Adaptively learning to recognize symbols in handwritten early music

Luisa Micó, Jose Oncina, and José M. Iñesta

Dept. of Software and Computing Systems, University of Alicante, Spain
{mico,oncina,inesta}@dlsi.ua.es

Abstract. Human supervision is necessary for a correct edition and publication of handwritten early music collections. The output of an optical music recognition system for that kind of documents may contain a significant number of errors, making it tedious to correct for a human expert. An adequate strategy is needed to optimize the human feedback information during the correction stage to adapt the classifier to the specificities of each manuscript. In this paper, we compare the performance of a neural system, difficult and slow to be retrained, and a nearest neighbor strategy, based on the neural codes provided by a neural net, trained offline, used as a feature extractor.

1 Introduction

Optical Music Recognition (OMR) investigates how computers can read music notation in scores. Although research in printed modern notation has achieved good performances [5], the task becomes much harder when dealing with collections of early music handwritten scores. In particular, this work is applied to documents written in the Spanish white mensural notation system from the 16th and 17th centuries (see Fig. 1), for which a perfect recognition cannot be expected as the initial output of the OMR system [4].

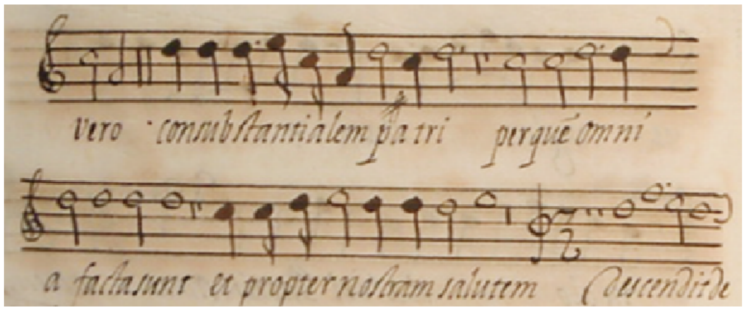
vero consubstantialem patri per quē omni
a facta sunt et propter nostram salutem descendit de

Fig. 1: The kind of documents processed (here a fragment of a page) are single-voice vocal music written in the Spanish variant of the white mensural notation.

The automatic pattern recognition approach has been traditionally focused on accomplishing a fully-automated operation. Nevertheless, in our approach, complete automation is not possible, although a perfect transcription of the original documents is needed for editing and publishing a collection. Therefore, we have to focus on the human-machine interaction tasks and how to optimize the expert user feedback loop [6].

The errors made by the system are usually seen as an issue outside the research process because correcting them is considered as the procedure for converting the system hypothesis into the desired result. However, semi-automatic approaches in which the human operator has the eventual responsibility of verifying and completing the task are the key to an efficient solution [7].

In this paper, we will study how using the user's corrections help the classifier to learn its model incrementally, decreasing the error throughout the task. Deep convolutional neural nets (DCNN) [3] are improving the state of the art in computer image analysis tasks. Although there are works already that permit to modify these recognition models as a continuous learning process as new classes of data arrive [9], the task is still computationally demanding. We explore the possibility of combining the ability of DCNN for extracting good image features, with the simplicity of a nearest-neighbor (1-NN) classifier to adapt its performance to a specific training set along the edition stage.

2 Data structure

The dataset for this study is a collection of pages $\mathcal{P} = \{P_1, P_2, ..., P_{|\mathcal{P}|}\}$ annotated with their ground-truth categories. This way, each page P_p can be considered as a training subset $\mathcal{X}^{(P_p)} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{|P_p|}$, where the $\mathbf{x}_i$ represent the symbol bounding boxes in page P_p and y_i their corresponding labels.

The symbol bounding boxes were extracted from the image and re-scaled to a 30×30-pixel window (only if the bounding box is bigger than that), keeping the aspect ratio of the box and padding the background with its maximum pixel value (see Fig. 2 for some examples).

When this window is the input for the DCNN, no additional processing is made, since the filters of the network input layer process the window regardless of its size. Nevertheless, the 1-NN will classify every window considering it as a vector $\mathbf{x} \in [0, 255]^{30\times30}$. Due to its sensitivity to the dimensionality of the feature space, we have transformed the windows by downsampling, keeping the central pixel of every non-overlapping 3×3 pixel area, assigning to it the mean of the 9 pixels involved. This is equivalent to low-pass filtering of the window, keeping the main features of the image in a smaller space ($[0, 255]^{10\times10}$).

3 Incremental learning

The key point in this work is to study how can we adapt the recognition model to the data and do it incrementally. In real operation, a collection of scores is

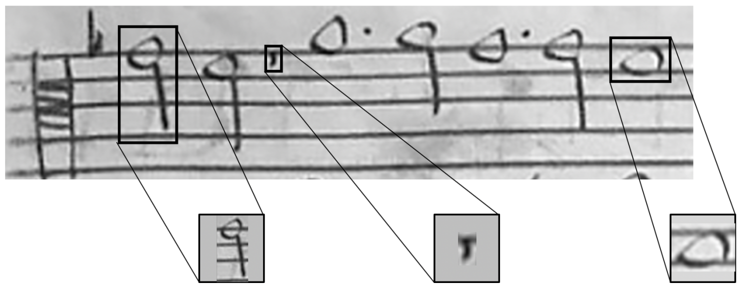

Fig. 2: Examples of bounding boxes for some symbols and how they are adapted to a 30×30 window in different situations. (Left and right:) fit to a 30×30 window, keeping aspect ratio and background padding; (center:) no bounding box stretching is done when it is smaller than the target window.

presented to the user by pages. Each page is processed, and the symbols are classified (see [1] for details). Then, the user makes corrections to the symbols that were incorrectly classified. This happens when the system hypothesis does not match the ground-truth label or when the symbol belongs to previously unseen classes. User corrections will be simulated. These interactions are utilized to improve the model for the classification of the next pages.

The recognition algorithm (the *model* $\mathcal{M}$) is a key issue in any pattern classification system, but in an interactive architecture, the most relevant feature is the ability of the algorithm to adapt to the specificities of the data through the error corrections made by the user. In the interactive paradigm, the efficient exploitation of *human expert knowledge* is the main objective, so the correctness of the system output is no longer the main issue to assess. The challenge now is the development of interactive schemes capable of efficiently exploiting the feedback to eventually reduce the user's workload.

In light of that, we have selected a very simple, but flexible, classification algorithm as the nearest neighbor is. It does not need a parametric analysis of the feature space for operation, and the training set $\mathcal{X}$ can be incrementally built by adding new pairs as they are found in the input in operation time: $\mathcal{X}^{(k+1)} = \mathcal{X}^{(k)} \bigcup \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$. Only an initial model $\mathcal{M}^{(1)}$, trained offline, is needed to start classifying. This model can be trained with the symbols on the first page $\mathcal{X}^{(P_1)}$, including the labels for the symbols on it, $y_i \in \mathcal{C}^{(P_1)}$, or with an initial subset of pages if the model needs more examples, as explained below.

Also, it is easy to add new classes dynamically by adding new labels, if needed. Besides, editing and condensing methods [8] can be easily applied to the training set if advised by the user corrections. The system must operate in real-time, so the user can interact with it comfortably. This is another feature that advises using simple, adaptive, and fast classification algorithms.

The algorithm outline is shown below (Algorithm 1). As explained, errors in a page P_p can be due to symbols belonging to unseen classes. In such a case, the interaction step includes the addition of the new class to the training set, with the symbols seen on the current page as prototypes. This algorithm will try to minimize the number of errors $\ell^{(p)}$, and so the need of user corrections, as the pages are processed.

Algorithm 1 Outline of the method

```
**Input:** A collection of pages $\mathcal{P} = \{P_1, P_2, ..., P_{|\mathcal{P}|}\}$
$\mathcal{X}^{(1)} = \mathcal{X}^{(P_1)}$
Train the model $\mathcal{M}^{(1)}$ with $\mathcal{X}^{(1)}$
for $p = 2$ to $|\mathcal{P}|$ do
    Apply $\mathcal{M}^{(p-1)}$ to samples in $\mathcal{P}_p$
    $\ell^{(p)} = |\{\mathbf{x}_i \in P_p \mid \hat{y}_i \neq y_i\}|$
    Interaction: the user fixes wrong $\hat{y}_i$ to actual $y_i$
    $\mathcal{X}^{(p)} = \mathcal{X}^{(p-1)} \bigcup \mathcal{X}^{(p)}$
    Train $\mathcal{M}^{(p)}$ with $\mathcal{X}^{(p)}$
end for
```

This algorithm does not change independently of the classification model utilized, $\mathcal{M}$. Only the $\mathcal{X}^{(1)}$ considered might be different, as explained below.

We want to explore a trade-off between accuracy in the classification and speed and flexibility in re-training the model. DCNN are state-of-the-art image classification methods, but the usual size of these models make their adaptation difficult and time-consuming. On the other hand, in many classical classification algorithms, like the 1-NN, the adaptation is straightforward, because it needs only updating the training set to adapt to a new situation in real-time (we consider as real-time any situation in which the user does not perceive that he or she has to wait for the system to make a decision).

Taking these considerations into account, we plan to compare three different classification models:

1. DCNN: the model $\mathcal{M}$ is a deep convolutional neural network. It is expected to achieve good performance (low error rates) but long retraining times.
2. 1-NN: $\mathcal{M}$ is a nearest neighbor classifier. Retraining and recognition can be done in real time, but higher error rates are expected.
3. NC+1NN: A DCNN learned on a subset of initial pages of $\mathcal{P}$ is used as a feature extractor (neural codes [2], NC) and the 1-NN is applied to the NC to implement the incremental classification described in the algorithm.

The network utilized is composed of 7 convolutional layers with 100 3×3 filters each. Then, a global max-pooling layer provides a $\mathcal{R}^{100}$ vector that will be the NC features for the nearest neighbor (in the case 3.) or fully connected to a layer with as many neurons as classes that will be classified with a softmax in the case of full DCNN classification (1.). The network architecture is displayed in Fig. 3. For training, Adam optimization has been used with a learning rate of 0.001 during 100 epochs, using minibatches of 64 images. Units have ReLU activations.

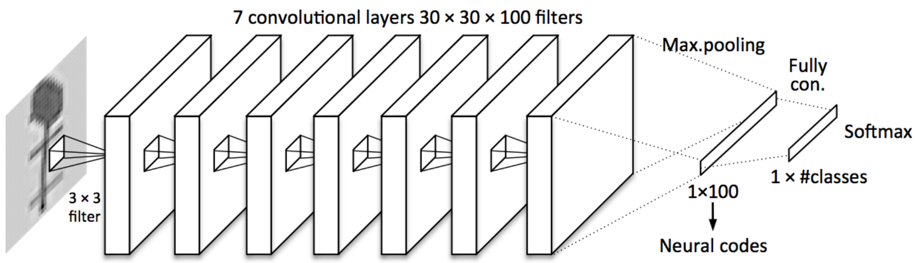

Fig. 3: Architecture of the DCNN used. Neural codes are the activations of the 100-neuron layer after pooling. For that, the last layer is removed and the neuron activations are the input to a nearest neighbor classifier.

4 Data and results

We have selected a Mass in A minor from a collection of sacred vocal music from the 17th century. In this case, we have $|\mathcal{P}| = 124$ pages. Each page has a maximum of 6 staves of monophonic music, containing between 20 and 30 symbols each on average, for a total of 17,114 samples.

Initial training set. According to the instructions in Algorithm 1, the initial training set is $\mathcal{X}^{(1)} = \mathcal{X}^{(P_1)}$, but some considerations about this follow.

The number of prototypes in $\mathcal{X}^{(P_1)}$ is 143 from 23 classes. The C and F clefs received special consideration. Since the initial pages were written in the G clef, 6 prototypes of each of the other clefs were included in $\mathcal{X}^{(1)}$ from other composition. This way, $|\mathcal{X}^{(1)}| = 143 + 12$ prototypes from 25 classes.

When the DCNN is utilized, either as a classifier or for feature extraction, we need more data to train such a large structure properly. For that, an initial subset of the first 16 pages was considered: $\mathcal{X}^{(1)} = \bigcup_{p=1}^{16} \mathcal{X}^{(P_p)}$. This way, $|\mathcal{X}^{(1)}| =$ 2440+12 prototypes from 44 classes. In this case, the Algorithm 1 runs for $p = 17$ to $|\mathcal{P}|$.

As the algorithm runs, the number of classes will increase. Each time an unseen class appears, it produces errors. As this page is included in the training set for the next step, the unknown class appears for the next iteration. The final number of classes is 53 for this composition.

Results. First, a study of the difference between with and without interaction is shown (see Fig. 4). The graph shows that, when the user corrections are used, a rapid drop in the error rate for the 1-NN is initially observed. The error rises again when another voice of the same composition begins to be processed. When the effect of the new pages is learned by the model, it is able to reduce the error again. Every time a change in the conditions happens the system degrades its performance, but it is able to continue learning later on, improving its performance.

On the contrary, when the training set remains fixed (using the 16 first pages for it), the error not only does not decay but tends to increase, because the system is not able to adapt to the specificities of the new pages.

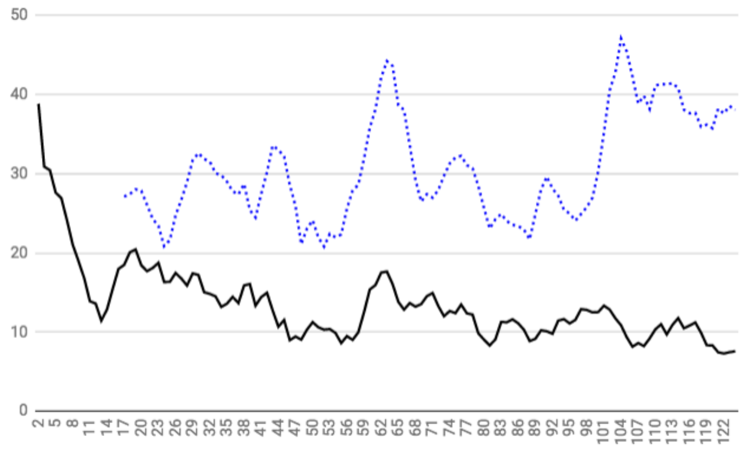

Fig. 4: Smoothed evolutions of the error rate (%) along the 124 pages of the studied composition, using 1-NN for classification. In solid line: error rate when using the interactive scheme. In dotted line: error rate without interaction, using a fixed training set.

One interesting feature is that pages from 60 to 65 have specificities not seen before. In that situation, both curves degrade. But when a similar situation occurs again from page 100 onwards, the incremental method is robust against that particular situation (solid line), while the non-incremental method is not and its performance degrades again.

Figure 5 shows the performance of the three recognition systems comparatively. When only the nearest neighbors are used (dotted line) the results are displayed from page $p = 2$, but when the nets are used they use the first 16 pages for initial training, so the results are displayed from $p = 17$.

The 1-NN adapts the best but performs the worst (an average error of 9.6% for the last 15 pages), but it runs in real time, without the user noticing any delay. On the other hand, the DCNN is able to reach a 1.1% of error at the end, but running the whole Alg. 1 using that model took 7749 seconds in a GPU computer (note from Alg. 1 that the network is retrained with the new training set after each page). The 1-NN applied to the neural codes was able to adapt to the data (less than using the 1-NN alone), reaching a nice 4.4% of error at the end. This approach works in real time, since the network learning is made only once, before starting the classification and adaptation.

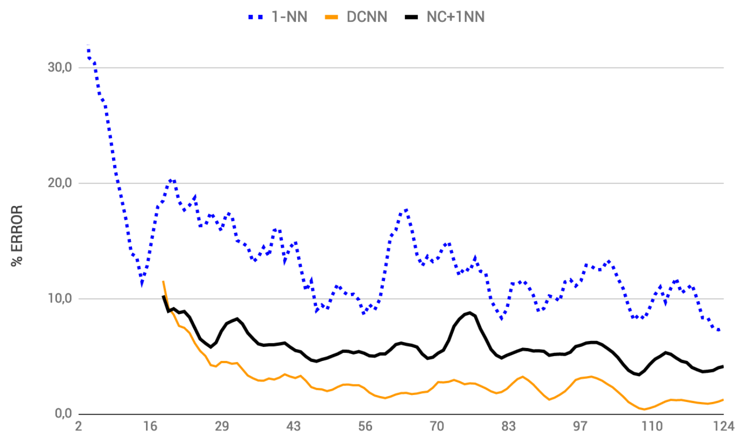

Fig. 5: Evolution of the error rates for the different methods considered. 1-NN (dotted): only nearest neighbors classifying the pixel values; DCNN (thin solid): only deep convolutional nets classifying the bounding boxes; NC+1NN (thick solid): nearest neighbors classifying the extracted neural codes.

5 Conclusion

The presence of the expert user in the learning loop opens up new possibilities for study adaptive learning algorithms. The presented study shows that a combination of DCNN acting as a feature extractor and a nearest neighbor for classifying the extracted neural codes can provide a good trade off between precision and real-time operation. In any case, there are many ways left to be explored to make the human-in-the-loop approach efficient and effective.

Acknowledgements

Work supported by the Spanish Ministry HISPAMUS project TIN2017-86576-R, partially funded by the EU.

References

1. Calvo-Zaragoza, J., Rizo, D., Iñesta, J.M.: Two (note) heads are better than one: pen-based multimodal interaction with music scores. In: Proc. of the 17th Int. Society for Music Information Retrieval Conference. pp. 509–514 (August 2016)
2. Calvo-Zaragoza, J., Gallego, A., Pertusa, A.: Recognition of handwritten music symbols with convolutional neural codes. In: 14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017. pp. 691–696 (2017)
3. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Neural Information Proc. Systems **25**, 1097–1105 (2012)

4. Pacha, A., Calvo-Zaragoza, J.: Optical Music Recognition in Mensural Notation with Region-based Convolutional Neural Networks. In: 19th International Society for Music Information Retrieval Conference. pp. 240–247 (2018)
5. Rebelo, A., Fujinaga, I., Paszkiewicz, F., Marçal, A.R.S., Guedes, C., Cardoso, J.S.: Optical music recognition: state-of-the-art and open issues. International Journal of Multimedia Information Retrieval **1**(3), 173–190 (2012)
6. Sober-Mira, J., Calvo-Zaragoza, J., Rizo, D., Iñesta, J.M.: Pen-based music document transcription. In: Proc. of the 12th IAPR Int. Workshop on Graphics Recognition, GREC (2017)
7. Toselli, A.H., Vidal, E., Casacuberta, F.: Multimodal Interactive Pattern Recognition and Applications. Springer Publishing Company, Incorporated, 1st edn. (2011)
8. Wilson, D.R., Martinez, T.R.: Reduction techniques for instance-based learning algorithms. Machine Learning **38**(3), 257–286 (2000)
9. Xiao, T., Zhang, J., Yang, K., Peng, Y., Zhang, Z.: Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In: Proc. of the 22nd ACM Int. Conf. on Multimedia. pp. 177–186. MM '14, ACM (2014)