Machine learning for page recognition

Morgan Asnar

09/4/2020

Contents

1 Problem definition

The following report address this particular problem; being able, from a picture of a page of a specific scholar manual, to return the page number.
As a database, we have a batch of 20 pictures for every manual page. The manual is composed of 180 pages, thus resulting in a total of 3600 pictures. Most of these pictures have a format of 3000 pixels length, and 4000 pixels height. Some are sideways, which result in pictures of 4000*3000 pixels.
This problem has two major difficulties. Firstly, *the size of the pictures* makes it hard for a computer to load them. Secondly, *we have very few exemples*, which will make any machine learning algorithm struggle to barely learn anything.
In addition, we should also considere that in order to verify our protocol, a part of this database should not be used for its training. To this end, we split the database into a training and a validation set, the latter representing 20% of the database. For each page, we are now left with 16 pictures to train the network,and 4 pictures to test it, thus reducing an already very small database.
Following are the different used approaches.

2 Deep learning approach

2.1 Database treatment

The main difficulty with a deep learning approach is to avoid overfitting. To this end, we will use two main concepts; dropout layers and data augmentation. For the latter, keras' built-in functions were used. The transformations applied to the pictures were the following. They were chosen such as, when applied to a picture of a page, they would give a picture of the same page.

- Noise addition
- Brightness modification
- Shear mapping
- Zooming
- Color shifting
- Rotation
- Vertical/horizontal shifting

In order to make such pictures load in batches large enough to allow our neural network to converge in reasonnable time, they were also reduced to a resolution of 300*400 pixels, which remains readable.

2.2 Network design

In order to improve learning efficiency of the network, it was chosen to use as first layers a pre-trained convolutional neural network, from which the last layer was removed. The goal of this is to process the pictures in order to extract the main pattenrs. As an output, we should have a picture that highlights mostly the borders of the letters, or the figures in the pages. These layers are not to be trained.
We then add k fully connected layers of N nodes, whith a dropout probability d. The goal of these is to extract features from the picture we got as an output from the previous layer. We shall put a rather high dropout in order to avoid them to learn our training dataset instead of their features. These are to be trained.
At the end of this network, we add a fully connected layer of 180 nodes, followed by a softmax layer to compute the probability for an input to be one specific page. the fully connected layer is also to be trained.
In the end, the output with the highest probability is the one we will keep as the class our network estimated.

2.3 Results and interpretations

This network, although capable of a bit of learning (validation up to 5%, while a network answering randomly would be at 0.56%), cannot be used as is.
The main reason seems to be the fact that the database is too small. However, we may argue that this network could be significantly improved, mostly by

tuning the parameters of the network mentionned earlier; k, N and d. This would yet require too much time, as the network take several days to reach a stable state (and end up overfitting).
It is also to be mentionned that the pre-trained network I used as a base was trained for a very general purpose. We may assume that a network trained on recognizing writting would lead to better results. However, the pages of the database being written in Odia, which is one of the numerous indian dialect, it is difficult to find a network trained on it.

3 K-neirest neighbours approach

3.1 Database treatment

This methode, as opposed to the last one, require a more structured database. In deep learning, we want our datas to be as big as possible in order to find the main features. Here, we want a few datas close enough to each other to form coherent clouds of datas.
To this end, every sideway picture was rotated to be straight. In order to limit datas, every picture is also converted to gray scale, which should be enough to recognize every page.
In order to limit computation, every picture was resized to 375*500, and "flattened" to a vector.

3.2 Results and interpretations

We might wonder if this approach is really relevant. To illustrate the efficiency of this method, we can observe the distances between a picture we want to classify and all of our reference pictures (figure 1).

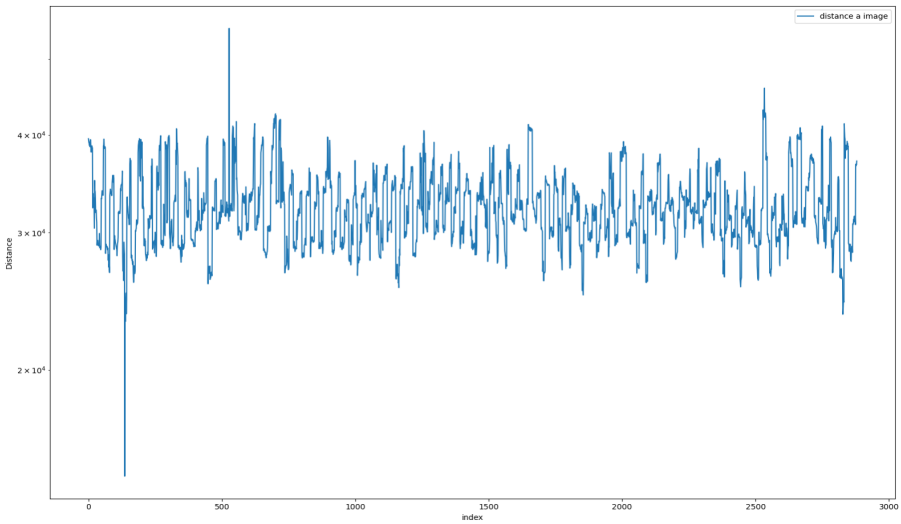

Figure 1: Euclidean distances between an image we want to classify and reference images

Here, we may observe that there are a few pictures for which the distance is significantly smaller than the mean. We have this kind of shape for most pictures, which tends to show that this method may work pretty well.
On figure 2 are the rates of correct classification with this method, using k = 1, 3, 5, 7, 9, 11, 13.

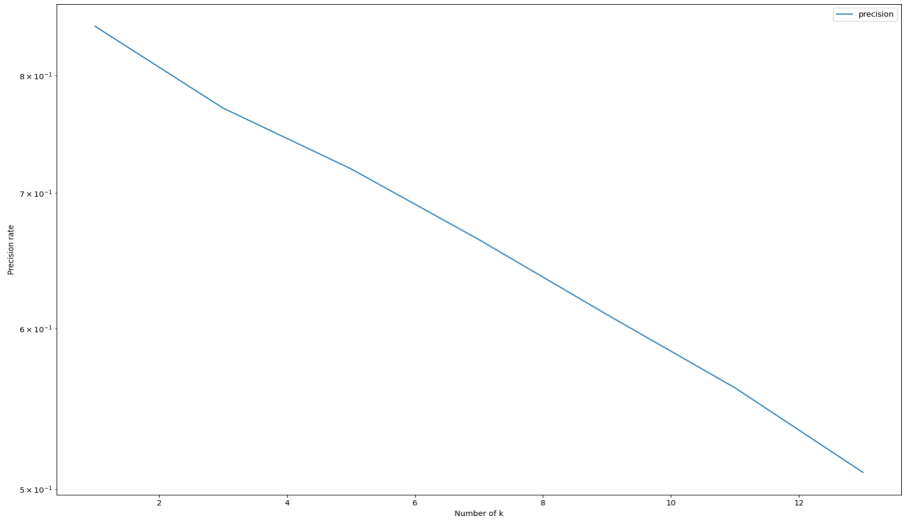

Figure 2: Rate of correct classification for several k values

On this figure, we see that when we increase the value of k, we have decreasing results. This may be due to the fact that all pictures being from the same book, they may all be pretty close to one another. However, it could also be due to the lack of variety in the database, making some pictures look too much alike. Basically, two pictures of the same page were taken following, so all the conditions were pretty much the same.
This implies that using these datas for a cellphone application, as an exemple, could lead to poor results, as the angle of the picture, or the lighting would change too much. It is also to mention that the results are slow (approximately 1 second), due to the comparison with every picture in our database.

4 Getting a better model with SIFT

4.1 Database treatment

Because we are using natural pictures, whose lighting, tilt and scale may vary greatly, it could be useful to change our algorithm in such a way that we extract the most stable features in the pictures. To this end, one can use the Scale-Invariant Feature Transform [1].
We apply the SIFT algorithm to our 375*500 gray scaled pictures, thus resulting in arrays of descriptors for the keypoints of our pictures.
For every query picture, we also compute these descriptors. We then match

[1] David G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", 2004

them to the closest ones in a reference picture, and compute the mean value of the distance between between the descriptors of the two pictures. We then apply a kNN algorithm to get the closest picture in terms of descriptors in our dataset.

4.2 Results and interpretations

This method requiring more computation than the previous one, the computer used for this was running at full capability, leading to way slower results, almost 30 seconds per query. It is likely that more computation power would lead to faster results. However, it can't be told of how much it would improve.
However, the observed result is a recognition rate of 1. Every single picture of our test database was correctly classified, using a kNN with k=1. Greater values where not tried due to computation time. The distances between descriptors of one picture and the descriptors of the pictures in the reference database has way less variance, as can be observed in figure 3.

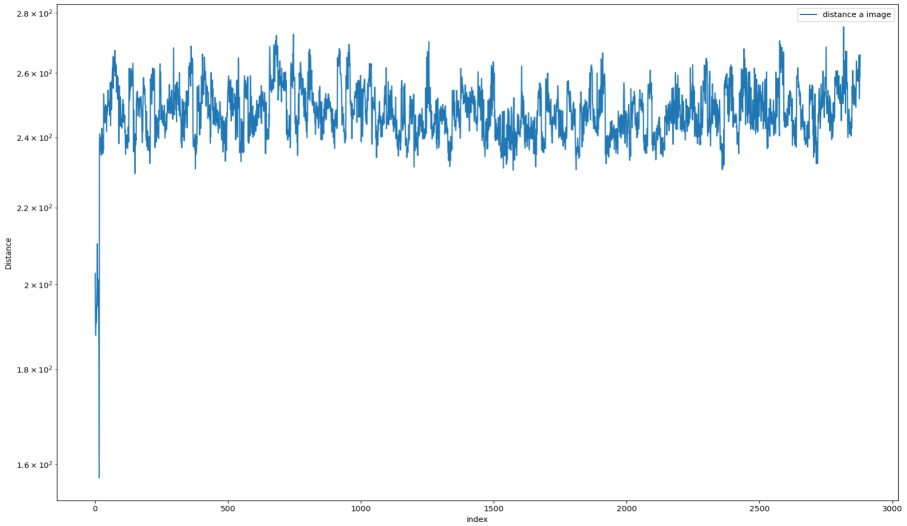

Figure 3: Mean Euclidean distances between the descriptors of an image we want to classify and the ones of reference images

With respect to this, we may try to improve the efficiency of this method, by reducing the size of the reference database. To this end, we create a specific sub-database of 3 reference pictures per class, which signifanctly decrease the number of references.
The computing time gets better, around 5 seconds per query, but is still quite long, depending on the application we want. The results are rather correct, with a classification rate of 0.9931 for a kNN with k=1, and 0.9917 for k=3. Even if the results are decreasing along k, it is to note that this might provide better results for a more general purpose, with more various query pictures.
At this point, we may also try to use only one reference picture for every page. This test gave a correctness rate of 0.9889.

We may argue that all pictures don't have the same amount of keypoints. Because of this, if we force match a picture containing numerous keypoints with one that contains fewer keypoints, the mean distance between all descriptors will greatly increase, as these keypoints don't exist in the reference picture. It is also to mention that force matching our keypoints with every keypoints of every reference picture takes a long time.
To avoid this, we create a database of all keypoints from every reference picture, and use matching of our query picture with this database, instead of the keypoints of only one picture. The returned list may contain keypoints from various pictures. We considere that the picture with the highest number of descriptors in this list is of the same class as the query picture. This methode gave a correctness rate of 1, with a query time of approximately 3.5 seconds. By doing so, we also have a major difference, which is that we search once through a bog database, instead of searching several times in smaller ones, as we were doing up until now. The fact that we were using small databases constrained us to use brute force search, as another approach would not have worked for such small dataset. Now, we may use approximation algorithms to get faster results. To this end, we use FLANN based matcher [2]. The recognition rate stays the same, and the query time drops down to 2.5 seconds.
It is to mention, despite the fact that every picture were rotated to be straight, this could be applied to rotated pictures, SIFT being rotation-invariant.
It was tried on the original picture collection, described at 2.1. The recognition rate is still of 1, and the query time remains similar.

5 Conclusion

The initial approach of this problem was to use deep learning to extract features from our reference pictures. However, due to the size of the database, it was impossible to get a model to not overfit. An Eigenface [3] -like approach was also tried, but not developed here due to very poor results. This is most likely due to a too high inter-classes variance.
A classic kNN approach performed rather well with low k values, which could hint that the intra-class variance was not too high, and that a more classic computer-vision approach could lead to correct results. However, the fact that the performances dropped as k increased could imply that a pixel-wise approach would not be sufficient.
This led us to try to find keypoints we could compare. This was done using the SIFT algorithm to get descriptors for every picture, and by comparing them. This approach led to great results, and due to its nature, allowed us to signifanctly reduce the reference database, to a single picture per class. However, it is to mention that this method takes some time, which might not be correct for every application. To reduce this time, and to improve correctness, we applied the matching to the whole keypoints dataset, and changed the way we searched through it.

[2] Relja Arandjelovic, "Three things everyone should know to improve object retrieval", In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), CVPR '12, pages 2911–2918, Washington, DC, USA,

[3] M. Turk, A. Pentland, "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71-86, 1991.

6 Annex

6.1 Regarding the code

This code was made in python 3, and makes great use of openCV. However, as SIFT is a patented algorithm, we should get a license to use them for a commercial use. As this application is stricly for non-profit application, this use is acceptable. However, for this reason, the SIFT algorithm is not implemented in the latest versions of the default openCV anymore. To use this algorithm, we use the opencv_contrib package, available on github. If the creation of the SIFT objects is not working, you may use the following, with package manager pip.

```
pip uninstall opencv-python
pip install opencv-contrib-python
```