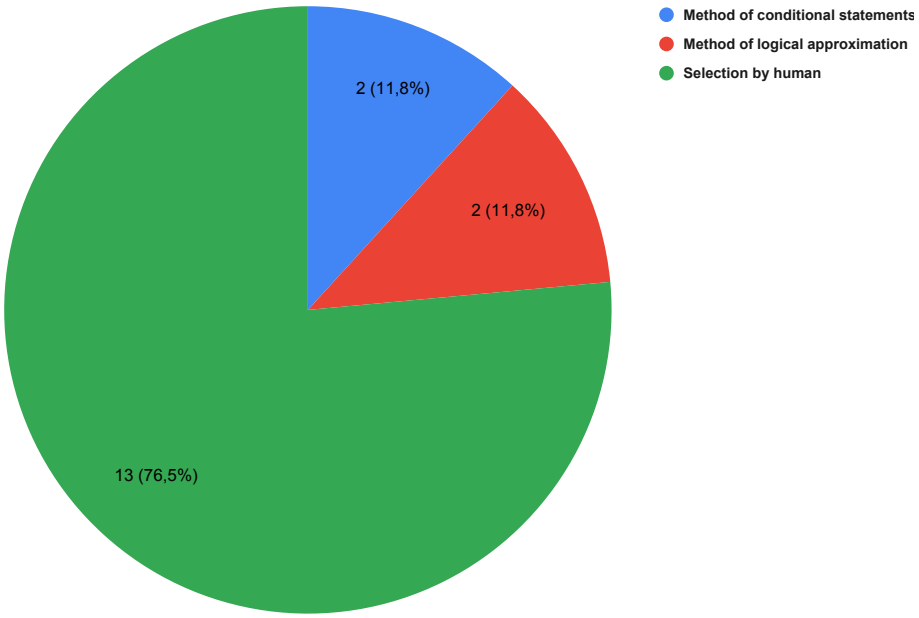


Figure 8.5: Victoria Falls - Survey results

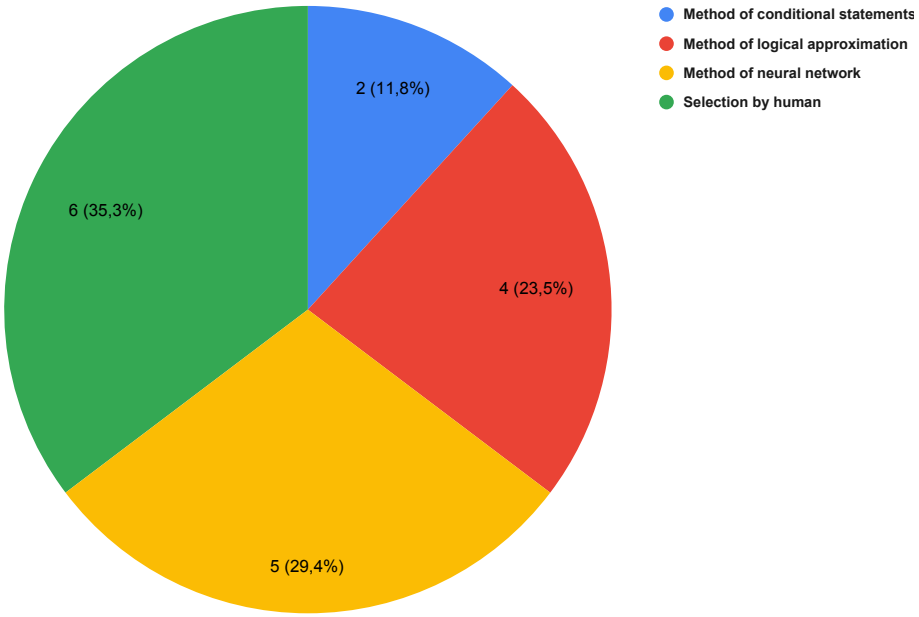


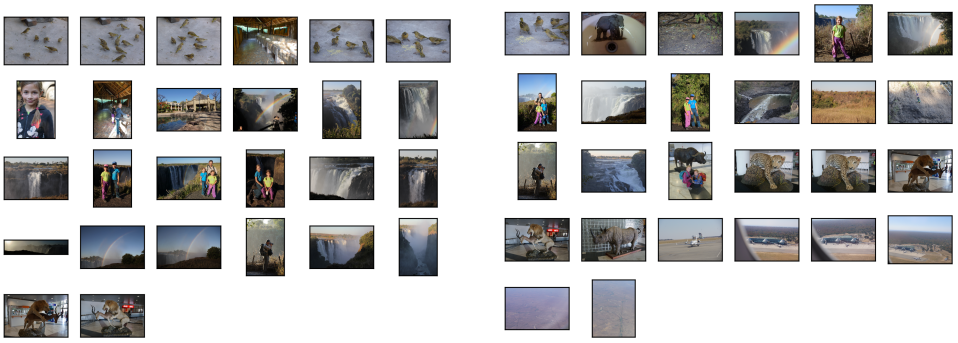
Figure 8.6: Sossuvlei - Survey results

The responses also included additional information. Three respondents stated that they prefer the Sossuvlei selection that features more animals which was created by our neural network. Another three respondents stated that they liked the sharpness and high quality images in the selection that was created by logical approximation.

8.2.1 Updating the model to more closely generating liked selection

The survey results for our first sequence (see Figure 8.5) show that the majority of people thought that the human created selection was the best. We want to test whether our automatic methods (logical approximation and neural networks methods) can be trained to closer resemble this human selection.

We have trained both models with the human-made selection (Figure 8.1d) as our reference; we are trying to update the models so that the output is closer to the reference. After training the models we have generated selections on the trained models (see Figures 8.7a and 8.7b)



(a) : Updated selection created by logical approximation

(b) : Updated selection created by neural network based method

Figure 8.7: Victoria Falls - updated selections created by automatic methods

We have also looked at selections of double the size (52 images) generated the same way to see if there is any change in a bigger selection. The results showing number of common images with the human selection (Figure 8.1d) can be seen in Table 8.4.

	LA*	Trained LA	NN	Trained NN
Original size (26 images)	3	3	3	6
Double size (52 images)	5	5	3	13

*LA = logical approximation

Table 8.4: Table of common elements with the selection made by human showing training capabilities of our proposed automatic methods

We can observe that the method using neural network shows ample trainability.

From the table, we can see that the method of logical approximation does not show any improvements after having been trained. This could be explained by

the fact that the method is based on logical function and has a firm structure that cannot be altered easily. It could also be explained by the fact that the training process is not working as intended. We have looked at the influence of updates on this method and observed only small changes in parameters after each update. The training shows unexplained bias towards content similarity. We were not able to find out what is causing this bias or the resistance to training.

Chapter 9

Conclusion

We set out a goal of creating a software capable of selecting representative images out of larger sequences. Our choice was to extract information about the quality of each image and its similarity with images that are neighboring it. We researched the software and solutions that have been developed, approaches to image similarity and quality extraction.

With selected approaches for data information extraction, we have proposed three methods of creating a selection. One that allowed for user input in form of selection settings and two automatic, which produce selections without the need for user input. These automatic methods were designed in a way that allows the underlying models to learn from user's preferences. We also included an option for writing the relative image rating into each image's metadata.

To improve the user's experience and to make the software more robust, we have created a graphical user interface which allows users to easily choose one of the proposed selection methods and then interactively view and alter the generated selection.

Lastly we evaluated the proposed methods. We generated selections by each method for two separate sequences and compared it to a selection that was handpicked by a human. The comparison had been done through a survey wherein we asked individuals to pick which of the selections they liked the most. We proved that selections generated by our methods can compete with handpicked solutions with regard to human judgment. We also tested the trainability of our automatic methods and showed the capabilities of the method using neural networks to generate selections closer to the preference of the user.

Appendix A

Detailed view of GUI

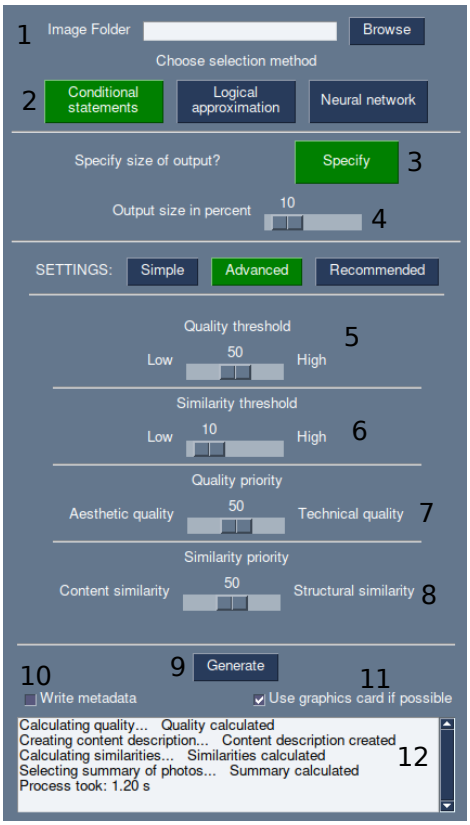


Figure A.1: Detail of settings

1. Image folder selection
2. Method selection
3. Button to select whether to specify size or use no limitation
4. Setting of size in case of size specification

Figure A.2: Detail of selection lists

5. Setting of parameter τ_q
6. Setting of parameter τ_s
7. Setting of parameter μ_q
8. Setting of parameter μ_s
9. Button to generate selection
10. Checkbox giving the option to write rating for each image into it's metadata
11. Checkbox giving the option to prefer using the CPU or GPU
12. Output box showing progress of the selection process
13. List of selected images
14. List of unselected images
15. Button to navigate up in list of selected images
16. Button to move selected image into unselected

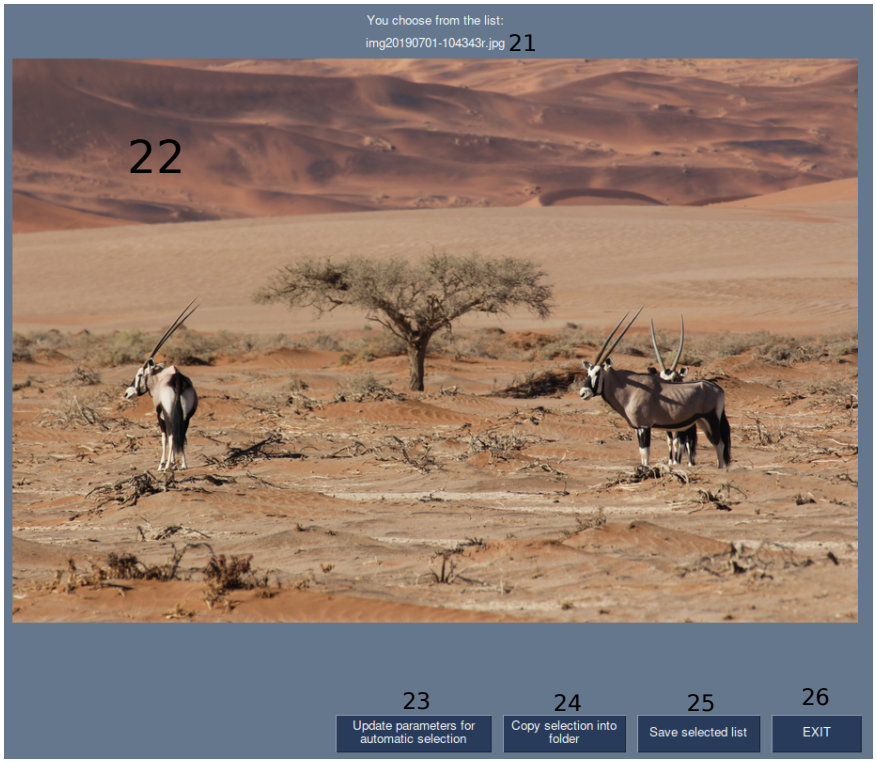


Figure A.3: Detail of image view and buttons

17. Button to navigate down in list of selected images
18. Button to navigate up in list of unselected images
19. Button to move unselected image into selected
20. Button to navigate down in list of unselected images
21. Name of displayed image
22. Displayed image
23. Button to updated models of automatic selection
24. Button to copy selected images into new folder
25. Button to create list of selected images
26. Button to exit the program

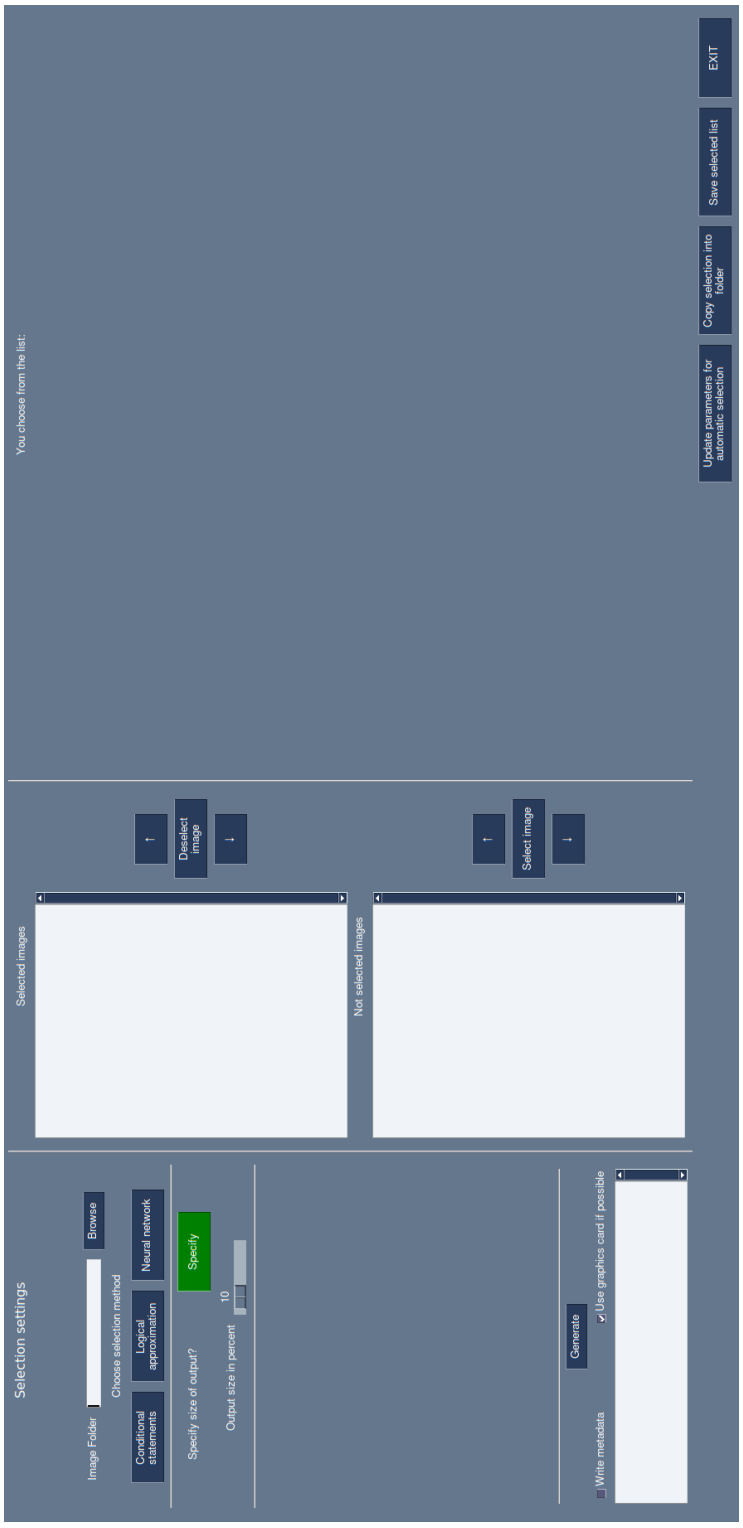


Figure A.4: Three parts of the program

Appendix B

IQA metrics used

B.1 Pearson's Linear Correlation Coefficient (PLCC)

The PLCC is a statistic that captures linear correlation between the predicted score and Mean opinion score. It has range of [-1;1]. It can be defined as:

$$PLCC = \frac{\sum_{i=1}^{n}(x_i - \hat{x})(y_i - \hat{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \hat{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \hat{y})^2}} \tag{B.1}$$

where n is the number of the considered samples, x_i and y_i are the sample points, $\hat{x}$ and $\hat{y}$ are the means of each sample distribution.

B.2 Spearman's Rank-order Correlation Coefficient (SROCC)

The SROCC uses the ranks rather that relative distance to assess relationship between prediction and ground truth. It has range of [-1;1] and is defined as:

$$SROCC = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2 - 1)} \tag{B.2}$$

where n is the number of the considered samples, d_i is difference between two ranks of each sample ($d_i = \text{rank}(x_i) - \text{rank}(y_i)$).

B.3 Kendall rank correlation coefficient (KRCC)

The KRCC uses the ranks to measure rank correlation. It has range of [-1;1] and can be defined as:

$$\tau = \frac{2}{n(n-1)} \sum_{i<j} \text{sgn}(x_i - x_j)\text{sgn}(y_i - y_j) \tag{B.3}$$

where n is the number of samples, x_i and x_j are samples from one list and y_i and y_j are samples from another list.

Appendix C

Bibliography

[1] J. Li, H. Lim, and Q. Tian, "Automatic summarization for personal digital photos," in *Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint*, 01 2004, pp. 1536 – 1540 vol.3.

[2] P. Sinha, H. Pirsiavash, and R. Jain, "Personal photo album summarization," in *Proceedings of the 17th ACM International Conference on Multimedia*, ser. MM '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 1131–1132. [Online]. Available: https://doi.org/10.1145/1631272.1631533

[3] M. Batko, P. Budikova, P. Elias, and P. Zezula, "Clan photo presenter: Multi-modal summarization tool for image collections," in *Proceedings of International Conference on Multimedia Retrieval*, ser. ICMR '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 541–542. [Online]. Available: https://doi.org/10.1145/2578726.2582623

[4] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, pp. 145–175, 2001. [Online]. Available: https://api.semanticscholar.org/CorpusID:11664336

[5] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.1136800

[6] S. Nowak, R. Paduschek, and U. Kühhirt, "Photo summary: Automated selection of representative photos from a digital collection," in *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, 04 2011, p. 75.

[7] H. Talebi and P. Milanfar, "Nima: Neural image assessment," *IEEE transactions on image processing*, vol. 27, no. 8, pp. 3998–4011, 2018.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] J. Wang, K. C. Chan, and C. C. Loy, "Exploring clip for assessing the look and feel of images," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 2555–2563.

[10] A. Mittal, A. K. Moorthy, and A. C. Bovik, "Blind/referenceless image spatial quality evaluator," in *2011 conference record of the forty fifth asilomar conference on signals, systems and computers (ASILOMAR)*. IEEE, 2011, pp. 723–727.

[11] N. Ponomarenko, L. Jin, O. Ieremeiev, V. Lukin, K. Egiazarian, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, and C.-C. Jay Kuo, "Image database tid2013: Peculiarities, results and perspectives," *Signal Processing: Image Communication*, vol. 30, pp. 57–77, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0923596514001490

[12] V. Hosu, H. Lin, T. Sziranyi, and D. Saupe, "Koniq-10k: An ecologically valid database for deep learning of blind image quality assessment," *IEEE Transactions on Image Processing*, vol. 29, pp. 4041–4056, 2020.

[13] C. Chen and J. Mo, "IQA-PyTorch: Pytorch toolbox for image quality assessment," [Online]. Available: https://github.com/chaofengc/IQA-PyTorch, 2022.

[14] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International conference on machine learning*. PMLR, 2021, pp. 10 096–10 106.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[16] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[17] K. Ding, K. Ma, S. Wang, and E. P. Simoncelli, "Image quality assessment: Unifying structure and texture similarity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 5, pp. 2567–2581, 2022.

[18] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[19] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o($1/k^2$)," in *Doklady NASSR (translated as Soviet.Math.Dokl.),*, 1983, pp. 543–547. [Online]. Available: https://api.semanticscholar.org/CorpusID:202149403

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Bartůněk Lukáš** Personal ID number: **498933**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Photo Culling - Selecting a Representative Set of Photographs

Bachelor’s thesis title in Czech:

Výběr reprezentativního souboru fotografií

Guidelines:

Given a large set of photographs, automatically select a smaller subset that contains the best photographs in terms of interest, technical quality and aesthetics, is representative and avoids duplicates or near duplicates. The resulting tool should run on both Linux and Windows, be able to process thousands of images in reasonable time and be usable within standard photographers’ workflow.
1. Familiarize yourself with existing tools and approaches in this domain and suggest a suitable architecture for the photo selection tool.
2. Test existing approaches for evaluating image quality, image similarity and image content.
3. Evaluate several methods for combining extracted image information for photo selection.
4. Implement an easy to use software tool for photo selection, allowing the user to set basic parameters of the process.
5. Experimentally evaluate the performance of the developed software by comparing it with other existing tools, by comparing it with human performance, or by evaluating user satisfaction.
6. [Optional] Add the ability to read and write image metadata for easier integration with standard tools.
7. [Optional] Create a graphical user interface allowing to interactively examine and alter the selection process.
8. [Optional] Improve the selection process by learning from a (possibly large) dataset of manually selected photographs.

Bibliography / sources:

[1] Nowak et al: Photo summary: automated selection of representative photos from a digital collection. ICMR '11: Proceedings of the 1st ACM International Conference on Multimedia Retrieval, April 2011
[2] M. Batko, P. Budikova, P. Elias, P. Zezula: CLAN Photo Presenter: Multi-modal Summarization Tool for Image Collections. ICMR 2014
[3] Talebi, Milanfar: NIMA: Neural Image Assessment. IEEE Transactions on Image Processing 27, no. 8, 2018
[4] Sinha, Pinaki & Pirsiavash, Hamed & Jain, Ramesh.. Personal photo album summarization. ACM international conference on Multimedia. 2009

Name and workplace of bachelor's thesis supervisor:

prof. Dr. Ing. Jan Kybic Biomedical imaging algorithms FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **04.09.2023** Deadline for bachelor thesis submission: **09.01.2024**

Assignment valid until: **16.02.2025**

prof. Dr. Ing. Jan Kybic Supervisor's signature	prof. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature