Accurate Graphic Symbol Detection in Ancient Document Digital Reproductions

Zahra Ziran(✉), Eleonora Bernasconi, Antonella Ghignoli, Francesco Leotta, and Massimo Mecella

Sapienza Università di Roma, Rome, Italy
{zahra.ziran,eleonora.bernasconi,antonella.ghignoli, francesco.leotta,massimo.mecella}@uniroma1.it

Abstract. Digital reproductions of historical documents from Late Antiquity to early medieval Europe contain annotations in handwritten graphic symbols or signs. The study of such symbols may potentially reveal essential insights into the social and historical context. However, finding such symbols in handwritten documents is not an easy task, requiring the knowledge and skills of expert users, i.e., paleographers. An AI-based system can be designed, highlighting potential symbols to be validated and enriched by the experts, whose decisions are used to improve the detection performance. This paper shows how this task can benefit from feature auto-encoding, showing how detection performance improves with respect to trivial template matching.

Keywords: Paleography · Graphic symbol detection · Image processing · Machine learning

1 Introduction

A huge number of historical documents from Late Antiquity to early medieval Europe do exist in public databases. The NOTAE project (NOT A writtEn word but graphic symbols) is meant to study graphic symbols, which were added by authors of these documents with several different meanings. This task is very different though from processing words and letters in natural language as the symbols that we look for can be orthogonal to the content, making contextual analysis useless.

Labeling document pictures with positions of graphic symbols even in an unsupervised manner requires the knowledge of domain experts, paleographers in particular. Unfortunately, this task does not scale up well considering the high number of documents. This paper proposes a system that helps curators to

This research is part of the project *NOTAE: NOT A writtEn word but graphic symbols*, which has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Advanced Grant 2017, GA n. 786572, PI Antonella Ghignoli). See also http://www.notae-project.eu.

E. H. Barney Smith and U. Pal (Eds.): ICDAR 2021 Workshops, LNCS 12916, pp. 147–162, 2021.
https://doi.org/10.1007/978-3-030-86198-8_12

identify potential candidates for different categories of symbols. Researchers are then allowed to revise the annotations in order to improve the performance of the tool in the long run.

A method for symbol detection has already been proposed in the context of the NOTAE project [1]. Here, the authors have created a graphic symbols database and an identification pipeline to assist the curators. The symbol engine takes images as input, then uses the database objects as queries. It detects symbols and reduces noise in the output by clustering the identified symbols. Before this operation, the user is required to decide the binarization threshold from a prepared selection.

The approach proposed in this paper has several differences with the original one. First of all, in this new version of the tool, we rely on OPTICS [2], instead of DBSCAN [3], for clustering purposes. OPTICS uses the hyper-parameters MaxEps, and MinPts almost the same way as DBSCAN, but it distinguishes cluster densities on a more continuous basis. In contrast, DBSCAN considers only a floor for cluster density and filters noise by identifying those objects that are not contained in any cluster. In addition, our proposed pipeline implements an algorithm that sorts the objects of a cluster by confidence scores and selects the top match. So, the pipeline can control the number of predictions over different types of graphic symbols.

Moreover, the first tool has shown an high number of false positive, which are difficult to filter out. Here, we show how automatic detection of symbols can benefit from feature auto-encoding, showing how detection performance improves with respect to trivial template matching.

The paper is organized as it follows. Section 2 summarizes prior work on document analysis and digital paleography tools, metric learning, and graphic symbols spotting. Section 3 present our data cleaning process and image preprocessing tailored to the specific data domain, i.e., ancient documents. Section 4 covers the inner details of the proposed method. Section 5 show experimental results. Finally, Sect. 6 concludes the paper with a final discussion.

2 Related Work

In [7], the authors discuss the recent availability of large-scale digital libraries, where historians and other scientists can find the information they need to help with answering their research questions. However, as they state, researchers are still left with their traditional tools and limitations, and that is why they propose two new tools designed to address the need for document analysis at scale. Firstly, they consider a tool to match handwritings which is applied to documents that are fragmented and collected across tens of libraries. They also note the shortcomings of computer software in recommending matching scores without providing persuasive and satisfactory reasoning for researchers, as the ground truth is itself the subject of study and active research. Secondly, they mention a paleographic classification tool that recommends matching styles and dates with a given writing fragment. According to them, it seems like paleography researchers are interested in the why of recommender systems outputs as much as they value their accuracy.

Variational auto-encoders (VAE) [8] train a model to generate feature embeddings under the assumption that samples of a class should have the same embeddings. In [9], given the intra-class variance such as illumination and pose, the authors challenge that assumption. They believe that minimizing a loss function risks over-fitting on training data by ignoring each class's essential features. Moreover, by minimizing the loss function, the model could learn discriminative features based on intra-class variances. Also, they illustrate how the model struggles to generalize as samples from different classes but with the same set of intra-class variances cluster at the central part of the latent space. In addition to the KL-divergence [10] and reconstruction loss terms as in prior work, they add two metric-based loss terms. One of the new terms helps minimize the distance between samples of the same class in the presence of intra-class variations. The other new loss term prevents intra-class variances, which different classes might share, overpower essential features representing representational value.

Their framework, deep variational metric learning (DVML), disentangles class-specific discriminative features from intra-class variations. Furthermore, per their claim, it significantly improves the performance of deep metric learning methods by experimenting on the following datasets: CUB-200-2011, Cars196, and Stanford Online Products. In this work, we sample from the latent space by calculating the Kaiming-normal function, also known as He initialization [11], and we use that as epsilon to relate the mean and variance of the data distribution.

In [12], the authors focus on the problem of symbol spotting in graphical documents, architectural drawings in particular. They state the problem in the form of a paradox, as recognizing symbols requires segmenting the input image. The segmentation task should be done on a recognized region of interest. Furthermore, they want a model that works on digital schematics and scanned documents where distortions and blurriness are natural. Moreover, they also aim to build an indexing system for engineers and designers who might want to access an old document in an extensive database with a given symbol drawing that could only partially describe the design. Having those considerations in mind, the authors then propose a vectorization technique that builds symbols as a hierarchy of more basic geometric shapes. Then, they introduce a method for tiling the input document picture in a flexible and input-dependent manner. Their approach approximates arcs with low poly segments [13,14], and puts constraints on subsets of line segments such as distance ratios, angles, scales, etc. This way, they can model slight variations in the way that symbol queries build full representational graphs.

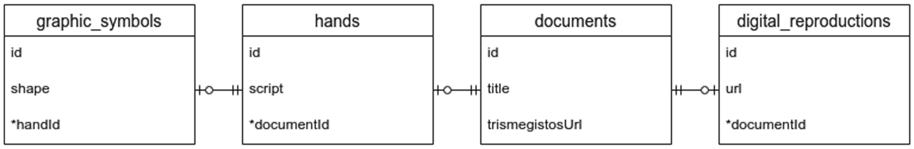

Fig. 1. Related tables in the NOTAE database

3 Data Preprocessing

3.1 Scraping Public Databases

With their expertise and knowledge of the domain, the NOTAE curators have gathered a database to find documents, hands, symbols, and digital reproductions, among much other useful information. The database tables are connected as a knowledge graph [15] (see Fig. 1). For example, a symbol present inside a script has an associated hand. The hand, in turn, comes from a document with an identification number. Then, we can get the digital reproduction of where the symbol comes from using the document ID.

3.2 Cleaning Duplicates

One of the implicit assumptions in dataset design is that sample images are unique. Scraped data is not clean, and it is likely to have duplicates. Web pictures come from different sources with different sizes and compression algorithms for encoding/decoding. So, comparing an image against the rest of the dataset to determine if it is a duplicate will not work most of the time. Using coarser features instead of raw pixels can destroy many trivial details that are not noticeable to the human eyes. Moreover, it is observed to be the case that some digital reproductions of the same picture have been ever so slightly cropped across public databases. So, the features need to be invariant under minor variations in data distribution but different enough between two unique pictures. The difference hash (dHash) algorithm [16] processes images and generates fixed-length hashes based on visual features. dHash has worked outstanding for our use case. In particular, generating 256-bit hashes and then using a relative Hamming distance threshold of 0.25 detects all duplicates. Among duplicate versions of scraped data, we chose the one with a higher resolution. By comparing image hashes against each other, we managed to clean the scraped data and thus create two datasets of unique samples such as graphic symbols and digital reproductions.

Figure 2 in particular shows the difference with respect to the quality of obtained results.

3.3 Binarization

Digital reproductions contain various supports such as papyrus, wooden tablets, slate, and parchment. In addition, due to preservation conditions and the passage of time, parts of the documents have been lost, and we deal with partial

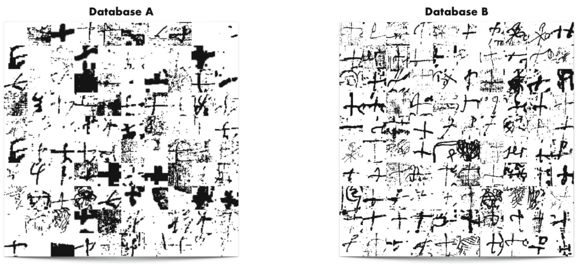

Fig. 2. The effect of the proposed solution. On the left the result without cleaning duplicates, on the right after the cleaning operation.

observations of ancient texts and symbols. Accordingly, a pre-processing step seems necessary to foreground the handwritten parts and clear the background of harmful features and noise. Then, the issue of what threshold works best for such a diverse set of documents surfaces. In that regard, we follow the prior work [1] and hand-pick one value out of the five prepared threshold values that are input dependent. We find the first two of the threshold values by performing K-means clustering on the input image and then choosing the red channel, which is the most indicative value. Next, we calculate the other three thresholds as linear functions of the first two (taking the average, for example).

Template matching works on each color channel (RGB) separately, and so it returns three normalized correlation values. Consequently, the proper peak-finding function should take the average of them in order to find the location of the most probable box (see further in Sect. 4.2 for more on peak-finding in template matching). However, since document pictures have a wide range of supports with various colors and materials, using color images is optimal, whereas binary images work the best. First, we remove the background using the selected threshold value. Next, we apply the erosion operator to remove noise and the marginal parts further. Finally, we fill the foreground with the black color to get the binary image. In our experiments, the binarization step has proven to be at least an order of magnitude more effective in reducing false positives, compared to when we tried color images.

3.4 Dataset Design

The simple baseline begins with the binarization of document pictures and template matching using the NOTAE graphic symbols database. These two steps make for an end-to-end pipeline already and identify graphic symbols with a given picture (see Fig. 3.) Next, we split our preferred set of unique binarized digital reproductions into three different subsets: train, validation, and test. The partitioning ratio is 80% training data and 10% for each test and validation subsets.

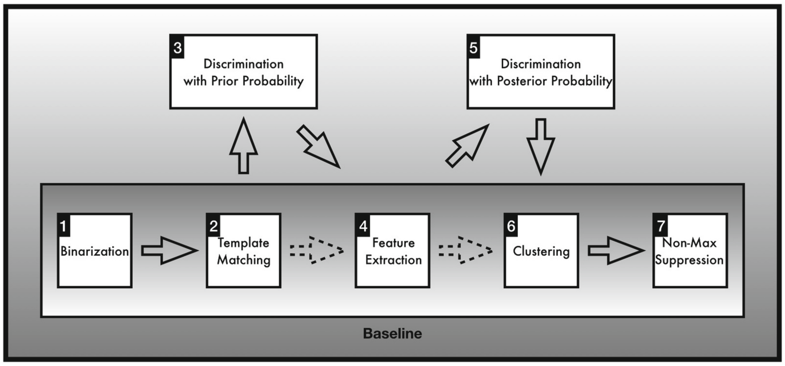

Fig. 3. Identification pipeline

3.5 Initial Symbol Clustering

As discussed in the introduction, in this new version of the annotation tool we moved from DBSCAN to OPTICS for symbol clustering. A description of how OPTICS forms denser clusters follows. First, it defines core-distance as the minimum distance within the Eps-neighborhood of an object such that it satisfies the MaxEps condition. In general, core-distance is less than MaxEps, and that is why there is a MaxEps rather than a fixed Eps in OPTICS. Then, it uses core-distance to define the reachability score as a function of one object concerning another. The reachability of object o with respect to a different object p is defined to be the maximum between either of two values: the core-distance of o or the distance between o and p. Reachability is instead not defined if objects o and p are not connected. Using a cluster expansion loop with the given core distances, OPTICS can reorder database objects between subclusters and superclusters where cluster cores come earlier and noise later. Object ordering plus reachability values prove to be much more flexible than a naive cluster-density condition in the way DBSCAN works.

4 Modeling Approach

Our target is to determine very particular graphic symbols in a digital reproduction and find their positions as smaller rectangles inside the picture frame. The NOTAE database supplies the templates we look for, so the simplest possible model can be a template matching algorithm. It takes a picture and a set of templates as inputs and returns a set of bounding boxes and the confidence scores assigned to each one of them as outputs. Then, one could select the final predictions from the top of the boxes sorted based on their scores. However, in practice, we observed that the simple model also returns too many false positives, bounding boxes with relatively high confidence scores but contain non-symbols. Moreover, the rate of false positives increases as a linear function of the database size. This inefficiency in naive template matching poses a problem since the NOTAE system design relies on the growth of the database for making its suggestions brighter. So, template matching is a simple and fast model for identifying graphic symbols in a document picture, but it has relatively limited precision.

In the identification pipeline, the template matching step is done for every graphic symbols database object. For one object (template), the algorithm returns a field of correlation densities over the input document picture, as many as the number of pixels in the given picture. So we select the one with the maximum score as the final match. Also, template matching uses five different sizes of each object. The scales range from 5% of the picture width up to 20% because that is about the size of the symbols in documents. Hence, the first step of the pipeline produces five bounding boxes per database object.

After template matching is over, we can recover some precision by way of updating confidence scores. Fast template matching is possible by transforming visual data from the spatial dimension to the frequency dimension. One can ignore some high-frequency features to speed up the process and then transform the results back to the spatial dimension. In [17] Fourier transform does so to reduce computation complexity. However, once the algorithm has queried the database and is done with its prediction process, then we can afford to update the confidence scores using a more computationally complex approach that would be quite infeasible right from the beginning. We engineer visual features for both database objects and regions of interest (ROI) for that purpose, as template matching predicts. Suppose u and v are two such features extracted using a method of our choice (u represents a template while v represents an identification ROI, for example.) If we find the lengths of these feature vectors then it becomes easy to see how similar they are:

$$correlation = \frac{< u, v >}{|u| \cdot |v|},$$

$< \cdot, \cdot >$ denotes the inner product on the vector space of features, $| \cdot |$ denotes the length of a vector and the correlation is in the closed interval $[-1, 1]$.

Due to reasons that will be discussed later in this section, we can build features in a particular latent space to preserve the save the same metric from the previous step. In fact, we propose to build a discriminator that uses the correlation between features to update the prediction probabilities and prune away false positives.

We already identified potential candidates for graphic symbols in a document picture, then discriminated against some of them based on engineered features, and finally, filtered outliers based on size. However, all those steps pertain to more individual and local symmetries rather than considering what an ensemble of identifications has in common. That is where clustering and unsupervised classification comes into play and further reduce false positives. Using the same engineered features, be it histogram of oriented gradients (HOG) or learned embeddings, a clustering algorithm can group the identified symbols into one cluster and label the rest of the identifications as noise. In this last major step to improve the results, global symmetries are the main deciding factor as to whether a bounding box should be in the graphic symbols group or not. In the clustering step, individual boxes relate to each other via a distance function. Setting a minimum neighborhood threshold, clusters of specific densities can form, as discussed in the previous section. At the end of every promising identification pipeline, they apply a non-maximum suppression algorithm. In overlapping bounding boxes, those with lower confidence scores are removed in favor of the top match. See Fig. 3 for a representation of our identification pipeline.

4.1 Updating Identification Probabilities

Suppose T, F, M and D be events: T as the event that a box is true positive, F as the event that a box is false positive, M as the event that the template Matching model labels a box as true positive, and D is the event that the Discriminator model labels a box as true positive. Also, suppose MD be the event that both the template Matching and Discriminator models label a box as true positive.

Please note that the sum of prior probabilities should be equal to one.

$$P(T) + P(F) = 1.$$

Next, let's appeal to the Bayes theorem. In the symbol identification task, write down the posterior probabilities of such events occurring:

$$P(T|MD) = \frac{P(MD|T) \cdot P(T)}{P(MD|T) \cdot P(T) + P(MD|F) \cdot P(F)},$$

or, in an equivalent way:

$$P(T|MD) = \frac{P(MD|T) \cdot P(T)}{P(MD)}.$$

First, the template matching model acts on the graphics symbols database. The input document picture is implicit here as it stays constant throughout the pipeline. Then, the template matching model returns one match per pixel in the document picture. A suitable cut-off threshold as a hyper-parameter will reduce the number of symbols based on the confidence scores. So, we only select the top match for each database object (template). Next, the discriminator model acts on the top matches. Furthermore, thus the template-matching model and the discriminator model participate in a function composition at two different levels of abstraction. In this composition, template matching works with raw pixels, whereas discrimination works with high-level embedding vectors.

$$\mathit{updated\ scores} = \mathit{Discriminate} \circ \mathit{Match}(\mathit{database}),$$

where $\circ$ denotes the function composition by first applying Match and then $\mathit{Discriminate}$ on the database, object by object.

If we assume that events M and D are independent (or slightly correlated), then we can say that they are conditionally independent given T.

$$P(MD|T) = P(M|T) \cdot P(D|T)$$

Therefore the updated probability will be:

$$P(T|MD) = \frac{P(M|T) \cdot P(D|T) \cdot P(T)}{P(MD)}$$

Performing some computation to simplify the posterior probability:

$$P(T|MD) = \frac{P(M|T) \cdot P(DT)}{P(MD)}$$

$$P(T|MD) = \frac{P(M|T) \cdot P(T|D) \cdot P(D)}{P(MD)}$$

$$P(T|MD) = \frac{P(M|T) \cdot P(T|D)}{Q(1,2)},$$

where $Q(1,2) = \frac{P(MD)}{P(D)}$.
Since $1 = P(T|MD) + P(F|MD)$, therefore:

$$1 = \frac{P(M|T) \cdot P(T|D) + P(M|F) \cdot P(F|D)}{Q(1,2)}.$$

Now, it is obvious that

$$Q(1,2) = P(M|T) \cdot P(T|D) + P(M|F) \cdot P(F|D)$$

And that conclusion implies that the updated probability is as follows:

$$P(T|MD) = \frac{P(M|T) \cdot P(T|D)}{P(M|T) \cdot P(T|D) + P(M|F) \cdot P(F|D)} \tag{1}$$

Q: Where do we get the value $P(M|T)$ from? **A:** The Average Recall (AR) of the template matching function gives the value for $P(M|T)$. It is the probability that the fast template matching algorithm identifies a symbol given that it is a true symbol. **Q:** Where do we get the value $P(T|D)$ from? **A:** The Average Precision (AP) of the discriminator function gives the value for $P(T|D)$. It is the probability that a symbol is true given that the discriminator model has labeled it positive. **Q:** What does $P(M|F)$ mean? **A:** It is the probability that the template-matching model identifies a symbol given that it is negative. **Q:** What does $P(F|D)$ mean? **A:** It is the probability that a symbol is false given that the discriminator model has labeled it positive.

The template matching model produces potential bounding boxes in a digital reproduction with the graphic symbols database. Next in the pipeline, we use an attention mechanism to discriminate for the boxes that are more likely to be true with the given digital reproduction. The discriminator is indifferent to the location of the query symbol and only cares about whether the matching box is

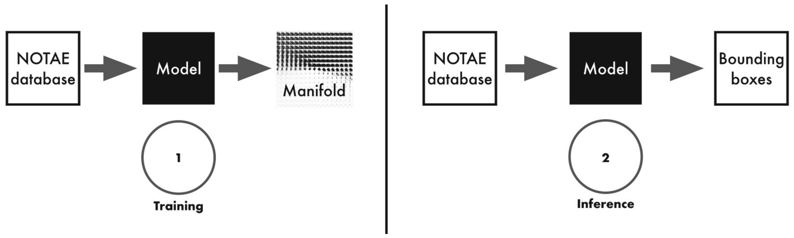

Fig. 4. Filtering noise with low overhead as the inference has lower latency.

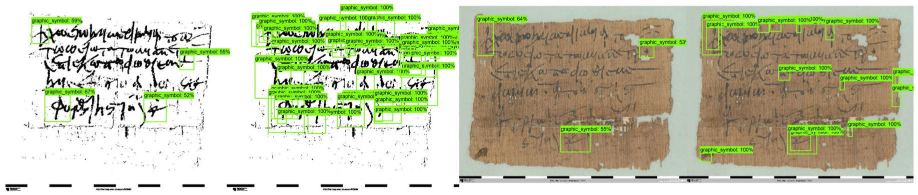

Fig. 5. Identifications on the left side and ground truth on the right.

similar to it or not. Therefore, the discrimination step is an image classification task in essence. Figure 4 shows how the two steps, symbol matching, and classification, share the same database objects. The discrimination model, step 5 in Fig. 3, introduces a posterior probability function $P(T|D)$ and assigns to every box a value from -1 to 1. The sequential update of information now changes first to consider event M and then update with event D.

Finally, we can normalize the discrimination confidence score by adding one unit and dividing it by 2 to get a correct probability value in $[0, 1]$, formally known as an affine transformation. Next, we use it to replace the score from the template matching step. The posterior probability $P(T|MD)$ is correlated to the scores coming from both steps: template matching and discrimination. The rest of the pipeline will work the same (see Fig. 5). In the following subsection, we are going to use this result to focus on the feature engineering that maximizes $P(T|D)$, that is, the true positive rate has given the second event, discrimination.

4.2 Latent Clustering

By now, we have established the probability that a graphic symbol is true given that the discriminator model has labeled it positive works based on a correlation between the source symbol and the target identification. As indicated, we need to look more closely at the choice of metric and distance functions. Because the more accurate we are in determining the actual distance between two objects, the better we can reason about if the two objects in question are related and why.

Suppose the distribution of the graphic symbols database is described by manifold M. Here, we do not assume any structure beyond that there is a probability $p(x)$ that we discover a given object x in it. Except for maybe a smooth frame at x for applying convolutional filters. Since it is a complex manifold, as is the case with most objects in the real world, it could be intractable to explain with a reasonable amount of information. Therefore, we defer to a latent manifold $\tilde{M}$ which is finite-dimensional and could potentially explain the most important aspects that we care about in objects from M. What we need here is a map, such as ϕ, from manifold M into manifold $\tilde{M}$ such that our choice of metric in the latent manifold $\tilde{M}$ results in a predictable corresponding metric in the original manifold M.

Accordingly, we could reason unseen objects knowing that for every input in the domain of graphic symbols manifold, there will be a predictable output in the co-domain of the latent manifold. Predictable in the sense that our metric in the latent space would work as expected. In this context, the encoder model plays the part of the inverse of a smooth map. It maps objects from the pixel space onto the latent space.

$$Encode : pixel\ space \mapsto latent\ space$$

Suppose that p and v are vector representations of an ROI (inside a document picture) and a graphic symbol, respectively. Next, we define a few smooth maps for computing the probabilities of our modeling approach.

$$P(M|T) := \arg\max_{i,j} Match(p_{i,j}, v),$$

given by

$$Match(p_{i,j}, v) = p_{i,j} * v \ = \ <\hat{p_{i,j}}, \hat{v}>,$$

The inner product between normalized elements from the template matching sliding window at (i, j) of the input picture and normalized database elements makes sense if both vector spaces are of the same actual dimension. Here i and j are the maximum arguments of the term on the right, which reflect our process of selecting the top match based on confidence scores. We take the maximum value among the inner products so that it corresponds to the most probable location in the document picture.

$$Discriminate(p_{i,j}, v) := P(T|D),$$

given by

$$Discriminate(p_{i,j}, v) \ = \ <Encode(p_{i,j}), Encode(v)>.$$

For taking symbols from the pixel space to the latent space (embeddings), we can use the encoder part of a variational auto-encoder (VAE) model. We trained a VAE model on the graphic symbols database in a self-supervised manner to get the embeddings of unseen symbols. The model uses a deep residual architecture (ResNet18 in Fig. 6) [18] and the bottleneck in this neural network would be the latent layer where the features are sampled from.

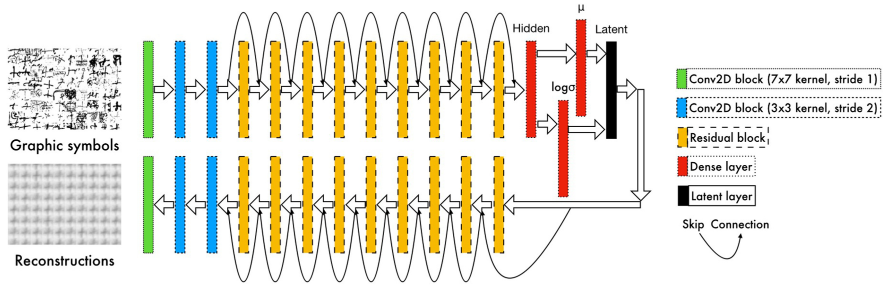

Fig. 6. The VAE encodes graphic symbols, upper row, and decodes them, lower row.

4.3 Optimization Objective

The loss function should look like the following equation since according to equation (1) from earlier in this section, we want the training objective to minimize $P(M|F)$ and $P(F|D)$ while maximizing $P(M|T)$ and $P(T|D)$ (up to a proxy function.)

$$\mathcal{L} = \alpha \cdot reconstruction + \beta \cdot [KL\ divergence],$$

where, α and β are hyper-parameters in $\mathbb{R}$. The reconstruction loss term above is the mean square error of the input image and its decoded counterpart. A point in the latent space should be similar to a sample from the normal distribution if we want the model to learn a smooth manifold. When the latent distribution and the normal distribution are the most alike, the KL-divergence loss term should be approximately equal to zero. Adding the relative entropy loss term to the loss function justifies our assumption on the learned manifold being a smooth one.

5 Quantifying Model Performance

In order to perform evaluation, it is helpful to imagine the annotation tool as a generic function that maps elements from an input domain to the output. In our case, in particular, we want to map tuples of the form `(document_picture, symbol)` to a bounding box array. As evaluation method, we employed mean Average Precision (mAP) [5], which outputs the ratio of true symbols over all of the identified symbols.

Additionally, we annotated the dataset using the Pascal VOC [6] format in order to evaluate the system using well-established tools.

We used an object detection model by the moniker CenterNet ResNet50 V2 512×512 [19], which was pre-trained on the MS COCO 17 dataset [20]. It is a single-stage detector that has achieved 29.5% mAP with COCO evaluation tools. In order to repurpose it for our work, we generated annotations for 183 unique digital reproductions using our pipeline and then fine-tuned the object detection model on the annotated data. It is not so easy to measure how helpful our

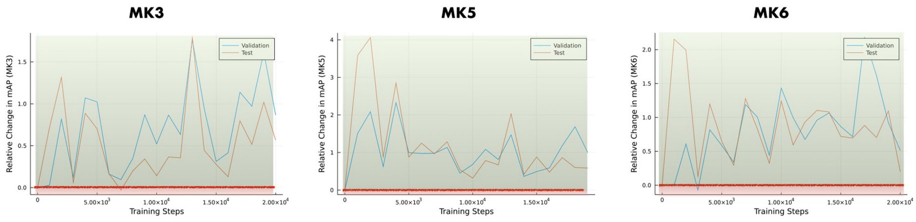

Fig. 7. Improvements in mAP validate the pipeline. The horizontal line is the baseline.

approach is using offline training as the model outputs have to be first justified by the model and then interpreted and validated by domain experts. Therefore, the evaluation protocol in this section merely focuses on the coherency and accuracy of the results. The different variants of the identified symbols datasets are partitioned with different ratios and random seeds, so they also serve as a multi-fold testing apparatus. This section considers improvements in precision since it is normal for symbol spotting methods to perform well in terms of recall.

In the spirit of an iterative pipeline design, we generated seven different identified symbols datasets. Using roughly the identical digital reproductions and graphic symbols validated our data and modeling approach. For the baseline, we bypassed steps 3 and 5 in the pipeline (Fig. 3) and also used HOG features to have a model as close as possible to prior work [1]. Next, we used the encoder with a binary classifier and generated mark 3. This modification puts steps 3 and 5 of the pipeline into effect. We have compared the evaluation results of MK3 with that of the baseline model, which is about double the precision, suggesting the effectiveness of the discrimination step in improving the true positive rate. Mark 5 follows the same architecture as mark 3. However, it adds discrimination based on bounding box area and foreground density after discrimination with posterior probability, which further improved the results (compare the third and the fourth columns in Table 1).

Then, we modified the pipeline by training the encoder and hard-wiring a discriminator function to calculate posterior probabilities using cosine similarity. The object detection model trained on the identified symbols mark 6 dataset yielded new evaluation results. MK6 annotations look much better than their predecessors in a qualitative way. Interestingly, MK6 annotations seem to generalize well over different scales (see the bottom image in Fig. 5), as it is the first dataset among the series to identify small symbols as well.

The evaluation of MK3 was when we picked up on the trend that we could gain model performance by focusing more on the data rather than the model. By manually labeling the binarized version of the graphic symbols database, we excluded almost half of the objects as non-symbols to get to a dataset of 722 graphic symbols. So, we should attribute some of the improvements over the baseline model to the data cleaning process. That process called for training the auto-encoder model again with the clean data. Table 2 brings the final improvement rates over the baseline with MK3, MK5, and MK6. We added the validation

Table 1. Symbol identification performance results related to the identified symbols datasets: the baseline, mark 3, mark 5 and mark 6 (all evaluated on their respective test sets at training step 2000.)

Metric	Baseline	MK3	MK5	MK6	Comment
AP	**0.011**	**0.022**	**0.048**	**0.028**	AP at IoU = .50:.05:.95 (primary metric)
AP@IoU = .50	*0.047*	*0.101*	*0.148*	*0.102*	AP at IoU = .50 (PASCAL VOC metric)
AP@IoU = .75	0.000	0.003	0.015	0.012	AP at IoU = .75 (strict metric)
AP@small	0.009	0.000	0.000	0.022	AP for small objects: area < 322
AP@medium	0.016	0.021	0.028	0.033	AP for medium objects: 322 < area < 962
AP@large	0.000	0.051	0.083	0.025	AP for large objects: area > 962
AR@max=1	0.003	0.008	0.019	0.006	AR given 1 detection per image
AR@max=10	0.024	0.053	0.071	0.033	AR given 10 detections per image
AR@max=100	0.083	0.126	0.158	0.085	AR given 100 detections per image
AR@small	0.075	0.000	0.000	0.021	AR for small objects: area < 322
AR@medium	0.102	0.130	0.094	0.088	AR for medium objects: 322 < area < 962
AR@large	0.000	0.076	0.246	0.086	AR for large objects: area > 962

Table 2. Guiding the identification pipeline design by measuring the relative change in mAP, dataset to dataset.

Identified symbols dataset	mAP relative change (valid.)	mAP relative change (test)
Mark 3	69%	51%
Mark 5	**102%**	**119%**
Mark 6	80%	86%

set to Table 2 and Fig. 7 in order to show that our approach is not sensitive to the choice of hyper-parameters. Because test results are strongly correlated with validation. MK5 performs at least twice better than the baseline, and so it is a good candidate to replace it as a new baseline. So, we expect it to perform as well on unseen data. The following relation allows us to calculate the relative change in mAP:

$$relative\ change\ in\ mAP = \frac{proposed\ mAP - baseline\ mAP}{baseline\ mAP} \cdot 100\%.$$

Table 2 presents the relative change in mAP while Table 1 puts the main challenge metric into its proper context. As an illustration, mark 5 outperforms the baseline by 102% and 119% in the validation and test subsets, respectively.

6 Conclusions

In this paper, we have shown how the detection scores provided by fast template matching can be the key to annotate extensive databases in an efficient way. In previous work, the idea is that the bigger the database grows, the more brilliant the symbol engine gets. However, more significant databases also cause more false positives due to inefficiencies in template matching. In this work, we first removed duplicates and then hand-picked binarized versions of the scraped images. Then, a series of identified graphic symbols datasets to validate our hypotheses on data and modeling was designed. The confidence scores of symbol matching using a binary classifier where the discriminative features sampled from the latent space as an approximation of the original space updated. Next, we justified our assumptions about the effectiveness of our distance function in providing a metric for filtering false positives. Not only we managed to recover results from the baseline model, but also there was a significant improvement in model performance across validation and test subsets. Even though many false positives make it through the final stage of the pipeline, we illustrated how a trained detection model generalizes well on the annotated data and why it solves the paradox of segmenting for spotting or spotting for segmentation. Our approach applies to intelligent assistants for database curators and researchers. In a domain where labeled data is scarce, we have adopted evaluation metrics that enable researchers to quantify model performance with weakly labeled data.

The fact that modifications to the pipeline have a clear impact on model performance regarding the relative change in mAP helps define a reward function. Based on the behavior of model performance, we believe that the relative change in mAP could introduce a new term to the loss function. In future work, we would like to see agents that can use this metric to fill in the gaps between sparse learning signals from domain experts during interactive training sessions.

References

1. Boccuzzi, M., et al.: Identifying, classifying and searching graphic symbols in the NOTAE system. In: Ceci, M., Ferilli, S., Poggi, A. (eds.) IRCDL 2020. CCIS, vol. 1177, pp. 111–122. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-39905-4_12
2. Ankerst, M., Breunig, M.M., et al.: OPTICS: ordering points to identify the clustering structure, ACM SIGMOD Rec. (1999). https://doi.org/10.1145/304181.304187
3. Ester, M., Kriegel, H.P., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, pp. 226–231. AAAI Press (1996)
4. Huang, J., Rathod, V., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. In: CVPR (2017)
5. COCO detection evaluation. https://cocodataset.org/#detection-eval. Accessed 17 Mar 2021
6. Everingham, M., Winn, J.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Development Kit (2012)

7. Wolf, L., Potikha, L., Dershowitz, N., Shweka, R., Choueka, Y.: Computerized paleography: tools for historical manuscripts. In: 18th IEEE International Conference on Image Processing, pp. 3545–3548 (2011). https://doi.org/10.1109/ICIP.2011.6116481
8. Kingma, D.P., Welling, M.: Auto-Encoding Variational Bayes (2014). arXiv:1312.6114
9. Lin, X., Duan, Y., Dong, Q., Lu, J., Zhou, J.: Deep variational metric learning. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11219, pp. 714–729. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01267-0_42
10. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Statist. **22**(1) 79–86 (1951). https://doi.org/10.1214/aoms/1177729694
11. He, K., et al.: Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In: Proceedings of the IEEE International Conference on Computer Vision (2015)
12. Rusiñol, M., Lladós, J.: Symbol spotting in technical drawings using vectorial signatures. In: Liu, W., Lladós, J. (eds.) GREC 2005. LNCS, vol. 3926, pp. 35–46. Springer, Heidelberg (2006). https://doi.org/10.1007/11767978_4
13. Ramer, U.: An iterative procedure for the polygonal approximation of plane curves. Comput. Graph. Image Process. **1**(3), 244–256 (1972). https://doi.org/10.1016/S0146-664X(72)80017-0. ISSN 0146-664X
14. Douglas, D.H., Peucker, T.K.: Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature (2011). https://doi.org/10.1002/9780470669488.ch2
15. Bernasconi, E., et al.: Exploring the historical context of graphic symbols: the NOTAE knowledge graph and its visual interface. In: IRCDL 2021, pp. 147–154 (2021)
16. Krawetz, N.: Kind of Like That, In: The Hacker Factor Blog (2013). http://www.hackerfactor.com/blog/?/archives/529-Kind-of-Like-That.html. Accessed 29 June 2021
17. Lewis, J.P.: Fast Template Matching, In: Vision Interface 95, Canadian Image Processing and Pattern Recognition Society, Quebec City, Canada, pp. 120–123 (1995)
18. He, K., Zhang, X., et al.: Deep residual learning for image recognition (2015). arXiv:1512.03385
19. Duan, K., Bai, S., et al.: CenterNet: keypoint triplets for object detection (2019). arXiv:1904.08189
20. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48