Czech Technical University in Prague
Faculty of Information Technology

ASSIGNMENT OF MASTER'S THESIS

Title:	Learning Convolutional Neural Networks for Age Estimation
Student:	Bc. Ondřej Novák
Supervisor:	Ing. Vojtěch Franc, Ph.D.
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Theoretical Computer Science
Validity:	Until the end of winter semester 2018/19

Instructions

The CNNs achieve state-of-the-art performance in many face recognition tasks including age estimation. The good performance requires a large set of annotated facial images. The public annotated databases have small or medium size and often cover a limited age distribution. The goal of this thesis will be to overcome the problem by designing a method for learning CNNs from weakly annotated examples that can be collected by an automated process. The first task will be to program a crawler downloading facial images from the Internet along with a profile information related to the age. The second task will be to develop an algorithm able to improve performance of the CNN by learning from the automatically downloaded data. A prediction accuracy of the CNN learned from weakly annotated examples will be compared against CNNs learned from existing public databases.

References

Will be provided by the supervisor.

[illegible]
doc. Ing. Jan Janoušek, Ph.D.
Head of Department

[illegible]
prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague February 20, 2017

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Learning Convolutional Neural Networks for Age Estimation

Bc. Ondřej Novák

Department of Theoretical Computer Science
Supervisor: Ing. Vojtěch Franc, Ph.D.

May 9, 2018

Acknowledgements

I would like to thank my supervisor, Vojtěch Franc, for his guidance throughout the whole work.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on May 9, 2018 . . . .

Citation of this thesis

Novák, Ondřej. *Learning Convolutional Neural Networks for Age Estimation.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018. Also available from: ⟨`http://cmp.felk.cvut.cz/~xfrancv/pages/wikipeople.html`⟩.

Abstrakt

V této práci jsme vytvořili nový dataset pro odhadování věku, Wikipeople databázi, sestávající se celkem z 217800 příkladů. Celý proces získávání databáze je do detailu popsán, tudíž je možné na tuto práci v budoucnu navázat a rozšířit existující dataset stejným způsobem, jako byl vytvořen. Výsledná získaná anotace je statisticky ověřena. Úloha odhadu věku je dále formulována jako klasifikační problém a podle toho jsou natrénovány konvoluční neuronové sítě. Přesnost modelu je vyhodnocena na sadě standardních benchmarků. Výsledky ukazují, že Wikipeople databáze představuje náročná data.

Klíčová slova odhad věku, konvoluční neuronové sítě, Wikipeople databáze

Abstract

In this work we put together a new dataset in the field of age estimation, the Wikipeople database, consisting of 217800 samples. The whole process of getting the database is described in detail so it is possible to follow up in the future and extend the dataset in the same way as was created. The correspondence of the resulting labels to the truth is statistically evaluated. Further, the age estimation task was formulated as classification problem and a set of convolutional neural networks were trained using this dataset in the respective manner. The performance is evaluated on set of standard benchmarks. The results suggest the Wikipeople dataset being a challenging one.

Keywords age estimation, convolutional neural networks, Wikipeople database

Contents

List of Figures

List of Tables

Introduction

Estimating human age is an interesting problem to study, both *per se* and because it has many direct applications ranging from forensics to retail business. The use in social media may be another example. Social sites may in addition benefit from access to a large set of annotated samples. This is however not the case in this field as far as we consider the publicly available datasets. The state-of-the-art applications are built based on datasets of limited qualities. Such quality may be the number of samples in the respective datasets, or the actual quality of the contents of the database which may exhibit certain non-standard features. Thus, one of the possible ways of how to push research in this area forward would be to create a new dataset and if possible to do it in an automated fashion so that it could be potentially extended in the future.

So far, we can deal with the following databases. FG-NET database [1], consisting only of 1002 facial images which are of a low quality by nowadays standard. MORPH database [2], containing a lot more samples, 55000, but on the hand it has significantly biased apparent age as the subjects of the database are criminal suspects. Later, even larger databases emerged. For example the IMDB-WIKI database [3] contains 524230 samples (460K IMDB + 60K Wikipedia) but yet again these images have specific distribution as the IMDB is actually comprised of only celebrities. But the database is created in an interesting way, not assessing the age directly, rather exploiting additional infomation (EXIF metadata) and then annotating the subjects with age based on a known year of birth and extracted information about the date of the image.

Face detection in this case is done automatically using an Adaboost detector which returns multiple detections per image. To get to the target face they used a heuristic approach of selecting the dominant face (in case there is one such). This way they get to a sample of 260K facial images labeled with age and gender. IMDB-WIKI is currently the largest age-gender database and is used across many state-of-the-art applications. For example, it was used by winners [3] of ChaLearn Looking at People Challenge 2015 [4] as well as by winners [5] of the follow up competition ChaLearn LAP 2016 [5], and many recent works e.g. [6, 7]. A disadvantage of this dateset is actually the unknown quality of the automatically generated annotations. As a result of this, the following works [3, 5, 6, 7] use IMDB-WIKI only for pre-training, splitting the learning into separate phases, and only then fine tuning on the target database. Recently [8] proposed a way of how to clean the dataset by exploiting the presence of multiple images corresponding to the respective subjects, learning an identity model and later using link the annotation more precisely. This is however possible only for the IMDB part of the dataset as the WIKI part does not contain such multiple samples of the subjects.

On the other hand, there are also works proposing an end-to-end learning, such as [9] with a Ranking-CNN model based on a series of binary CNNs each of which is trained with *ordinal* age labels. The prediction is then made by aggregating the respective binary outputs. The evaluation benchmark is Morph. [10] propose another CNN model for ordinal regression problem, similarly to [9] where it is transformed into a series of binary classification subproblems using a CNN with multiple outputs. There is also a new dataset created as part of the work: Asian Face Age Dataset (AFAD) with 160K+ faces. The benchmark of choice was the AFAD dataset and Morph. [11] approaches the problem differently as it at first extracts face features by a pre-trained VGG-Face model (a face descriptor), then they train Kernel Extreme Learning Machine with the extracted features as inputs. Benchmark dataset is ChaLearn. [12] introduce multi-scale analysis to CNN architecture and they use Morph in the experiments. [13] extract features from different layers of their CNN instead of the last layer as usual. Experiments carried out on Morph and FG-NET. [14] proposes a simple CNN for age and gender prediction with the subsequent evaluation on ADIENCE database. [15]

Please pardon the typographical quality of this work and rather refer to the project website for an updated version.

Chapter 1

Database

1.1 Overview

There has been created a new database as part of this work. The database is called *Wikipeople* and is made publicly available at: `http://cmp.felk.cvut.cz/~xfrancv/pages/wikipeople.html`

It based on 1231822 Wikipedia articles [1]. These articles come from Wikipedia categories of:

- People who were in the category *Living people* [2]
- People who were either born or died in or after 1900 [3]

All of that as of April 2017 when these identifiers were collected. It is about items contained in the respective categories which are expected to be people's biographies but there are still outliers in a sense that you can come across an article concerning non-humans (i.e. animals) and/or lists of people etc.

These source HTML files of the respective articles were downloaded and are also available for reuse in an compressed archive. Additionally, 646134 images were downloaded (belonging to 396628 articles) and in 460085 of those there was a human face detected. In total, 366978 single-detections belonging to 299471 articles (i.e. different people).

Additional information were also collected during the process of downloading the respective articles and images. Each image has its own separate Wikipedia page including details about the file—these pages were also downloaded in the same format as the articles, they are available for further use in a form of source HTML files.

[1] Precisely, 1231822 unique Wikipedia article identifiers

[2] `https://en.wikipedia.org/wiki/Category:Living_people`

[3] Example category: `https://en.wikipedia.org/wiki/Category:1955_deaths`

Another source of information is a Wikidata [4] item connected with the subject of the Wikipedia article. These pages are not directly available in this database; they were used just in an online manner to extract properties such as gender of the subject–and these extracted features are stored in a single database description file.

Putting it all together, there are:

1. Source HTML files of 1M+ Wikipedia articles (biographies)
2. Source HTML files about 600K+ Wikipedia images (from the articles)
3. The images themselves (mostly JPEGs, then PNGs and a small part of GIFs)
4. Database description file containing information extracted from the source files mentioned in 1. and 2. together with other sources
5. Webpage devoted to the database with further description and all the files available for download: (in case of any problem, to report an error, or just give feedback, you can reach the author directly at novako20@fit.cvut.cz): `http://cmp.felk.cvut.cz/~xfrancv/pages/wikipeople.html`

This whole database can be thought of as a metadatabase (or, a Wikipedia article corpus). Meaning it contains the whole of crawled Wikipedia pages and images together with information related to them. This can be used not only for the purposes of this work, but can also have broad applications in different areas such as tasks related to natural language processing. The target dataset created from this database is described in detail in the section devoted to experiments. The idea is to have a new standard dataset in the field of age and gender estimation from facial images. However, the dataset comes as a result of meeting certain criteria for the images, i.e. it is only one the many possibilities of how to create such a dataset out of the database. Thus, future work can also be about enhancing the selection method and create an ever larger (or in any other quality superior) dataset from the very same database.

Now let us go through the process of creating the database. First, we have to obtain the list of Wikipedia articles of our interest. As the target goal is to have an annotated database of people (their face images together with information about their age–at the time of when the picture was taken), we would like to find a way to get a list of all biography articles on Wikipedia. There is a web service called PetScan [5] which allows you to do just that—exporting Wikipedia Category items (pages). Our target categories consist of a category Living people complemented by single-year births and deaths categories. This export can be accompanied by additional information such as Wikidata item

[4] `https://www.wikidata.org`

[5] `https://petscan.wmflabs.org/`

connected with the subject of the respective articles. This proves to be very useful later as a source of other complementary information (e.g. demography) which makes this database unique from the perspective of richness compared to the available databases in this particular field.

The process in short consists of the following steps:

- Downloading all the source HTML files of the pages from the initial dataset (list of Wikipedia page—articles from the selected categories)
    - All of these files were saved and are made available for download in a compressed archive. These files can be used as an input dataset for any other potential application being created with the intention of having such Wikipedia articles as a source–without the need to crawl these articles again.
- Processing of additional information available in the initial dataset, such as the abovementioned Wikidata items.
    - However, these are not saved for potential future use a source files, only the extracted information is part of the database description file, so it can be used in this way. (Additional steps include making the process parallel, error-prone (recovery from errors), able to be in sync (what has already been done—successfully downloaded—when going through the step repeatedly), etc.)
- After download of the initial source files (Wikipedia articles), there follow two other steps. First, process these files and extract information leading to progress in the process. Second, follow up on these extracted pieces of information and continue crawling (this time shifting from downloading the articles to downloading images–or, precisely, this time crawling the Wikipedia //file pages about the images themselves (like an ”About” page)).
- The previous step can be seen as potentially recursively repeated (though in reality it was undergone only once) to get to a state where we have the images as image files, and extracted all necessary (or potentially useful) information information (in a form either the crawled source files, or only processed information stored in database description file, i.e. a metadata database) for further use—in experiments.

1.2 Crawling

The following steps described below in relative detail led to creation of the database.

First, download all Wikipedia articles identified by the article identifier.

Then, extract revision identifier, a number identifying revision (i.e. an edit) of the page, one can access the whole history using `&action=history` parameter in the URL; this can be subsequently used as a permalink, a permanent link to a certain Wikipedia article in a given point of time (pointing to a certain revision). As accessing the article without specifying any revision id brings you to a current, the most recent version of the article, you can obtain the (in-that-time) current revision ID without the need to access the history page (using the `action` parameter) by parsing the source HTML file and finding there this piece of information.

Now we have got tuples (article ID, revision ID) for each article ID.

From Wikidata item corresponding to the subject of the article we can determine the gender (if such property is present, more on that later).

Now abstract from having multiple (article ID, revision ID) tuples. We are presented with a single HTML file, this file contains source HTML code of a Wikipedia article. The task is to extract all images in the article. Details are in the implementation source files. Here suffice to say, that we restrict possible images to ones having resolution higher than 50x50 pixels. [6]. Another restriction for the (image) file was about the extension, `.svg` graphic content was not taken into account as an image and thus skipped. The reasoning behind this and also possible problems arising from interpreting the extension are discussed in further detail later.

Now we have a set of images in the order of appearance in the source HTML code. Image ID constructed as tuple (article ID (subject identity ID), image index). Problems arise from indexing, as different strategy could produce different indexes, or find actually different images (based on (inclusion) requirements), so it's better to have the image identified by (article ID, filename) (note that files [7] do not posses its own article ID property, thus must be identified by a different means). Notice that an image file can appear multiple times on a page or, rather, multiple times across articles. This piece of information can also be collected and used in further stages (when selecting images as inputs for a model). In this work, this kind of information was not used, but it is possible to reconstruct it from available description files, and use it afterwards.

As the previous step was about getting images from HTML, we got images in a form of links. Precisely, links to two entities: first, link to the image itself, and second, link to image file page (a general file page, not just for images,

[6]Due to a bug in the implementation, the check of resolution was performed *greedily*, meaning the result of the check came from the first dimension size of a value less than required, i.e. 40x50 excluded, 50x40 included (1st dim: 50 == 50, ok, do not check following dimensions), however, due to this bug, a superset of the intentional set was created, i.e. no image was excluded mistakenly, only (possibly) several images not conforming to the required size were also added

[7]Example *file*: `https://en.m.wikipedia.org/wiki/File:Bolt_se_aposenta_com_medalha_de_ouro_no_4_x_100_metros_1039118-19.08.2016_frz-9565.jpg`

image files might contain extra metadata such as EXIF), a page containing details about the file (image). The strategy used in this work was to skip immediate download of the linked image (the first entity) and instead of this, use the other link, download the file page as an additional source HTML and decide upon choosing the file to be downloaded based on the information from this page.

Separation of steps. We have a set of inputs, let us say for each item (= article ID) we have additional information extracted during previous steps. Anything could go wrong in any of the steps, so the of items from previous step (as an input to the current step) can be thought of as a superset of what the current step outputs (restricting on the items; the items are always getting additional information, but for some items the extraction may fail—for the reasons discussed in this chapter—) and in case of a required step, e.g. download of the target image file, the item is excluded from the output set. This is rather an illustration of how to think about the respective steps. In practice, the current step is marked with an attribute `ok` *set to false, and based on this the whole item is marked as* `ok` *false, or it gets checked during following steps to find out whether to work with this very item or not.*

Captions. But before moving to the next stage, there is still a piece of information we want to get from this Wikipedia article source HTML. The images usually have captions describing what is going on in the picture, who is there, and, for us most importantly, *when* was the image-photo taken. The result of caption extraction in this stage is tuple (target, text) for each image (as discussed above, *image* in this context means a tuple of (actual image link, image file Wikipedia page)). The meaning of a *text* is clear, it is the text content of the image caption. Several notes. `Target` tells us about specific parent in the HTML tree structure that this image is contained in. There are basically two types of an image-containing sections—it is either an `infobox` element, or a `thumb` element. Other (or unidentified) image embeddings are treated as a separate group. This information can be used in image selection process (see for example how the final dataset was made with respect to this very attribute), i.e. when selecting the images as inputs to a model.

The captions have to be again extracted from the HTML based on the position of the image in the HTML markup and its surrounding HTML tags. The naive approach would be only to check the parent tag, but as the structure can be much more irregular, a more thoughtful approach was necessary.

Redirections. The last but not least, similarly to extracting the Wikipedia article revision ID, you can get additional specification of the page from the JavaScript (JS) which is part of the source HTML. During later stages of cleaning the dataset, it is possible to get a useful piece of information from the JS attributes. But this can get really tricky – usually, when you come across an article which redirects to another one, you can see this:

50 Cent
From Wikipedia, the free encyclopedia

(Redirected from 50 cent)

But the actual redirection is done at the JS level, not at an HTTP request-response level. When you open such a URL in a browser, you get the *content* of a page where the original one ”redirects”, and the URL just get changed by the JS to point to the *target* (”redirected”) URL. So the only way how to detect such a redirection when downloading the articles programatically—i.e. only downloading the HTML source and not other linked (source) files in the HTML, and not interpreting the JS (though it is possible to do so by other means)—is to either check the subheader following the title (as seen in the example above, under the ”From Wikipedia...” text), or to check again the embedded JS which was used to extract information about revision ID. More details can be seen in the implementation source files.

Why do we actually want to avoid such articles. They can point articles which are lists, or they point to a subsection of another article, etc. Could be of use in a different setting, having global file ids, because otherwise a list article about 100 unique subjects (e.g. a list of people receiving certain honors) means 100 ids pointing to the same page extracting 100x the same N photos which are present on the page and only a single—not knowing which one—could have a meaningful label (i.e. connected to the subject who is the source of redirection). (Could do that based on `#anchor` specification in the HTML, but this is really beyond the scope of this work.

That is all the information used from the Wikipedia article source HTML. Meaning, for example, that *text* content of the page (the article itself) is not used at all. The dataset can be thus further extended and/or made more complete (in case of some information missing) by exploiting the information provided here.

For example, not all items have a corresponding Wikidata key (i.e. associated a Wikidata item about the same subject as the Wikipedia article). And even if so, the gender property might be missing. As already stated, there 1231822 items altogether. 4474 of those do not have a corresponding Wikidata item. Even when having so, the gender information (the gender property to be precise) is missing in 4488 cases. Additionally, even when having the Wikidata item and there is a gender information, in 248 cases it is not any of the values `male, female`. So we are left with 1222612 subjects annotated with gender in an expected way.

This kind of information could be added (or rather estimated) based on a text-processing model (with unspecified complexity, could be as simple classificator as `he/she` word count a the text, provided we deal with Engligh Wikipedia articles, of course). But this extra work was considered not worth the effort based on the abovementioned counts of Wikidata items (and gender property) presence.

The other implication being the sole fact of such information presence. The whole database could be used to model further text-images relationships, or not considering images at all. It is a corpus of more than a million Wikipedia

Table 1.1: Gender values in the database as extracted from Wikidata

Male	1001303
Female	221309
N/A	8962
Transgender female	163
Transgender male	38
Intersex	18
Genderqueer	18
Unknown value	5
No value	1
Gender non-conforming	1
Kathoey	1
Fa'afafine	1
Female organism	1
Male organism	1

articles (biography articles apart from outliers) and as such should be seen.

Moving to the next stage, downloading a Wikipedia file detail page, in our case we deal with image files. The procedure is pretty much the same as in the first stage when downloading the Wikipedia article pages. As inputs we have a list of links (or links created with the knowledge of certain identifiers to form the whole URL; in the previous case this was the article ID) and we shall download and save the linked content (the HTML source file) for further processing. Apart from obvious problems during crawling of the respective web pages (such as having a wrong URL—an error during extraction in previous steps—, or HTTP 5xx error codes pointing to some problems on the side of Wikipedia servers) there is also another possible source of problems. Because of (time) separation of these steps being described, it can happen (and it happened) that some of the images got deleted in the meantime. The deletion can happen for various reasons, such as using a non-free image. Such thing happened in 4564 cases for the file page (it returned HTTP error 404), plus additional 4 cases when the reality was the same (the page content provided information that such a file does not exist, though the HTTP code was 200), finally additional 50 images (as images themselves) were delete in the meantime between getting a link to the respective images and the actual download.

Loading file page source HTML and processing it. In this stage, the task is to process the downloaded Wikipedia file page source HTML and extract valuable information. There are basically two distinct types of information to be obtained. First, the actual image. There is always displayed the target image at the top of the page. Additionally, there could be links to the image in an form of "previews" in different resolutions (width and height dimension sizes). The image itself is links to the full resolution, i.e. the image in its original form. But the link is an image link (no standard text hyperlink). This is the displayed image and this particular embedded image (not the it links to), can be (and usually is) of a different size than the original one, of one of the preview sizes (and this also usually differs from the one embedded in the Wikipedia article page—this is usually smaller). One has to select which of these different resolutions to choose to download. The strategy used was to always download the image in a resolution as shown on the page. This was expected as a reasonable proxy for choosing a well-suited resolution for target database. Meaning too low for the image to contain enough visual information, but also not too high as images of high quality do not have to available in practice, so the developed model would be ill-suited compared to the actual environment in which it would be expected to be used. The later defined ”caption” dataset has an average size of a face bounding box (as detected by a face detector) equal to 150 pixels (150x150 box; median value is lower, 138 pixels).

The type of the image file was at first determined by the file extension extracted from Wikipedia file page title. Later during evaluation of results of this stage, there were found several examples of files for example with a `.pdf` extension, which definitely could not qualify to be embedded in HTML source file in an `<img>` tag, suggesting a mismatch between the originally extracted file extension and the true one. In such case, the ”true” extension is determined from the image link.[8]

This was found to be the case especially for images in TIFF format for which the previews were generated in JPEG or PNG image file formats. Also, there were several occurrences of PDF and also other not-by-default image file formats (which could also be later used to determine whether to include certain image in a database or not, as for example `.pdf.jpg` file).

In the end, we get a link to the image in the resolution as shown on the Wikipedia file page, and a link to the full resolution which is the link the image actually point to. (In case of these links are identical, it is assumed, that the preview is actually not a preview and it is the image in its full original resolution; if needed, this assumption can be easily challenged by parsing the respective Wikipedia files page source HTML files). Please note that the previews are also called *thumbs* in the database description files, as the preview files links contain a `/thumb/` in its path (in the URL).

Now let us get to the other type of information contained in the Wikipedia

[8]File extension actually does not mean anything from the point of view of the file content, i.e. changing the extension does not really alter the *content* in any way (making it a different file format) and in the very same way the extension does not have to mean anything in the URL either, i.e. there could be a standard HTML webpage on a URL ending with a `.png`, for example, but it is generally expected that such extensions extracted from the identifiers (a filename, a URL) generally conform with the actual content. (And of course, HTTP provides further options on how to specify the content—a Content-Type and/or Content-Disposition response headers.)

file page. This effectively comprise all other available information. This page is actually a detail page describing the file, so we can expect to extract valuable information for our target task—*age* estimation from facial images. A thorough inspection of this page for several different (image) files gave us the following insights:

This Wikipedia page does not necessarily have a history as known from the Wikipedia article pages. Instead, there is always a `File history` section on the page. (The previously described procedure of obtaining the link to the image is contained in an always present section `File`.)

There is a `Summary` section, containing several properties connected to the image such as `Description` or `Date`. This section, however, is not standardized in a sense that there might arise difficulties localizing it and extracting the respective information.

At the end of the page, there could also be an optional `Metadata` section, from which can be extracted additional information directly contained in the image file a form EXIF metadata. Because of the assigned task, we are especially interested in the information connected to the date and time of creation of the file, which, however, can get a bit problematic, as can be seen directly from the disclaimer opening the section: *"This file contains additional information, probably added from the digital camera or scanner used to create or digitize it. If the file has been modified from its original state, some details may not fully reflect the modified file."* The following problems may arise:

- The information about image file creation is completely missing.
- There are several EXIF metadata properties connected with such type of information, and they might me inconclusive when examined together, i.e. not all providing the same value.
- The resulting value might still be in conflict with the other information associated with original image creation time.

To conclude this stage, we had to deal with another source HTML file, extracting links to the target image itself and then extracting all the useful information provided, i.e. properties of the image—both in a form human input data (e.g. the summary section) and also the metadata obtained directly from the image file (although it is possible to also alter this data manually afterwards).

Finally downloading the images. There is nothing special about this phase as the process is the same as in the previous cases of large-scale downloading.

One last final step being running the face detector on downloaded images. In this work there is used a commercial implementation of the Waldboost face detector.

Wikidata database augmentation. Other information enhancing the richness and allowing for additional criteria for the selection process, i.e. let us select only people born outside the USA a use the trained model on examples having such property of an opposite value (test on US people only). Or, use it even more thoroughly in a cross-validation manner, such that the test set consists solely of examples from certain territory (or any other available property, occupation for example, to see if the model learned on professional sportsmen are of any use on non-sport based population) which is excluded during training and see where the performance deteriorates the most (provided there are enough examples in both train and test split).

There is a great opportunity in additional mining of information available at Wikidata with respect to the presented possibilities during the selection process. Additionally, the gathered data is now their "as is" form, as found on the linked Wikidata pages, but one can make use of the structure of the provided information, such that each property value as an item has its own properties (and their values) in the same way as any other item on Wikidata. An example of potential use: For people, there is a property "place of birth" (P19), the interesting outcome would be to separate people into geographically distinct groups. The city granularity represents a greater detail than necessary, thus further processing is needed to map a city to a region or a country (or a continent). But there is a possible caveat as the country property of the respective city can be outdated in a sense that it maps to a country which no longer exists. And again, such country can have a property "followed by" or "replaced by" (which would have to be gone through recursively) to eventually get to a current country as a group target. But even without this extended context there is still possible to do such clustering with the information provided directly (i.e. only first-level depth). The resulting dataset is described also using this information.

Chapter **2**

Method

2.1 CNN design

This work uses a convolutional neural network (a CNN) as a model. The reason for this is both general—CNNs achieve state-of-the-art results in the computer vision domain (dealing with image data), such as classification in ImageNet challenge [16], or image segmentation [17]—, and specific, with direct application in this work—CNNs are successfully used for the tasks of face recognition [18] and mainly in the state-of-the-art applications dealing with age estimation as discussed in the **??**.

The model is used to approximate the relationship between inputs and outputs in a supervised manner. The inputs being the input facial images, the outputs the respective gender and age class labels. The inputs and outputs comprise the dataset. Description of the dataset creation follows later [ref]. Now let us abstract from it and deal solely with the CNN model architecture.

There are several defining parameters of a CNN-based model. First of all, the convolutional filters (kernels) — its number and type (size, stride, padding) in each convolutional layer. There could also be applied non-standard convolutions such as a depthwise-separable convolution [19]. The CNN architecture also make use of pooling layers and additional regularization layers such as [link] batch normalization or dropout (both can be seen a method of regularization). At the end after the last convolutional layer there is a block of dense layers (fully connected layers) followed by a final (classification) layer. Each layer is followed by a non-linear activation. The specific parameters (and possibly the whole architecture is subject hyperparameter optimization and is carried out in the chapter 3.)

The last (classification) layer directly defines the way how the CNN is going to be trained as we have to map the desired outputs to the outputs of the network. This done by using a softmax activation at the end and thus perceiving the output as probability distribution among the available classes in classification setting. The respective classes point to certain (gender, age) categories. A natural choice is to have a single class for each {`male, female`} `x` {`1,2,...,100`} `(or AGE_MIN,...,AGE_MAX)` of the possibilities when using a year granularity and male/female gender values.

Another parameter from a lower-level perspective is actually the input, the dimension sizes, e.g. width and height of the input image and also the number of channels used (in case of multi-spectral imaging or just the difference between dealing with color channels or single-channel grayscale images). This has implications for the first convolutional layer and possibly for the number of neurons in the fully connected layers as explained later. Using a high-level library for implementation can abstract from such parameters (only specifying the input size) but it is important to keep in mind the implications for example in number of learnable parameters of the model.

We built a custom default architecture to be used as a baseline for performance evaluation, hyperparameter selection, and comparison to other architectures. The architecture is defined as follows 2.1:

Layer (type)	Output Shape	Param #
Input image	(100, 100, 1)	0
conv2d_1 (Conv2D)	(98, 98, 32)	320
activation_1 (Activation)	(98, 98, 32)	0
conv2d_2 (Conv2D)	(96, 96, 32)	9248
activation_2 (Activation)	(96, 96, 32)	0
max_pooling2d _1 (MaxPooling2	(48, 48, 32)	0
conv2d_3 (Conv2D)	(46, 46, 64)	18496
activation_3 (Activation)	(46, 46, 64)	0
max_pooling2d _2 (MaxPooling2	(23, 23, 64)	0
conv2d_4 (Conv2D)	(21, 21, 64)	36928
activation_4 (Activation)	(21, 21, 64)	0
max_pooling2d _3 (MaxPooling2	(10, 10, 64)	0
dropout_1 (Dropout)	(10, 10, 64)	0
conv2d_5 (Conv2D)	(8, 8, 128)	73856
activation_5 (Activation)	(8, 8, 128)	0
dropout_2 (Dropout)	(8, 8, 128)	0
conv2d_6 (Conv2D)	(5, 5, 128)	262272
activation_6 (Activation)	(5, 5, 128)	0
dropout_3 (Dropout)	(5, 5, 128)	0
conv2d_7 (Conv2D)	(1, 1, 2048)	6555648
activation_7 (Activation)	(1, 1, 2048)	0
dropout_4 (Dropout)	(1, 1, 2048)	0
global_average_pooling2d_1 (	(2048)	0
dense_1 (Dense)	(2048)	4196352
activation_8 (Activation)	(2048)	0
dropout_5 (Dropout)	(2048)	0
dense_2 (Dense)	(120)	245880
activation_9 (Activation)	(120)	0

The convolutional layers are defined as follows

1. 32 filters, size 3x3, stride 1, no padding
2. 32 filters, size 3x3, stride 1, no padding; pooling size 2x2, stride 2
3. 64 filters, size 3x3, stride 1, no padding; pooling size 2x2, stride 2
4. 64 filters, size 3x3, stride 1, no padding; pooling size 2x2, stride 2
5. 128 filters, size 3x3, stride 1, no padding
6. 128 filters, size 4x4, stride 1, no padding
7. 2048 filters, size 5x5, stride 1, no padding

Let us add several remarks to this architecture. The last two convolutional layers can be seen as fully connected as width-height dimension is reduced to 1x1 as a result of the convolutional and pooling layers. The last layer preceding those, the 5x5 convolution, can be on the other hand seen as a learnable global pooling layer as it uses the whole space (width and height dimensions) to map it to a single vector (though of variable depth—mapping from 128 to 2048).

Another possibility is to use a yet developed architecture successfully tested on a certain benchmark. This benchmark does not have to directly related to our task but we can make use of a transfer learning setting, using a pre-trained network on a possibly different computer vision task and use its learned weights as a powerful image feature extractor, using its "bottleneck features" (outputs of a certain layer of the CNN) only as inputs to our model, or fine tuning the last layers on a task we deal with. This is could also be used in a setting when there is not enough data to train the whole CNN from scratch.

Similarly to the successful application transfer learning in IMDB-WIKI setting [3] we tried VGG16 [20] as the model architecture together with the learned weights on the ImageNet challenge [16].

To make use of recent development it the area of CNN architectures, another model architecture was selected for experiments, the Xception [19] model also pre-trained on ImageNet. Xception takes inspiration from Inception [21] and generalizes the concept of separable convolutions, also using 1x1 convolutions as an embedding to a different-dimensional space. Also based on residual connections as known from ResNets [22]. Global pooling at the end—useful for bottleneck feature extraction as images of different size as inputs end up with the same-sized vector of features at the final layer; this is rather interesting from the engineering perspective.

These model architectures (VGG16 and Xception) were eventually examined in both setting, transfer learning (pre-trained on ImageNet) and learning from scratch.

Apart from the Xception model architecture, a smaller network based on the same principles was used, a SmallXception CNN. [9]. However, this architecture had to be slightly modified as for 100x100 inputs it completely eliminates the width and height dimensions. A stride of 1 is used for convolutional layers (instead of 2), and pooling with kernel size 2 (instead of 3), otherwise the architecture is used as is.

2.2 Label extraction

Let us describe the transition from a general database of Wikipedia article to a final dataset consisting of both input facial images and target output labels. Especially, the process of getting the weak annotations and transforming them

[9]`https://gist.github.com/fchollet/0f8eaa08f09e33e547431023d5955709`

into a target label, and selection process of choosing the target subjects (i.e. identities, people in our case) based on matching certain criteria.

The initial data crawling as described in the database chapter consisted of downloading *all* of the images (again, matching certain criteria such as minimal resolution) found on a Wikipedia article page. However, there was a precisely defined sequence of steps to get to the final image. The image was not downloaded directly in the form presented on the article page ("as is"), but rather first accessing its Wikipedia file page containing details about the target file together with a link (or a list of links in case of different-sized image previews in addition to the original image) to the image itself. This can be seen as on of the selection criteria as having the image stored means implicitly having all other information at disposal (collected during the steps leading to the final image download).

Now the task is to select images from the database into the dataset based on certain qualities (extracted information throughout the crawling process). We have already covered the implicit selection step. Next step is obvious as we have to deal with facial images we have to use a face detector to see if there are any faces in the image. The detector used is a commercial application of a standard Waldboost detector [23], provided as a courtesy of Eyedea [10]. This implies the (source code of the) face detector is not made available as part of this work, however, the list of eventually detected faces is published and can be used directly without the need for a separate step of running a face detection first before being able to work with this dataset. The face detection adds to the selection process the necessary condition of actually succeeding, i.e. not ending up in error. This happened for example for all `.gif` files (assumed to be images in GIF format). This way altogether 5384 out of 646134 images are lost for potential use in the dataset.

The obvious next step is to see the statistics depicting number of detected faces in the images:

We can see that certain proportion of images are not marked as face-containing. The table is divided into two subtables. First showing the overall counts for all images (successfully run through a face detector). 640750 images belonging to 396431 unique subjects. 28 % of those do not contain any face according to the detector. The detection counts are otherwise dominated by images with a single detection (i.e. a single face in the image) with a share of 80 % (of the images with at least a single face detected). Similar trait can be found in the other table restricting the images to only to those appearing first on the page (the top-most on the page, or even more precisely, the one found as first in the source HTML file). 16 % of such images do not contain any face, while 87 % of all images with a positive face detection contain precisely one.

[10]`http://www.eyedea.cz`

Table 2.1: Number of detected faces and corresponding image counts

*	640750	1.0000
0	180665	0.2820 (of all)
1	366978	0.7976 (of imgs w/ facedet)
2	51217	0.1113
3	16717	0.0363
4	8150	0.0177
5	4604	0.0100
6	2982	0.0065
7	1963	0.0043
8	1444	0.0031
9	1043	0.0023
10+	4987	0.0108

Table 2.2: Number of detected faces for *1st-images*

*	395777	1.0000
0	62695	0.1584 (of all)
1	289182	0.8682 (of imgs w/ facedet)
2	29269	0.0879
3	6890	0.0207
4	2938	0.0088
5	1464	0.0044
6	878	0.0026
7	552	0.0017
8	396	0.0012
9	261	0.0008
10+	1252	0.0038

Having that seen, we restrict ourselves to further deal with the last mentioned category—images appearing as first in the articles and having a single detection. Why such a decision: The first image on a page exhibit a feature of being placed in an `infobox` (83 % of 1st-images compared to less than 1 % in the case of second). `Infobox` is a special (HTML) structure of a Wikipedia article which can be seen as a profile summary of the subject. So it is expected to contain precisely the image we are looking for, a profile image, an image where there is a face of the target subject. (This information was extracted during the crawling after detailed analysis of possible placements of images in the structure of the page). Multiple detections add another layer of complexity into dealing with such data so we first resort to model the target relationship with a single-detection inputs. Future applications may extend the scope in both of these qualities.

Note: It may seem to not make sense the presence of 395777 ”1st-images” (and thus 395777 unique subjects respectively) and 396431 unique subjects with multiple images in a way that having multiple images implies having the 1st one. This is due to possible errors during the process as the image could have been registered on the page and before getting actually to the download the file could get deleted (or, the image was not on the page at all, just the HTML embedding, but the file could already be deleted). Thus, the actually available images for a certain subject could be the ones with indexes (2, 3) out of (1, 2, 3), for example.

The last step concerning the images is to finally get the detected face images out of the whole images. A standard set of tools was used to do so, details in [ref] Implementation. Important to note that the bounding box returned from the detector is at first increased in size by 1.25 at each side, working further with a bounding box containing a face of size 1.5x1.5 the original. Still even during this stage there could be errors occurring especially because of one way or another damaged image files.

Note: The checks during the download process are not enough *per se*. As such process is rather about getting the actual data (bytes) and marking it as OK when all the data get downloaded and saved without an error. But this does not really tell us anything about the *quality* / nature of the data. In short, image files may be corrupted in a way that the face extraction ends up in an error.

Note: Unfortunately, although the tools used come from a standard package (Python port of OpenCV library), there have been several problems detected afterwards. The OpenCV opens and reads the file even when it is damaged in a way that there are missing pixels in the image—though such error should have resorted into an error during download if after a manual check the file appears to be OK on the web—, other libraries may fail to open and read the image at all, and most importantly, we do not want such images as part of the dataset (or at least know about such nature of the data, it was otherwise a quiet error the whole time). However, this is only the case for a total 295 out of 646134 images. The affected files are reported in the errata section of the database web *wikipeople* [11].

This was the selection process concerning the images as files and image data they contain. Now let us focus on the selection from the perspective of available metadata, the extracted information. Such information has already been used in part as part of the decision picking the images appearing first on the page.

The selection process continues as follows. First, we want to make sure we actually deal with *articles*. One can get such information from the initial embedded JS code from the Wikipedia article source HTML file (when trying to avoid additional requests and work only with the already crawled pages). There is a flag variable stating whether it is actually an article (and not a Category page for example). A similar variable contains information about

[11] `http://cmp.felk.cvut.cz/~xfrancv/pages/wikipeople.html`

page redirection. This could be both helpful or harmful. The redirection helps with case sensitivity (the titles are otherwise case-sensitive) or in a case actually redirecting to (a possibly subsection of) another Wikipedia article. Such article may be in a form of a comprehensive list of people. The implications of such redirection are obvious and is already discussed 1.2. The problem is that tracking such a redirection is not straightforward just from the source HTML files (and saved HTTP status codes during crawling)—this is because Wikipedia does not redirect at the HTTP level, but rather *displays* the content of the redirection target get, displays information about a ”redirection”, and changes the URL to the target one (through JS). However, it is still possible to find out whether there was a redirection or not, so we use this information with the intention to exclude such subjects (with a redirected article page) from further consideration about including it to the dataset. In short, article is excluded when is marked in the metadata database as ”not OK” (revision ID extraction was unsuccessful—as a result of an internal Wikipedia error), or is not an article, or is a redirect. 2298 subjects are excluded as a result of applying this rule (out of 396628).

But there two more crucial requirements on the subject. Target label consists of a tuple (gender, age) for each facial image. Thus, we have to know the gender and the birth year (assuming we get to information about when the respective images were taken—to be able to compute the age part of the label). There is a possibility to model such features based on article text for example, or other extracted information. But date of birth (the year) comes for free for a large part of the subjects in the database thanks to ”births” categories which were at the very beginning of the database creation. These are the birth categories for the respective years ranging from 1900 up until 2017. Even in case this was not enough, there is the other source of information, the Wikidata, with such property present in most cases (for most subjects), to get both the gender and the year of birth. For 23729 of 396628 subjects the year of birth is not available, for 3185 this is the case for gender, altogether 23837 subjects get excluded in this step.

Now let us finally move to the crucial part—extracting the image annotations—the weak labels–to be able to assign an age label to a facial image. The other part of the label—the gender—is extracted from metadata (originally extracted from Wikidata item related to the subject) and is considered as ground truth without further modelling.

There are basically two direct types of sources of annotation with respect to the target age label for the images. The indirect ones consist again of exploiting all other available information such as the article text. First, there is a place in the HTML structure of the source file of the Wikipedia article where the image actually is. This place (i.e. the HTML element) can get us to the image caption (if there is such) simply navigating up through the HTML tree. Or, actually, not that simply, as one has to pay attention to special tags as the element contains the description does not have to be a direct parent element of the element containing the image. The implication is that stopping too early may not yield the caption text (even if there is on), on the other hand stopping too late may yield a whole paragraph (or text in any other element), i.e. a text where the caption being just a part of it. Having that done in a thoughtful fashion, we end up with a caption text. This is to be processed later together with other extracted features which are described below.

Second, there is the Wikipedia file page related to the image file in question. One has to identify possible useful features for the task from the information provided on the page. Several of these were identified and are described below. See an example to get a gist of where these pieces of information come from.

- a filename, placed at the top
- ”Summary” fields description and date
- ”File history” fields date and comment
- ”Metadata” EXIF fields
  - (date of creation, date of digitizing, date of modification; in Wikipedia HTML element classes: `exif-datetimeoriginal`, `exif-datetimedigitized`, `exif-datetime`)

Let us go through an example:
`https://en.wikipedia.org/wiki/File:Edsger_Wybe_Dijkstra.jpg`
Extracted values for the example file page:

- Filename: *”File:Edsger Wybe Dijkstra.jpg”*
- Summary description: *”English: Portrait of en:Edsger W. Dijkstra, one of the greatest mathematicans in history of modern mathematics. [...]”*
- Summary date: *”1 Aug 2002 (as metadata)”* [12]
- File history date: *”05:05, 12 June 2008” (using the bottom-most record, the oldest one)*
- File history comment: *”{{Information —Description={{en—1=Portrait of en:Edsger W. Dijkstra, one of the greatest mathematicans in history of modern mathematics.}}”* [13]

[12]The summary date field might contain the specified date not only as a text but also as an HTML element attribute, being it so, one does not have to parse the date from the text, but get it directly in an always-the-same format.

[13]The file history comment field was found to sometimes contain the initial input to the page (in Wikitext syntax). The idea is that it might contain for example information aimed for summary category—this information is preserved here (if available, i.e. if the comment field really contains this kind of information) and it could be changed in the summary (as in any other part of the page), possibly deleted. This field is part of the extracted features rather as a last resort option if there is not any other annotation.

- Metadata exif created: *"23:10, 8 January 2002"*
- Metadata exif digitized: *"23:10, 8 January 2002"*
- Metadata exif modified: *"00:58, 27 April 2015"*

Now we have several text features possibly containing a year which we would be able to connect with the year of birth to get an age label. We make a crucial assumption that the extracted year from such a text is actually telling the year when the image was taken. We further make a statistical evaluation of this assumption and also test this experimentally. Altogether there are 9 distinct text features: caption, filename, summary description, summary date, file history date, file history comment, and three EXIF metadata fields. Definitely not all of the features are actually expected to fulfill the above-mentioned assumption. For example the file history date is one hand always present (and in an always-the-same format), it is an analogy of Wikipedia article history, but on the other hand one cannot really expect this to be the true year annotation (by this we mean the year when the image was taken, as discussed above; or even more precisely, when the scene depicted in the image took place, i.e. it could be a scan a photograph of an older printed photo, etc. so we are interested in the true "original" year) of the respective image. The meaning is clear, it is the date the image was uploaded to Wikipedia, but does not have to be the true year. However, such information is still useful as it can be used to extract other semantic meaning, such as the upper bound on true year.

Considering such meanings of the respective features we advance to a step where information gets combined and as a result we get 5 individual year annotations. As this step is crucial for the whole work, it can be seen in full detail in implementation source files.

This heuristic approach make use of lower and upper bounds. These are initialized as follows:

lower_bound = year of birth (always present, otherwise excluded from consideration)
upper_bound = year of death (if present, otherwise 2017 as this is the year when the images were crawled)

The protocol for year extraction from text features is the same across the features. It utilizes regular expressions to extracted the year.

Meaning there have to be 4 digits preceded and followed by non-digit character. This regular expression also matches a spurious year such as in text "0123", but these matches cannot qualify as the year annotation for the feature due to the use of lower and upper bounds (both initialized to meaningful values).

The following abbreviations are used for the respective text features:

`name` ... filename
`desc` ... summary description
`date` ... summary date
`time` ... summary date in a form of an HTML attribute
`hist` ... file history date
`init` ... file history comment
`exif_orig` ... exif metadata created
`exif_difi` ... exif metadata digitized
`exif_modi` ... exif metadata modified
`caption` ... image caption (extracted from the Wikipedia article)

When extracting a year annotation from the text features there is always the protocol in place:
For each match (using the regular expression), evaluate the match as a number, and if it is within the bounds (greater than the lower bound, as we do not expect any people of age 0, and less or equal than the upper bound) [14] then accept it, otherwise continue with another match; if no suitable year found and there is no further match, return None, a signal that there is no suitable year.

As mentioned above, at first there is the `hist` feature which is used solely to update the upper bound (i.e. if following the protocol results in a year annotation—implicitly meaning adhering to the bounds, thus lower or equal to the upper bound—, this annotation is used to update the upper bound).

Next follow all the `exif_*` features, all of them are gone through the protocol, the minimum (if any of the features resulted in a year annotation) is used to create a new year annotation EXIF. The EXIF is then used to once again update the upper bound.

Now having the upper bound at its final value, let us move to the other features to form new year annotations. The respective features are always gone through the protocol get a year (or nothing).

DATE annotation is created from `time` preferred to `date` (if there is a year as result, then stop and use this value).
DESC annotation from `desc` preferred to `init`
NAME annotation solely from `name`
CAPTION from `caption`

[14]There is actually one additional check. To accept a year equal to the upper bound in case the upper bound is equal to the year of death, the *previous* match is not allowed to be the year of birth as this is assumed to be displaying obviously the birth and death years and not the true year in case of the latter one.

The result this time is 5 individual year annotations, CAPTION, NAME, DESC, DATE, and EXIF. First, we exclude examples not having any year annotation, i.e. the process of getting a year out of a text did not result in any year for any of the target 5 annotations.

Now one the possibilities is to use one of these year annotations directly. When going manually through the Wikipedia articles, the caption text seemed as having the true annotation in almost all cases so it would be a natural candidate for selection. The previous statement about the true annotation is also later evaluated statistically.

The use of caption as a sole annotation eventually raises the question about the respective annotations distribution. The caption annotation may not be present and we could potentially lose a lot of examples. Statistics about co-occurrence of all the possible binary combinations follow:

The counts are based on the set of so-far selected images, in total 217862 of them, with age in range [16,76). [15] We can see that by restricting to the caption annotation only, we would lose more than half of the samples, thus a way to combine the other year annotations has to be devised.

After detailed statistical analysis of the respective annotations we ended up combining them using a simple median selection. This is because the other annotations (other than the caption) proved to be of similar qualities as the caption. This is also documented later.

2.3 Dataset statistics

In total, three distinct datasets have been created, called "caption", "nocaption", and "anycaption", as a result of the described selection process. All datasets are comprised of an age range [16,76). These datasets are available for download at the database webpage. [16]

The "caption" dataset.

It consists of (image, gender, age) items where the caption year annotation is available, and this is used as a ground truth label in the (gender, age) label tuple. In total, there are 99071 samples. This dataset is used to create a validation set and a test set identical across datasets. 30000 samples are randomly selected from all of the classes (distinct labels) using stratified sampling so that the class distribution is preserved as is in the initial population. These 30K are then randomly split into two halves of 15K (again, preserving the distribution) to form a validation set and a test set. The training set consists of the rest, the remaining 66071 samples.

[15] Speaking about implies that the age label had to constructed somehow, i.e. combining those 5 year annotations into a single one, as there are also samples not having the caption year annotation which was initially selected as a single annotation mapped to the age label. This is done by selecting a median of all the annotations as is also discussed further.

[16] `http://cmp.felk.cvut.cz/~xfrancv/pages/wikipeople.html`

Table 2.3: Counts of respective annotation combinations; Exclusive counts: 1 means present, 0 means not present

CAPTION	DATE	DESC	NAME	EXIF	#
0	1	0	0	0	25218
0	1	1	0	0	24736
0	1	0	0	1	19054
1	1	1	1	1	16187
0	1	1	0	1	15458
1	1	1	0	1	14925
1	1	1	0	0	13796
1	1	1	1	0	13351
0	0	0	0	1	8317
1	1	0	0	0	7680
0	0	1	0	0	7647
1	1	0	0	1	7421
0	1	1	1	1	6554
0	1	1	1	0	5910
1	0	0	0	0	4694
1	0	1	0	0	4037
1	1	0	1	0	3988
1	1	0	1	1	3344
0	1	0	1	1	2839
1	0	1	1	0	2464
0	1	0	1	0	2268
1	0	0	1	0	1435
1	0	0	0	1	1119
0	0	1	0	1	1099
0	0	0	1	0	1018
0	0	1	1	0	985
1	0	1	0	1	770
1	0	1	1	1	549
1	0	0	1	1	345
0	0	1	1	1	344
0	0	0	1	1	310
0	0	0	0	0	0

Table 2.4: Counts of respective annotation combinations;
Non-exclusive counts: 1 means present, 0 means any (present or not)

CAPTION	DATE	DESC	NAME	EXIF	#
0	0	0	0	0	217862
0	0	0	0	1	98635
0	0	0	1	0	61891
0	0	0	1	1	30472
0	0	1	0	0	128812
0	0	1	0	1	55886
0	0	1	1	0	46344
0	0	1	1	1	23634
0	1	0	0	0	182729
0	1	0	0	1	85782
0	1	0	1	0	54441
0	1	0	1	1	28924
0	1	1	0	0	110917
0	1	1	0	1	53124
0	1	1	1	0	42002
0	1	1	1	1	22741
1	0	0	0	0	96105
1	0	0	0	1	44660
1	0	0	1	0	41663
1	0	0	1	1	20425
1	0	1	0	0	66079
1	0	1	0	1	32431
1	0	1	1	0	32551
1	0	1	1	1	16736
1	1	0	0	0	80692
1	1	0	0	1	41877
1	1	0	1	0	36870
1	1	0	1	1	19531
1	1	1	0	0	58259
1	1	1	0	1	31112
1	1	1	1	0	29538
1	1	1	1	1	16187

In total, there are 66071 samples, 50657 males and 15414 females (0.7667 / 0.2333). Minimal age 16, maximal age 75. (30K + 70238 when including all age categories; 758 samples (0.76 %) of age < 16; 3409 samples (3.40 %) of age > 75 (including 33 samples of age > 99))

Validation set consists of 15000 samples, 11503 males and 3497 females (0.7669 / 0.2331).

Test set consists of 15000 samples, 11500 males and 3500 females (0.7667 / 0.2333).

The ”nocaption” dataset.

These are the samples which miss the caption year annotation (and contain at least one other). There are 121757 samples, 100953 males and 20804 females (0.8291 / 0.1709). This whole sample is used as a training set as validation and test set are held fixed across the datasets. The age label is created from all available year annotations (caption, date, desc, name, exif) using a median value. (128844 when including all age categories; 993 samples (0.77 %) of age < 16; 6094 samples (4.73 %) of age > 75 (including 53 samples of age > 99))

The ”anycaption” dataset.

These samples contain both samples for which there is a caption year annotation, and those where there is not. There are 187800 samples, 151580 males and 36220 females (0.8071 / 0.1929). This count does not correspond to the sum of the previous datasets; this because of the method of getting the age label. A median of all available year annotations is used (as in ”nocaption” dataset) and as a result of this, 62 samples from ”caption” training set no longer conform with the age bound 16 to 75 (inclusive). Similarly, 34 new samples now fall into the dataset (having the caption year annotation which differs from the median value of all annotations and now is within the bounds). This change does not alter the labels previously from the ”caption” dataset in any significant way as the median value is equal to the one from caption annotation in 98.12 % of samples (and is within a 5-year difference in 99.60 % of samples).

Please see the enclosed SD card or refer to the website for distribution plots of the respective datasets.

Additionally, we have also demographic information available as part of the database.

For example, see the most represented countries and branches of occupation:

2.4 Statistical evaluation

Please see the enclosed SD card for exhaustive evaluation protocol. There have been developed separate tools just to assess the quality of the annotations.

*	199752	of total 217800 (0.9171)
1	61498	United States of America
2	16900	United Kingdom
3	8700	Germany
4	7973	France
5	6883	Canada
6	6447	Australia
7	4749	Italy
8	4314	Sweden
9	4183	Kingdom of the Netherlands
10	3747	India

Table 2.5: Distribution of ”anycaption” dataset w.r.t. country of citizenship

*	199409	of total 217800 (0.9156)
1	27817	politician
2	21277	association football player
3	15662	actor
4	7331	singer
5	4925	baseball player
6	4722	writer
7	4219	journalist
8	3517	painter
9	3259	American football player
10	3054	ice hockey player

Table 2.6: Distribution of ”anycaption” dataset w.r.t. occupation

Chapter **3**

Experiments

3.1 Model parameters

There are other non-CNN specific parameters which include:

- The loss function used, plus possible other additional functions used to assess the performance of a model.
- The choice of the optimizer, i.e. in how is the learning process specified (how is the loss function minimized).
- Other NN-specific parameters, as the number of neurons in the top layers (after the convolutional ones), or whether there are any.
- From the perspective of research reproducibility, all other parameters needed to re-instantiate the whole experiment. This includes for example discussing initialization of layers at the very beginning, or specifying what an epoch means online training (when the inputs get prepared online, when training is in progress, usually due to image augmentation or because the size of dataset is large than what fits into memory).

Let us first go through this further specification.

The default loss function is a cross-entropy. [17] This is based on the assumption we deal with probability distributions as the outputs. So it is tightly connected to the semantic meaning of the output. As we use age range 16–75 in our experiments, we further make this a classification problem of assigning an input to an output category with the meaning of target age (with a year granularity, so we have 60 age categories). This is further divided into categories distinct for males and females, so together 120 classes and the

[17]"Sparse categorical cross-entropy" as called in the Keras library, or "Softmax cross-entropy" as called in the TensorFlow computational framework. These two software packages form a basis of our experimental work.

output can be then seen as an (age, gender) joint distribution estimate. One can then easily get marginal distributions and further process the result to get to a point estimate (the latter in case of age distribution; such method in our case consists of taking the median value, i.e. the 50th percentile).

Back to the loss function. The label when training is converted to one of the available classes, i.e. assigning a probability 1. The value of a loss function for the respective input then restricts just to to the $-\log label\_class\_prediction$

Note: The activation in the last layer is thus a softmax function to get values which can be thought of as a probability distribution. Additionally, the fact that predictions for other classes do not directly contribute to the loss is OK from the perspective of learning. Because of such an activation function, the result actually depends on all the other values forming the softmax basis (the values for other classes before activation). This way it is connected, and during training, during the optimization step, the gradients get distributed to those neurons as well.

The optimization method used is an SGD (stochastic gradient descent) with momentum. There are also other more advanced ways how to deal with momentum (such as Nesterov momentum). Or use adaptive learning rates (AdaGrad, RMSprop, Adam) but these might suffer from generalization problems [24]. We stick with this default selection.
decay = 0.0005
momentum = 0.9
learning rate is subject to further hyperparameter optimization, batch size 100 (if not stated otherwise)

The fully connected layers. We resort to having the top layers in its configuration as seen in model architecture section. The results of experiments are provided for the respective configurations. This was especially about the VGG16 network architecture, as there could be basically 3 different top layers. The one originally used in the VGG16 network; the one used by our custom default model; no dense top layer (apart from the classification) as used in the Xception model architecture.

To make this list complete, we further add that all the layers (all the respective neurons) (if trainable, i.e. not in the pooling layer for example) get initialized (if not the case of transfer learning) as follow:
weights ... Glorot uniform initialization [18] (using a multiple of 6 of number of inputs and outputs, as in [25]) biases ... zeros

In case of the epoch, it generally means going through all of the inputs exactly once. In case of online training where the images are augmented, for example adding a horizontal flip for each of the input, the dataset now actually consist of twice the number of original inputs. However, the epoch

[18] `https://github.com/keras-team/keras/blob/895dba6c/keras/initializers.py#L338-L355`

in that case is still the number of original inputs. The reason for this is being able to measure effect of such online augmentation because otherwise having the dataset N-times larger (for N resulting inputs from a single original one) would also have implications for training from the perspective of using a non-annealed learning rate for a longer period time , for example (for a larger number of steps of the optimizer, having other parameters fixed) when there is a non-fixed learning rate used (and the changes depend on the epoch number).

Training samples get shuffled at the beginning of each epoch to get random batches during learning phase. In case of augmented images, the original ones get shuffled (as representations of all the augmented images) and when the respective image (or rather its identifier) is in the batch, a randomly chosen augmentation of the listed ones is selected, applied, and the result used as a training sample. Such augmentation is then removed from the list of possible ones for the respective image (until all of them have been used during subsequent epochs, then start again).

3.2 Performance evaluation

At first, we conducted a set of experiments devoted to hyperparameter optimization using. For this, we used the default model, ”caption” dataset, and grayscale facial images of size 100x100. The optimized hyperparameter were these respective ones:

- Method of normalization
- Use of virtual examples
- Use of a dropout layer
- Learning rate policy

Now let us describe the possible variants.

Normalization. Either normalization of [0-255] (8-bit input values) to [-1,1] (linear scaling) (varaint A), or to [0,1] with subsequent subtraction of a mean image computed across all training input examples (variant B).

Virtual examples. Using augmented images created from the original facial image as found by the face detector (which outputs detections as square boundind boxes which could be rotated according to the face position). Altogether 10 resulting images using different zoom, horizontal flipping, and rotating. A single resulting image as a column in the following table:

Dropout layer. Trying different values of dropout for specified dropout layers, or in case of value 0.0, replace by a batch normalization layer instead.

Learning rate policy. This comprises using a constant learning rate throughout the whole training process. Alternatively, use gradual learning rate decay

```
[1, 1,  1, 0.95, 1.05, 1,  1,  1, 0.95, 1.05] # zoom (1x)
[0, 0,  0, 0,    0,       1,  1,  1, 1,    1 ]   # mirror
↪  (T/F)
[0, 5, -5, 0,    0,       0,  5, -5, 0,    0 ]   # rotation
↪  (degrees)
```

Listing 1: Virtual samples specification

from an initial value to a final value with the steps scaled logarithmically or linearly, for a total of selected number of steps (epochs).

Now to select the hyperparameters we have to first define a protocol how to evaluate results. As we always have the dataset split into training, validation, and testing samples, the procedure is obvious, but nonetheless mentioned for nonambiguity. The learning phase uses only the training samples, performance at each epoch is evaluated on validation samples. State of the model at epoch giving the best performance is then evaluated on testing samples and these results (testing performance) are then used as a result of the model (and so also when comparing different models).

Last to add, how the performance is actually assessed. There are several metrics. [19] MAE, CS5, and accuracy. MAE stands for mean absolute error, it is used to evaluate age predictions using absolute deviations from the true age. CS{N} gives a proportion of age predictions of up to (and including) N years of deviation. Accuracy of classification is used to evaluate gender predictions.

Applying the Hoeffding's inequality, we get to a 0.65 bound to have 95 % confidence of the *real* MAE being in the interval of resulting MAE 0.65 (years) interval.

Based on the experiments we selected the following configuration.

Normalization variant A, variant B has not proved to be superior. Similarly, not using the virtual examples.[20] Other parameters as follows: batch size 100, use of a dropout, dropout value 0.5, linear learning rate decay for 50 epochs going from 0.01 to 0.00001.

This configuration yields the following baseline results:
MAE 7.448
CS5 0.488

[19]Not a metric in a standard mathematical definition, rather as a general evaluation criterion. To choose the epoch giving the best performance, we look for minimal MAE on the validation samples.

[20]Although using such samples is a common practice, in our setting it could be due to the in-the-wild nature of the dataset but also not having conducted enough experiments to state clearly that this does not yield better results, as the online training was by an order of magnitude slower if compared to using a pre-computed dataset, i.e. in this default model an epoch taking an hour instead of a few minutes.

GER 0.066 (gender prediction error)

Now the crucial question is what happens when the dataset is replaced by ”nocaption” and later ”anycaption”. Does the process of getting an age label from weak annotations work in a sense that we have a dataset usable for learning the mapping between facial images and age yielding comparable results as the above?

”nocaption” dataset
MAE 7.242
CS5 0.497
GER 0.064

”anycaption” dataset
MAE 6.942
CS5 0.517
GER 0.056

And the answer is yes. The ”anycaption” results being a bit better which can explained a few ways. First, randomness in training (initialization of layers, specific sequence of batches). This effect should not be large as we have evidence from training the on the ”anycaption” dataset twice (and the error development can also be seen from the respective values after each epoch) and the resulting difference in best MAE of magnitude 0.03 (knowing this we also have a cue about the number of meaningful decimal places); however this effect is also to be kept in mind. Second, the age labels might be really of superior quality compared to the ”caption” ones. Third, the effect comes as a result of a larger dataset (approximately twice the ”caption” size). Anyway, the net effect is important. Finally, the ”anycaption” dataset gives the best results.

Further, there were additional experiments conducted using this default model architecture on the ”anycaption” dataset. ”Soft” learning and using class weights for samples. Soft learning uses Kullback-Leibler divergence as a loss function and the ground truth class distribution is not created as probability 1.0 for the target class but rather as sharply peaked distribution at the true class but with nonzero probability assigned to neighboring classes (meaning neighboring age, not the other gender—class belonging to the other gender stay at zero probability). Important to note that this really has to be a distribution, i.e. summing to 1, because then the Gibbs' inequality holds (KL divergence always ¿= 0, and == 0 if we deal with identical distributions), otherwise the loss can be negative and not having an optimal zero loss, i.e. the optimization steps may not lead to actually getting closer to the target distribution.

Soft learning scheme used: exponentially (base 4) decreased (from 1) based on distance from the true class to up to 3 years of age deviation, then normalized to 1 to get a distribution.

Soft learning
MAE 7.041
CS5 0.516
GER 0.060

Result: no evidence for this to be a superior way of training compared to the original one.

As the gender distribution is imbalanced, having only approximately 20 % females, and also because we have lower number of samples at the age-tails (young and old people), the idea is to weight classes so that the underrepresented ones have higher weight.

Weighting scheme used: 1 + log-scaled inverted proportion of particular class samples. Minimal possible weight is 1 in case all of the samples coming from a single class (which in practice does not make much sense for training on a classification task with multiple classes; this just to illustrate).

Class weights
MAE 6.606
CS5 0.536
GER 0.046

Yes, this way we can get better performance, gender error is decreased by 1 percentage point. This is actually the best MAE, CS5, and gender error attained by this model architecture (and using 100x100 grayscale images).

For this model we further provide benchmark results. The displayed results always follow the same pattern: first, there is evaluation on training, validation, and testing data. Then the benchmarks follow: Wikipeople test set (as defined in this work—you can see the same results for the test split if trained on Wikipeople dataset), IMDB-WIKI [3], ChaLearn [4], APPA-REAL [6], Morph [2], and FG-NET. [1]

Now using a color. Additionally, there are initial layers at the beginning learning the colorspace, as the best option found in [26]. Interesting to note that in this case, dropout was replaced by batch normalization, otherwise the performance was as bad as MAE equal to 12.

However, all of this beaten by the SmallXception model, trained only on the "caption" dataset, not using color. Original Xception architecture suffered from overfitting and was not further finetuned further as the small version is of a competitive size to our default model and thus is more suitable to find out if there are gains coming from the model architecture, the use of residual connections etc. And apparently there are.

Table 3.1: Performance when not using class weights

	TRN	VAL	TST	WIKI	IMDB	CHAL	APPA	MRPH	FGNT
MAE	7.075	7.010	6.942	6.942	7.392	7.758	7.950	6.014	8.622
CS5	0.518	0.511	0.517	0.517	0.474	0.469	0.437	0.507	0.335
CS10	0.783	0.786	0.782	0.782	0.758	0.748	0.729	0.863	0.689
G err	0.043	0.056	0.056	0.056	0.057	0.072	–	0.070	0.109
G (F)	0.149	0.161	0.164	0.164	0.089	0.095	–	0.350	0.202
G (M)	0.017	0.024	0.023	0.023	0.031	0.049	–	0.019	0.023

Table 3.2: Performance when using log-scaled class weights

	TRN	VAL	TST	WIKI	IMDB	CHAL	APPA	MRPH	FGNT
MAE	6.603	6.696	6.606	6.606	6.995	7.384	7.398	5.833	8.135
CS5	0.542	0.537	0.536	0.536	0.502	0.496	0.465	0.543	0.397
CS10	0.806	0.802	0.806	0.806	0.781	0.762	0.761	0.858	0.739
G err	0.031	0.047	0.046	0.046	0.045	0.069	–	0.069	0.095
G (F)	0.071	0.096	0.100	0.100	0.053	0.072	–	0.205	0.158
G (M)	0.021	0.032	0.030	0.030	0.039	0.066	–	0.045	0.037

Table 3.3: Adding a color and learning the colorspace

	TRN	VAL	TST	WIKI	IMDB	CHAL	APPA	MRPH	FGNT
MAE	–	6.286	6.257	6.257	6.850	6.431	6.758	5.954	6.770
CS5	–	0.567	0.565	0.565	0.514	0.554	0.516	0.539	0.501
CS10	–	0.824	0.825	0.825	0.789	0.815	0.794	0.849	0.824
G err	–	0.044	0.042	0.042	0.049	0.066	–	0.056	0.145
G (F)	–	0.126	0.120	0.120	0.083	0.094	–	0.292	0.281
G (M)	–	0.019	0.018	0.018	0.020	0.038	–	0.013	0.018

Table 3.4: Using SmallXception model instead of the default one

	TRN	VAL	TST	WIKI	IMDB	CHAL	APPA	MRPH	FGNT
MAE	–	6.472	6.408	6.408	6.734	6.399	6.819	5.705	6.903
CS5	–	0.551	0.555	0.555	0.521	0.549	0.503	0.563	0.461
CS10	–	0.815	0.815	0.815	0.796	0.818	0.795	0.865	0.846
G err	–	0.049	0.046	0.046	0.046	0.073	–	0.059	0.105
G (F)	–	0.130	0.126	0.126	0.072	0.106	–	0.192	0.187
G (M)	–	0.024	0.022	0.022	0.026	0.043	–	0.034	0.028

Table 3.5: Performance of VGG16 model in transfer learning setting

	TRN	VAL	TST	WIKI	IMDB	CHAL	APPA	MRPH	FGNT
MAE	5.384	5.645	5.592	5.592	5.916	5.390	5.869	5.013	6.162
CS5	0.643	0.607	0.615	0.615	0.577	0.621	0.560	0.618	0.511
CS10	0.874	0.862	0.862	0.862	0.846	0.879	0.852	0.914	0.862
G err	0.019	0.033	0.033	0.033	0.026	0.041	–	0.037	0.086
G (F)	0.063	0.087	0.091	0.091	0.034	0.052	–	0.166	0.163
G (M)	0.008	0.016	0.015	0.015	0.020	0.030	–	0.013	0.014

Further experiments using the colorspace were conducted in the transfer learning setting. Many different approaches were examined. There is a possibility to train such models in a sequence of steps. Having first the weights of the original model, the top (fully connected) layers are dropped and replaced by a custom top. This can be built the same way original model is, having the same number of dense layers and differing only in the very last layer as the number of neurons there depends on the number of output classes. Another option is to have *really* a custom top layer, not the one which is used in the original model. Finally, the last option is to have there directly the classification layer.

Without further ado, based on the results of the experiments, we resorted to keep the structure of the dense layers as in the original architecture, and to train in a transfer learning setting without first training the dense layers only. The idea is otherwise to train only the top, having convolutional blocks kept fixed, so that if there is intention of further finetuning the convolutional layers, these do not get destroyed by high gradients when the classification is completely off the target at the very beginning. But we were not able to get reasonable results by just training the top. This implies both that sticking to such training protocol only adds to complexity (as it has to be done in several distinct steps) but still there will be the problem of possibly destroying the learned features in the convolutional blocks. The final setting is thus rather using the transfer learning for lower-level feature extraction than to use it as a complex-feature extractor. Still, this provides better results than training the net from scratch (by a margin of 1.0 in MAE), so the transfer learning proved to be useful even when used in this non-standard way.

VGG16, transfer learning, using ImageNet weights, setting last 4 convolutional blocks and the top layers as trainable, using 128x128x3 inputs (color images), linearly decayed learning rate for 20 epochs going from 0.001 to 0.00001, dataset ”anycaption”.

Training only on ”caption” dataset, learning rate decayed for 50 epochs from 0.01 to 0.0001. Overfitting (best validation epoch number is 6) but overally the best results (when looking at MAE; the gender error is higher).

Last but not least, we have to provide results attained by training on a

Table 3.6: Best performing model overall in MAE on Wikipeople dataset

	TRN	VAL	TST	WIKI	IMDB	CHAL	APPA	MRPH	FGNT
MAE	4.171	5.585	5.519	5.519	5.973	5.085	5.637	4.785	5.895
CS5	0.739	0.616	0.623	0.623	0.575	0.660	0.589	0.653	0.546
CS10	0.939	0.863	0.867	0.867	0.840	0.890	0.863	0.919	0.869
G err	0.026	0.044	0.045	0.045	0.047	0.062	–	0.054	0.202
G (F)	0.107	0.172	0.174	0.174	0.095	0.114	–	0.332	0.414
G (M)	0.002	0.006	0.006	0.006	0.007	0.013	–	0.003	0.005

Table 3.7: VGG16 model trained on IMDB-WIKI dataset, using images of size 100x100x1

	TRN	VAL	TST	WIKI	IMDB	CHAL	APPA	MRPH	FGNT
MAE	6.500	6.739	6.547	8.628	6.547	7.213	7.875	6.539	8.634
CS5	0.526	0.507	0.514	0.383	0.514	0.478	0.420	0.473	0.347
CS10	0.810	0.799	0.810	0.671	0.810	0.767	0.719	0.803	0.694
G err	0.031	0.033	0.032	0.072	0.032	0.077	–	0.087	0.138
G (F)	0.048	0.045	0.047	0.228	0.047	0.123	–	0.415	0.286
G (M)	0.019	0.023	0.020	0.025	0.020	0.033	–	0.027	0.000

Table 3.8: VGG16 model trained on IMDB-WIKI dataset, using images of size 128x128x3

	TRN	VAL	TST	WIKI	IMDB	CHAL	APPA	MRPH	FGNT
MAE	4.519	5.389	5.198	6.750	5.198	4.840	5.706	5.536	5.577
CS5	0.700	0.618	0.636	0.511	0.636	0.658	0.577	0.561	0.568
CS10	0.919	0.881	0.889	0.797	0.889	0.912	0.854	0.875	0.893
G err	0.017	0.022	0.019	0.055	0.019	0.044	–	0.046	0.152
G (F)	0.023	0.025	0.025	0.180	0.025	0.073	–	0.254	0.310
G (M)	0.012	0.020	0.015	0.016	0.015	0.017	–	0.008	0.005

contemporary standard dataset. We have chosen IMDB dataset as it is of similar size, so the hyperparameters are expected to not be completely off. (As one can see batch size and learning rate connected, keeping one fixed and optimizing the other, an analogy of this being consideration of the dataset size as already discussed when dealing with generation of augmented images.)

3.3 Model visualizations

Please see the enclosed SD card graphical visualizations of model performance and individual predictions on selected samples revealing the reasons for erroneous predictions.

Conclusion

The major contribution of the work is the creation of a new database. There are not only database for the purposes of age estimation research but there is also a general dataset containing biographies of a large number of people, a Wikipedia article corpus let us say which can be subsequently used as a starting point for creation of dataset of completely different qualities thanks to both the information richness and also thanks to description of the process of how to get to meaningful data out of the dataset. Speaking directly about the created datasets, the final Wikipeople dataset "anycaption" is of size comparable to the otherwise largest available dataset. In addition, apart from detailed description of the steps leading to the eventual form (which is of great importance from the perspective of research reproducibility and also to make it easy to find potential errors in the work), the dataset has been also evaluated both statistically based on a manual check of several features with reported results and also in the target application, learning the age and gender representation using convolutional neural networks. The reported results may serve as a baseline for what can be achieved. An important part of the experiments was comparing it to results obtained by training on a yet standard dataset. Seeing comparable results on the rest of the benchmarks and a significantly higher error on the Wikipeople data suggests the distribution being the most challenging one (if not erroneous, which was statistically evaluated as not being the case) so in this sense ideal for further progress in the field.

Bibliography

[1] Panis, G.; Lanitis, A.; et al. Overview of research on facial ageing using the FG-NET ageing database. *IET Biometrics*, volume 5, no. 2, 2016: pp. 37–46.

[2] Ricanek, K.; Tesafaye, T. Morph: A longitudinal image database of normal adult age-progression. In *Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on*, IEEE, 2006, pp. 341–345.

[3] Rothe, R.; Timofte, R.; et al. Dex: Deep expectation of apparent age from a single image. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 10–15.

[4] Escalera, S.; Fabian, J.; et al. Chalearn looking at people 2015: Apparent age and cultural event recognition datasets and results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 1–9.

[5] Antipov, G.; Baccouche, M.; et al. Apparent age estimation from face images combining general and children-specialized deep learning models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 96–104.

[6] Agustsson, E.; Timofte, R.; et al. Apparent and real age estimation in still images with deep residual regressors on APPA-REAL database. In *Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on*, IEEE, 2017, pp. 87–94.

[7] Lapuschkin, S.; Binder, A.; et al. Understanding and Comparing Deep Neural Networks for Age and Gender Classification. *arXiv preprint arXiv:1708.07689*, 2017.

[8] Franc, V.; Cech, J. Learning CNNs for face recognition from weakly annotated images. In *Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on*, IEEE, 2017, pp. 933–940.

[9] Chen, S.; Zhang, C.; et al. Using ranking-cnn for age estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[10] Niu, Z.; Zhou, M.; et al. Ordinal regression with multiple output cnn for age estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4920–4928.

[11] Gurpinar, F.; Kaya, H.; et al. Kernel ELM and CNN based facial age estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2016, pp. 80–86.

[12] Yi, D.; Lei, Z.; et al. Age estimation by multi-scale convolutional network. In *Asian Conference on Computer Vision*, Springer, 2014, pp. 144–158.

[13] Wang, X.; Guo, R.; et al. Deeply-learned feature for age estimation. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, IEEE, 2015, pp. 534–541.

[14] Levi, G.; Hassner, T. Age and gender classification using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 34–42.

[15] Eidinger, E.; Enbar, R.; et al. Age and gender estimation of unfiltered faces. *IEEE Transactions on Information Forensics and Security*, volume 9, no. 12, 2014: pp. 2170–2179.

[16] Russakovsky, O.; Deng, J.; et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, volume 115, no. 3, 2015: pp. 211–252.

[17] He, K.; Gkioxari, G.; et al. Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017: pp. 2980–2988.

[18] Schroff, F.; Kalenichenko, D.; et al. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[19] Chollet, F. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint*, 2016.

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

[21] Szegedy, C.; Liu, W.; et al. Going deeper with convolutions. Cvpr, 2015.

[22] He, K.; Zhang, X.; et al. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[23] Sochman, J.; Matas, J. Waldboost-learning for time constrained sequential detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, IEEE, 2005, pp. 150–156.

[24] Keskar, N. S.; Socher, R. Improving Generalization Performance by Switching from Adam to SGD. *arXiv preprint arXiv:1712.07628*, 2017.

[25] Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[26] Mishkin, D.; Sergievskiy, N.; et al. Systematic evaluation of CNN advances on the ImageNet. *arXiv preprint arXiv:1606.02228*, 2016.

Appendix **A**

Contents of enclosed SD card

All supplemental graphical content, evaluation results, Wikipeople database description files (i.e. datatabase metadata), Python source files (of the crawler and of the CNN model), saved model weights. Additional materials provided on the project webpage at: `http://cmp.felk.cvut.cz/~xfrancv/pages/wikipeople.html`