CCN for Automatic Data Collection and Filtering

Saverio Meucci
saverio.meucci@stud.unifi.it

Abstract

In this paper it is presented a procedure to automatically create a large dataset of images containing faces, that will be used to train a CNN. This procedure is based on two phases: first, the automatic collection of the images thanks to some search engine and second, a filtering both automatic, thank to a pre-trained CNN, and manual, with the help of a tool for visual validation.

1. Introduction

Modern convolutional neural networks (CNNs) show more and more promising results in today's research. However, training such networks to have the best possible performance require a huge amount of data. Acquire those data is not always an easy task: most very large dataset are non freely available to the public; also, build from scratch such a huge dataset of millions of contents, like images, require a remarkable effort, not only in terms of acquiring those contents, but especially in terms of the annotation process, meaning a ground-truth creation, fundamental for any training technique.

This different of accessibility to data represents a bottleneck for the research in this field, where only research teams with adequate resources can afford to obtain or directly create large datasets and therefore do experiments and achieve results that are significant, since the number of very large dataset publicly available is quite small.

In this paper it is discussed the creation of a procedure, based on the work of [1], thanks to which build a large dataset of images that can be used in the training process of a CNN. This procedure takes advantage of the modern search engines, and, at the same time, has the objective of decreasing at a minimum the cost in terms of human time necessary to the acquisition and annotations of the images of the dataset, from the collection phase to the filtering and classification phases, leaving to a human user only the final phase that consists of a visual validation of the automatically obtained results.

As anticipated earlier, this procedure is divided in two main phases. The first one consists in the creation of the dataset, that is the acquisition of the images from the web; the last phase, instead, consists in the filtering of the acquired images, thank to the usage of a face detector and a classification process. This phase has the task of clean up the dataset in order to make it usable for a training process of a CNN. Finally the results are validated by a human user with the help of customized web app.

This procedure is mainly focused on images containing faces.

2. Dataset Creation

The phase for the dataset creation consists in the acquisition of images exploiting the modern image search engines. To begin with, it is necessary to decide a list of personalities that are sufficiently popular, so that for each identity there will be a high number of images, between 500 and 1000) as results of the search engines. Once a list of identities have been determined, we can proceeds to acquire the images.

The acquisition procedure is divided in two parts: a collection phase and a download phase.

The collection phase takes advantage of three search engines, that are Bing, Yahoo and AOL, to query and obtain images for each identity. A query returns a HTML page, that contains a limited number of results for an identity; that means that the query needs to be repeated with an increasing offset, until the desired number of results have been retrieved. Each HTML page contains the links to the image related to the query. Because every search engine has a different layout for the returned HTML pages and a different way of presenting the links in the pages, it is necessary to implement a parser for every search engine used. Moreover, since most search engines do not allow too many consecutive queries, two proxies have been used that divided the queries from the search engines, in order to not saturate the availability of the requests.

Once that the links of the images are obtained for each identity, each link is saved on a table of a database with information about the related identity, the search engine used and the rank of the result.

The download phase takes place once the database of links have been created. The procedure is straight forward: the images are downloaded and saved in a folder related to the identity, using the links contain in the table. This phase does not require particular precautions such as the usage of a proxy to mask the IP address, since the requests for the links are forwarded to different IP addresses and not the few same ones, such as it is with the collection phase, where only the ones of the search engines are used.

After these two phases we obtain a dataset of images for a list of identities automatically acquired from the Internet. The size of the created dataset is customizable changing the number of identities used and the number of links that are obtained for every identity from each search engines.

3. Face Detection

The previous phase had the task of acquiring the images to create a large dataset. The successive phases have the objective of filtering the obtained dataset, in order to remove, for each identity, the images that are not relevant. In fact, the results of the queries to the image search engines for a specific identity, will also contain images that are not related to that identity, thus the need of a filtering phase. Also, since the objective of this paper is the creation of datasets of images containing faces, we need to detect and select the faces in the retrieved images.

This task is accomplished implementing a face detector, written in C++ using the dlib library [2] . This face detector is compile into an executable that takes an image, or list of images, and returns the coordinates of the bounding box surrounding the detected faces. The dlib C++ library implements a face detector by using the classic histogram of oriented gradients (HOG) feature combined with a linear classifier, an image pyramid and sliding window detection scheme.

Figure 1: Example of a face found by the detector with its relative bounding box.

Considering the retrieved images, some will have no detected faces, others will detect the face of the identity the image is related to, others still will detect more than one faces, meaning that not all of them will be of the related identity but will belong to other subjects present in the images. This problem will be tackle in a later section of this paper.

To recap, this phase processes each image of the dataset with a customized face detector and obtains the coordinates of the bounding box related to the detected faces. These coordinates are then save in a table of the database along with the image from which are extracted and the identity related to the image.

In the next phases we will refer to images as the detected bounding box, even though these are not saved as images until the end, to avoid duplicating the data.

4. Duplicate Removal

Using an image search engine, it is easy to notice that in the results there will be present images that are duplicated, both because the returned links refer to the same location and because the same image is stored in to two different locations. Whatever the case, in the created datasets there will be presents a certain number of duplicated images for each identity. That is not desirable, since the dataset need to be used to train a CNN. A training phase need to generalized as much as possible the objects that is learning; a duplicated image does not add any useful information to this process.

These duplicated images need to be removed from the dataset with a duplicate removal technique to improve purity. For this task, the images are represented as extended bounding boxes related to the detected faces , in order to consider not only the face but also the contest of the image; the entire images are not used to decrease the computational time needed., since the images obtained with the extended bounding boxes are much smaller than the original. That is necessary because, since we are building a very large dataset, millions of images need to be processed.

For each image a feature vector are computed using the Vector of Locally Aggregated Descriptors (VLAD) encoding, using the implementation from the VLFeat library [5]. The VLAD is a feature encoding and pooling method that encodes a set of local feature descriptors extracted from an image using a feature dictionary built using a clustering method. These feature vectors are then clustered within the images for each identity; only a single element per cluster will be retained in order to removed the duplicated images.

The retained images are saved in the database without removing the duplicate images yet; this is done because if there is an error in the procedure the original images are not lost, since acquiring millions of images from the Internet is a n expensive task.

5. Classification

At this point, we have a set of images and a database containing information, thanks to which we can obtain a dataset of images of faces associated with each identity of our list and where the duplicates have been removed.

Still we are not sure about the association between faces and identities: the original image might contain more than one face leading to multiple bounding boxes and so multiple images, each containing a face, but without knowing which of these faces needs to be correctly associate to the identity;

moreover, the face detector might make a mistake and detect as face something in the image that is not a face.

Therefore, it is necessary a classification phase in order to clean the dataset from the images that are not relevant to the associated identity.

In this phase, we consider as images the bounding boxes extracted by the face detector, in order to focus the classification to the detected faces. Taking advantage from a pre-trained CNN of faces from 1000 identities, we extract fom the net the last layer of convolution as feature vector from each images in the dataset. Later, for each identity, we take 50 images and their feature vectors to train a linear SVM. Thus, we obtain a classification model for each identity. These images that are used in the training phase, were chosen by taking in to account the ranking of the search engines, assuming that the higher the ranking the more relevant the images, meaning that they will more likely contain faces correctly related to the associated identity.

In order to reduce the computational complexity, each identity is trained against other five identities randomly chosen. This choice does not compromise the accuracy of the classification because we are not interest in classifying an image by associate it to the correct identity; we are interest in determine if an image, that is already associated to a identity, is correctly associated to that identity or not; It is, in fact, a simple binary classification for each identity and so we need to train each identity separately. In fact, it is less expensive, in terms of time complexity, to train many models but with a dataset of limited size, than to train one model with the entire dataset containing a million images.

Once the training phase is over, for each identity we will obtain a model that will be used to classify the faces detected in the previous steps and remove the ones that, according to the prediction, do not belong to that identity. The information of the classification are stored on the database, associating to each image the results of the prediction: 1 if they are correctly associated to the identity and 0 if they are not.

The feature vectors were extracted by the net using the MatConvNet toolkit [4] and the model was trained using the LIBSVM library [3].

6. Validation

As expected, the classification phase is not perfect ma but it is subject to a degree of error. This error depends on the intrinsic nature of machine learning, but also by the fact that the faces used in the training of the models are not sufficiently clean. For example, if an image contains two faces, each of them will have the same ranking; thus, if the ranking is high, they will both be used in the training phase but only one of them will correctly represent the identity. Since it is not possible to resolve these problems in the previous steps, we have classification models that have been obtained with a training set containing errors.

In order to resolve these errors and to evaluated the performance of the procedure, a method for a visual validation of the results of the previous phases, and in particular to the ones of the classification, was implemented. A web application was created for this purpose, thank to which are shown to a user the images associated to each identity and the results of the classification. The original images are cropped using the extended bounding boxes, in order to allow a better and lighter visualization in a web browser. The images with a red overlay represent the ones that the classifier consider not belonging to that identity; the ones without overlay are considered correctly associated to that identity. An user can interact with the web application, by browsing each identity and the images that are shown in a gallery; the user can also double click an image to correct the results of the classification: double clicking an image with a red overlay will remove the said overlay; double clicking an image without an overlay will add a red overlay.

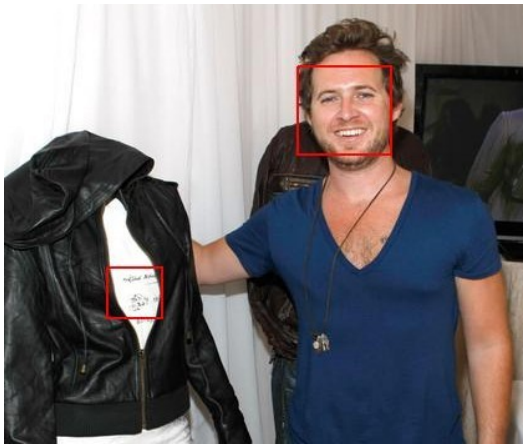

Figure 2: Example of an image for which the face detector return a face but also an error.

This process is faster than the standard annotation process since the user only need to validated the results without the need to label each images in the first place. Moreover, the interface was designed to be intuitive and to function with the minimum number of operation to execute by the user. The user can also choose to remove an identity from the dataset; the reference image for that identity will be shown in red as a result.

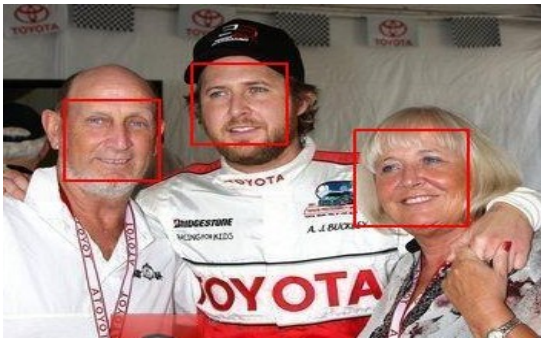

Figure 3: Example of an image for which the face detector detects more than one face.

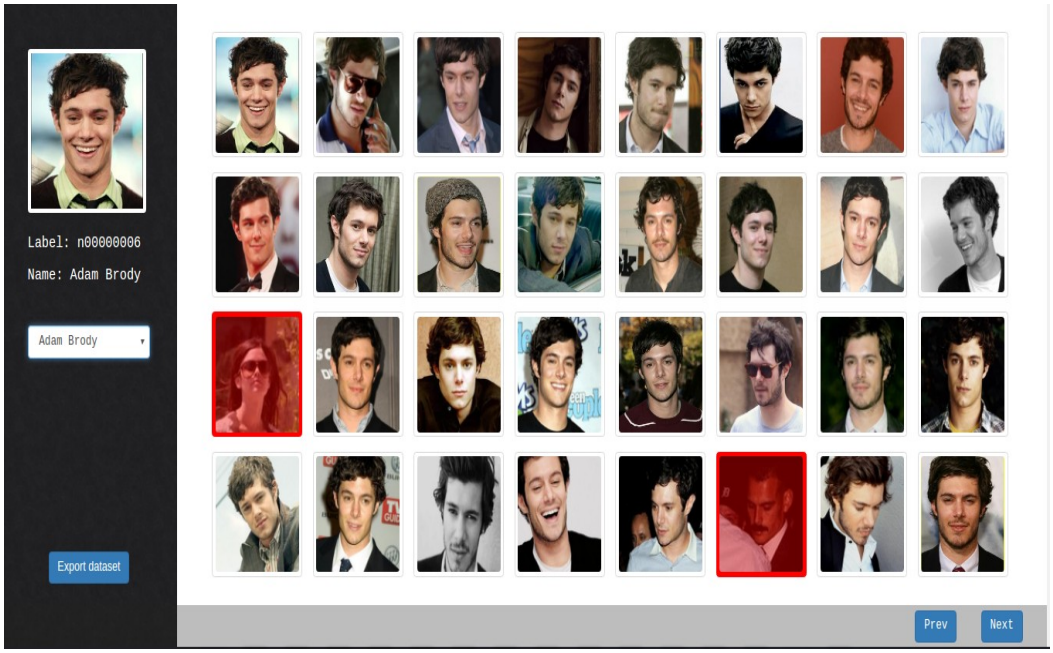

Figure 4: A screenshot of the implemented web application.

By validating the results, the user is also creating a ground-truth that will be later used to measure the performance of the classification. The changes produced by the user will be saved in the database, in order to be persistent.

After the visual validation, we will have obtained a very large and clean dataset, automatically built except for the last phase, that can be used for the training of a CNN.

7. Experiments

To evaluate the performance of this procedure experiments have been performed with two different datasets. One dataset (test A) contains 50 identities, such as celebrities and actors, the other (test B) contains 50 identities, such as football players. The identities for the first dataset are a selection of the ones used by the pre-trained CNN that was used to extract the feature vectors for the images.

All the phases of the procedure have been executed to create this two dataset. Finally, the visual validation was carried out thanks to the implemented web application (Fig. 4). Using the ground-truth created by the validation phase, it was possible to confront the results of the automatic classification.

Test	TPR	TNR	FPR	FNR	Accuracy
A	0,993	0,874	0,126	0,007	0,973
B	0,978	0,839	0,161	0,022	0,952

Table 1: Values of TPR, TNR, FPR and FNR along with the classification accuracy for test A and B.

As shown in Table 1, the results are good for both test A and B. As expected, test A performed better than test B because its identities are the ones that the pre-trained CNN used, leading to better feature vectors. However, the results for test B are still good and only slights worse that the first test. It is to be noted that the method recognized better the faces that are correctly associated to the identity than the ones that are not, as one can see from the higher value of FPR than the value of FNR.

In total, the average number of images per identity for the prediction is 409 for test A and 439 for test B; the average number of images per identity after the validation is 412 for test A and 448 for test B.

8. Conclusions

The implemented procedure works as follows. For each identity, the images are downloaded from selected search engines; a face detector is used to detect the faces contained in the images; a duplicate removal method is used to removed images that are duplicate; using a pre-trained CNN, feature vectors are computed for each image and linear SVM is trained for each identity in order to remove the images that do not belong to that identity; finally, the results of the classification are validated and corrected by an user thanks to a web applications specifically developed for this purpose.

However, there are some matters that need to be addressed:

- The collection phase is not reliable since it depends on how the search engines present the HTML pages. This can change at any time and so a new parser

need to be implemented and in the worst case a new search engine need to be used.

- The duplicate removal phase is only able to remove perfect duplicate. Images with different resolutions or scales are perceived as different and they aren't always removed, as shown in Fig. 5. These leads to redundancy in the dataset.
- As mentioned at the beginning of the validation section, having more than one faces in an image can lead to having a training set for the classification that contains error, negatively affecting the computed model.
- Not every identity has many results in the search engines; also, an identity can have a name that leads to bad results in the search.
- The whole procedure is quite slow especially for the classification phase and for the visual validation.

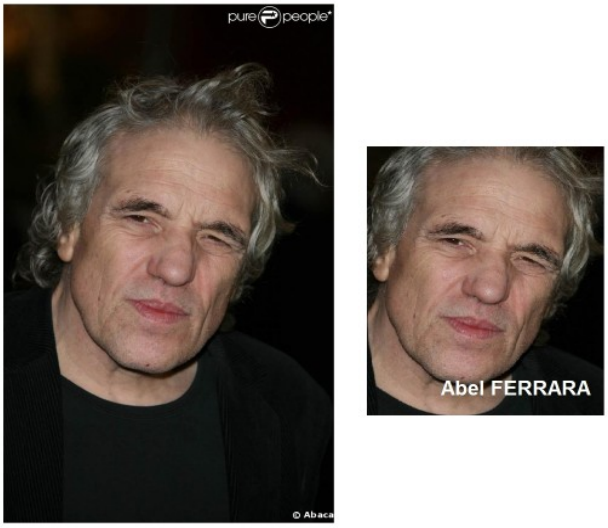

Figure 5: Example of two images, representing the same photo, but with different resolutions and scales so that the duplicate removal phase is not able see them as duplicates.

Regardless of this problems, the procedure shows promising results, leading to a dataset of very large dimension that is automatically built from scratch. The only phase that need manual intervention is the last one, that is the visual validation step.

References

[1] O. M. Parkhi, A. Vevaldi, A. Zisserman, Deep Face Recognition, British Machine Vision Conference, 2015.

[2] Davis E. King, Dlib-ml: A Machine Learning Toolkit, Journal of Machine Learning Research, 2009, 10, 1755-1758.

[3] Chang, Chih-Chung and Lin, Chih-Jen, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent System and Technology, 2, 3, 2011, 27:1—27:27.

[4] A. Vevaldi and K. Lenc, MatConvNet: Convolutional Neural Networks for MATLAB, Proceedings of the ACM Int. Conf. On Multimedia, 2015.

[5] A. Vevaldi and B. Fulkerson, VLFeat: An Open and Portable Library of Computer Vision Algorithms, 2008.