Handbook of Digital Egyptology: Texts

Edited by Carlos Gracia Zamacona & Jónatan Ortiz-García

Universidad de Alcalá
EDITORIAL UNIVERSIDAD DE ALCALÁ

Con el patrocinio de la Comunidad de Madrid, Proyecto *The Earlier Ancient Egyptian Mortuary Texts Variability* (www.mortexvar.com), Programa Atracción de Talento 1 (2018-T1/HUM-10215).

Plaza de San Diego, s/n
28801 Alcalá de Henares
www.uah.es

I.S.B.N.: 978-84-18979-09-5
Depósito legal: M-34533-2021

Composición: Solana e Hijos, A. G., S.A.U.
Impresión y encuadernación: Solana e Hijos, A.G., S.A.U.
Impreso en España

7. How to compute a shape: Optical character recognition for hieratic

Bernhard Bermeitinger
Institute of Computer Science, University of St. Gallen

Svenja A. Gülden
Altägyptische Kursivschriften, Akademie der Wissenschaften und der Literatur Mainz

Tobias Konrad
Altägyptische Kursivschriften, Akademie der Wissenschaften und der Literatur Mainz

Abstract
This paper is written in the framework of the project Altägyptische Kursivschriften and presents an experiment for applying OCR to Hieratic, examining the final step of the OCR pipeline. A convolutional neural network is used to classify individual hieratic characters. The results prove that the classification of hieratic characters is in principle possible, but they also show, where improvements and combinations with other methods promise a better performance of recognition models.

Keywords
Hieratic – handwriting recognition – classification – convolutional neural network – ancient Egyptian script.

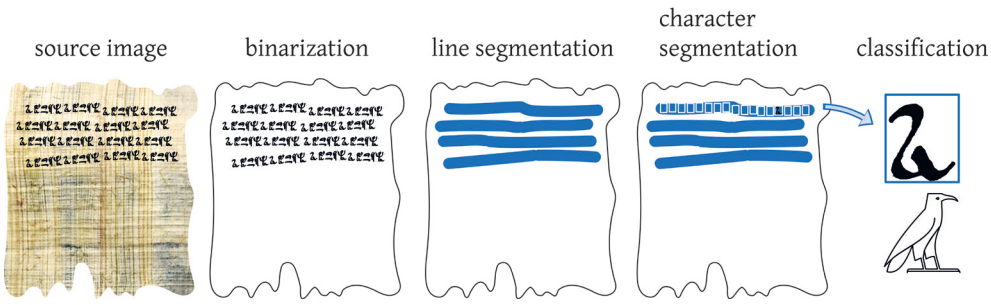
1. What is optical character recognition?

1.1. Introduction

Optical character recognition (OCR) is a technique for the conversion of textual image data to machine-readable text data. Being developed for printed media, it works quite well for font-based alphabetic characters. The pipeline for converting image

data to computer-readable text can be divided into two main sections: *preprocessing* and *recognition* (fig. 1). The *preprocessing* consists of various image manipulation techniques such as binarization.[1] With this method, it is easier to separate the text from the background. After this cleaning process, the text has to be segmented into several lines. The last step of the preprocessing is called character segmentation. This finishes the preprocessing procedure, and finally, the *recognition* step is performed, where single characters are classified.[2]

Fig. 1. Visualization of a simple pipeline for optical character recognition



For the handwritten text recognition (HTR) of a cursive script, like hieratic, one also finds terms like handwriting recognition (HWR) or handwritten character recognition (HCR). The boundaries between OCR and HTR (and other terms) are fluent and not yet well defined. Despite the inconsistent use of these nomenclatures, the main difference between OCR and HTR is that OCR techniques focus on a character level—which means the recognition step is performed on a single separated character. Due to the nature of some cursive scripts (e.g. high frequency of ligatures), HTR models have to be trained on word level. This leads to a much higher amount of training data for HTR models.[3]

---

[1] Additional optimization techniques may be performed in those cases where the image is rotated or deformed.

[2] This pipeline describes a very simple approach. However, to achieve results with a low error rate resp. recognition with high accuracy, statistical models have to be used. Modern software uses different models that are trained for specific languages or even specific textual genres. For a genre-specific approach, see for example Strange et al. 2014. This step can be added to the OCR pipeline and is called post-processing resp. error correction, see Miyagawa et al. 2019: chapter 2.

[3] A good overview of software tools and workflows can be found in Franzini et al. 2018: chapters 2 and 4. For character error rates and the effect of training see Hawk et al. 2019.

1.2. Software

The use of recognition resp. classification methods for the hieratic script have not been very common in the past. Ahead of his time was a project by Erhart Graefe in the late nineties in which he used a simple artificial neural network to detect and classify single hieratic signs within a papyrus.[4] Unfortunately, this project has never passed the stage of preparation. A similar approach was done by Stephen Quirke, who tried to identify individual signs on scans of papyri. His trials lead to considerations about computer-aided paleography and the requirements that must be fulfilled both for Egyptology and data science.[5] A short description of actual OCR applications used in Egyptological research (as of 2015) is given by Rosmorduc.[6] He only mentions projects that are not focusing on hieratic but indeed show the potentials as well as the difficulties even when OCR techniques are confronted with more normalized hieroglyphic forms.[7] Most recently, the preliminary considerations of Quirke (2010) led to the development of an application by the team of the Digital Lahun Papyri project that applies the whole pipeline of OCR on a set of hieratic papyri from Lahun.[8]

Although it has not been tested on ancient Egyptian hieratic, we would like to mention a selection of state-of-the-art software that finds applications in many actual projects—especially in the field of digital humanities. Special software for OCR tasks is, for example, *Tesseract*[9], *OCRopus*[10] (sometimes named *OCRopy*), its branches *Kraken*[11] and *Calamari*[12], and finally, a software environment called *OCR4all*[13] that merges several of these applications. Good results in HTR are achieved by the extensively used *Transkribus*.[14] It tends to be the only software available that has been developed to the stage of usability. The other software available is mostly individual studies and experiments or preparatory work.[15]

---

[4] Graefe 1999.

[5] Quirke 2010: 289–292.

[6] Rosmorduc 2015: 7.

[7] These projects are especially Franken 2013; Nederhof 2016; furthermore, three papers can be added: Arrivault – Richard – Bouyer 2005 is focusing on handwritten hieroglyphs, whereas an approach for identifying signs within pyramid and other hieroglyphic texts can be found in Elnabawy et al. 2018 and Barucci et al. 2021.

[8] Haliassos et al. 2020. As a first part, the classification step is performed only by the distinction of three labels, see Haliassos et al. 2020: 127.

[9] <https://tesseract-ocr.github.io> [accessed: 5/22/2020].

[10] <https://github.com/tmbarchive/ocropy> [accessed: 5/22/2020].

[11] <http://kraken.re> [accessed: 5/22/2020].

[12] <https://github.com/Calamari-OCR/calamari> [accessed: 5/22/2020].

[13] <https://github.com/OCR4all/OCR4all> [accessed: 5/22/2020]. For a detailed description see Reul et al. 2019.

[14] <https://transkribus.eu/Transkribus> [accessed: 5/22/2020].

[15] For the sake of completeness, it has to be mentioned that a few other programs are available, e.g. T-PEN (<http://t-pen.org/TPEN> [accessed: 5/22/2020]), the Virtual Transcription Laboratory (<http://wlt.

2. OCR for Hieratic

2.1. Challenges

The alphabet of the English language contains 26 different characters, each with two letters of different shapes (upper and lower case) plus 10 numeral digits. Therefore, any OCR software has to distinguish between 62 different shapes for classification. On the contrary, the hieratic script has more than 500 different characters,[16] many of them with a minimum of two distinctive shapes (form classes). That means the number of shapes that have to be trained for hieratic is about 10 times higher than for alphabetical scripts. Another problem for the recognition is that on the one hand, different characters can look very similar, and on the other hand, one character can have multiple hieratic form classes. The hieratic signs for  (Gardiner A47) and  (Gardiner M18) share similar shapes, while the signs for  (Gardiner G17) show both abbreviated and more detailed forms that are very different from each other.[17]

A similar intersection of classes of specific characters in Latin alphabetic (printed) script is, for example, the confusion between the shape of the letters "c", "e", and "o". This problem has been solved by incorporating word lists and character frequencies of the specific language (and genre) at a post-processing step,[18] but for ancient Egyptian hieratic, no model exists so far.[19] This situation is further complicated by the specific nature of hieratic as *scripta continua* so that there is no natural word separator recognizable by any software.[20]

2.2. A new sign list for hieratic signs

The ancient Egyptian scholars already organized the hieratic signs in lists. They compared them to the hieroglyphs and sometimes described them.[21] Modern

---

synat.pcss.pl/wlt-web/index.xhtml> [accessed: 5/22/2020]), and Transcript (<https://www.jacobboerema.nl/Transcript> [accessed: 5/22/2020]), but these are not as popular as Transkribus (this list of software derives from <http://dhmuseum.uni-trier.de/node/49> [accessed: 5/22/2020]).

[16] Verhoeven 2015: 34.

[17] Van der Moezel 2018: 63–72; figs. 5–6 and 9.

[18] See also note 5.

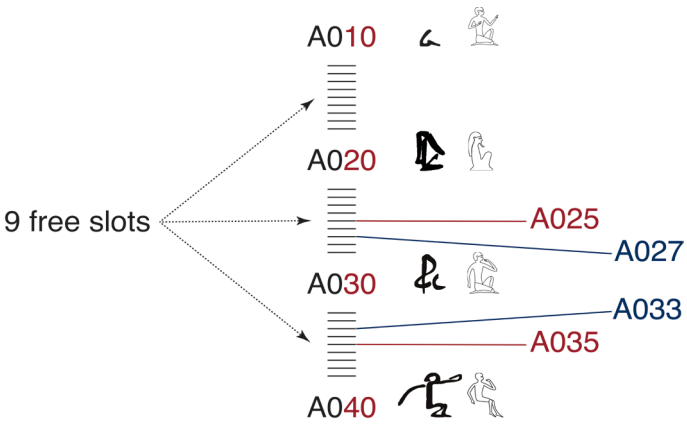
[19] For a probability-based model that has already been tested on encoded hieroglyphic texts see Nederhof – Rahman 2017.

[20] Another challenge to solve is the diversified orthography of Egyptian writing and the placement of signs above or below each other. This also increases the amount of training data. Finally, there are some problems that result from the materiality of the writing surface. In some cases, the text bearing object is in a poor condition and the ink is very faded. As a result of that, a simple binarization of the input image is not sufficient for preprocessing and more elaborated algorithms have to be used to make the ink visible again. Another property that is impeded by the quality of the writing surface is the noisy structure of the papyrus sheet. For the latter see e.g. Shomaker 2019: fig. 10.2.

[21] Gülden 2018: 83–84

researchers, starting with Champollion, also compiled lists of the ancient Egyptian signs.[22] In his *Hieratische Paläographie*[23], which still is a standard work for hieratic studies, Georg Möller organized the signs in taxonomic classes and introduced a new numbering system for the hieratic (and the hieroglyphic) signs.[24] Gardiner published in his *Egyptian Grammar* a *Sign-list*—again with reworked taxonomic classes and a new encoding system.[25] Although this *Sign-list* was developed (only) as a pedagogical tool, most of the modern hieratic paleographies follow his classification. But using the Gardiner codes for hieratic signs is problematic—many hieratic characters find no match in Gardiner's *Sign-list*. This leads to individual numbering systems in various hieratic paleographies and/or the use of additional codes developed for the extended library of *Hieroglyphica* or *JSesh*. Joining these different codes in a hieratic sign list is almost impossible and one of the reasons why a new classification and numbering system for hieratic graphemes (see fig. 2)[26] is currently set up by the project *Altägyptische Kursivschriften. Digitale Paläographie und systematische Analyse des Hieratischen und der Kursivhieroglyphen* (AKU)[27].

Fig. 2. The new AKU numbering system has 9 free slots between the entries that allow adding new graphemes in a later stage



[22] For an overview and short description of the various systems see Gülden 2018: 84–87.

[23] Möller 1927a, 1927b, 1936a, 1936b.

[24] Möller 1927a: V. The Möller codes are still in use but mainly as a supplement to the Gardiner codes.

[25] Gardiner 1957: 438–548.

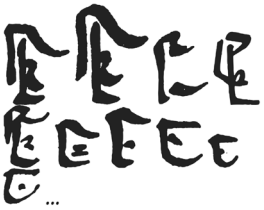
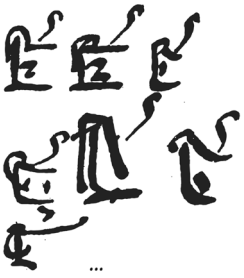
[26] See Gülden – Verhoeven 2017: 3 and 6; Gülden 2018: 87; van der Moezel 2018: 52–54. Please note: the AKU-numbers are still object of change.

[27] The project is hosted at the Academy of Sciences and Literature, Mainz and is a collaboration between Johannes Gutenberg-University Mainz and Technical University of Darmstadt. It combines traditional Egyptological research with methods from the field of digital humanities. For actual news see <https://aku.uni-mainz.de> [accessed: 5/22/2020] and <https://aku.hypotheses.org> [accessed: 5/22/2020].

Other reasons for developing a sign list for the hieratic script are much more important: the discrepancies between the hieratic and hieroglyphic script that one can observe. One example, that hieratic signs with similar shapes can stand for different Hieroglyphs is described above. Another example is clearly illustrated in fig. 3. To transcribe the two graphemes AKU-A150 and AKU-A155, the sign Gardiner A53 is used. But the hieratic signs show a significant difference, compared to AKU-A150, the sign AKU-A155 clearly shows an additional attribute. Not only to document this but also to enable a (digital) analysis, the AKU project records the signs under separate numbers.

However, to allow a search for the various codes in hieratic paleographies and hieroglyphic sign lists, the AKU project has set up concordances and also links to online resources such as e.g. the *Thot Sign List*.[28]

Fig. 3. Examples of two different hieratic graphemes that share the identical hieroglyphic number when transcribed into hieroglyphs

AKU no.	hieratic examples	associated numbers		hieroglyph
A150		Möller	10	
		Gardiner	A53	
		Thot Sign List	TSL_1_832	
		...	...	
A155		Möller	10	
		Gardiner	A53	
		Thot Sign List	TSL_1_832	
		...	...	

3. The AKU project's approach

From this initial point, the AKU-project has created an experiment based on the acquired data that will give us insight both in the challenges of implementing

[28] <http://thotsignlist.org> [accessed: 5/22/2020].

OCR on the hieratic script and the first steps in analyzing the structure of Egyptian handwriting.[29] The following experiment deals only with the last step of the OCR pipeline (classification) and serves as a feasibility study for further research. Approaches for the other steps have to be developed separately.

However, the AKU project is building up a paleographic research tool to collect and analyze the repertoire of the hieratic script.[30] The project is neither focusing on OCR nor annotating at word level—the idea for this experiment was to build a classifier on the basis of the data, that has been collected for our research. The data originates on one hand from already published (printed) paleographies, and on the other hand from digital facsimiles (drawings) based on scans or photographs of the inscribed objects. In addition to this, we receive data in various formats from our colleagues.[31] For a digital analysis, the material, therefore, needs to be prepared in different ways.

3.1. Data generation

3.1.1. *Retro digitization*

Since the AKU project also uses data from already published paleographies—above all Georg Möller's *Hieratische Paläographie*[32]—retro digitization was, in the beginning, one of the main steps of the procedure. For already published material, we scan the paleographic tables with a resolution of 1200 pixels per inch (PPI) and store the files as *Tagged Image File Format* (TIFF). The individual signs are cut out from these images manually and to enhance contrast histogram equalization is performed. The images are cropped automatically to the outer boundaries of the hieratic signs to remove unnecessary white areas. At this stage, the images contain a high amount of unwanted noise. This is the result of the scanning and enhancement processes that intensify the structure of the paper of the published paleographies. To remove this noise a binary opening is applied to the images.[33] Afterward, the images are transformed into *Scalable Vector Graphics* (SVG)[34] and converted back to raster

---

[29] The experiment was realized in collaboration with Bernhard Bermeitinger (Institute of Computer Science, University of St. Gallen).

[30] This paleography will be published as an online database, see Gülden – Verhoeven 2017; Gülden et al. 2020.

[31] For different paleographic formats see Gülden 2018: 99–102.

[32] Möller 1927a; 1927b; 1936b; 1936a.

[33] For these easy image processing tasks, we use the software library *Scikit-Image* for *Python*, see van der Walt et al. 2014 and <https://scikit-image.org> [accessed: 5/22/2020].

[34] Conversion of raster graphics into vector format is performed by the software *Potrace* (<http://potrace.sourceforge.net> [accessed: 5/22/2020]). For technical details see also Selinger 2003; for the file format SVG, especially version 1.1 see World Wide Web Consortium 2011.

graphics to get a uniform appearance of the hieratic signs and—more important—to remove the differences caused by the quality of the paper or the printing.

3.1.2. *Digital drawings*

The AKU project creates digital hieratograms[35] by drawing vector graphics based on digitized originals (e.g. papyri, ostraca, or dipinti) in a 1:1 ratio. The digital photos or scans should have a high resolution so that they can be displayed by a zoom level of at least 1200 %.[36] The signs are saved as individual vector graphics (SVG).

Since the XML code of the file describes the vector paths, the height, and width of the signs, etc. they are particularly suitable for digital analysis. Due to the fact that OCR is based on raster graphics, we convert the vector graphics (SVG) into raster graphics (TIFF).

3.2. Experimental setup

As outlined above, the last step for OCR is to classify a sign—or to assign a label to any unknown sign that is given as an input image. For this task, we use an artificial neural network. The implementation of such networks can be divided into two different steps: training and validation (or testing). In the first step, the network is trained on annotated data and learns which label (in this case the AKU number)[37] corresponds to a specific hieratic sign. The model created by the training process can then be evaluated with another part of annotated data, that has not been used for the training to measure the performance of the classification.[38]

3.2.1. Data distribution

Preliminary experiments have shown that each label must be represented by at least 50 images to have a meaningful classification. Obviously, this reduces the amount of available training data. We run the classification experiment with two different datasets.

---

[35] For this and other terms, the AKU project uses see Gülden 2018: 103–104.

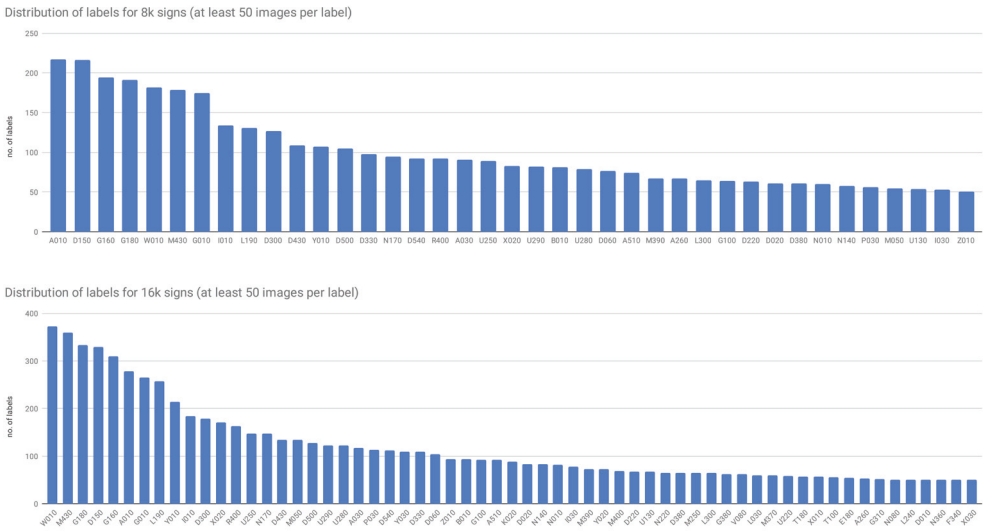
[36] This can be achieved by scanning e.g. a papyrus sheet with a resolution of 600 PPI or more. In case there was no possibility to get high-resolution images other values worked well, but with a more limited size, the drawing process becomes more difficult and less accurate. For the process of drawing see Gülden 2018: 95–98; Gülden et al. 2020.

[37] The label does not have a meaning for the neural network—one could also use the traditional Gardiner-numbering system, but this system is not suitable to describe hieratic signs for digital analysis (see above).

[38] The annotated data for training and validation is generally called ground truth.

The first dataset contains about 8,000 and the second 16,000 individual hieratic signs with an unbalanced distribution (fig. 4). This different distribution causes a variable number of labels and signs per label that can be trained. In dataset *8k*, there are 39 labels present (~3900 single signs in total), whereas dataset *16k* contains 61 labels (~7350 single signs in total). The occurrences of labels with more than 50 single instances give us an impression of most frequent signs documented in hieratic paleographies but do not depict the real distribution of sign frequencies in hieratic script.

Fig. 4. Distribution of labels for datasets 8k and 16k



3.2.2. *Software and setup*

The experiment is executed in Google's *Colaboratory Environment*[39] which is a cloud-based environment for easy code execution. The programming language is *Python*[40] which is the underlying execution engine for Google's version of *Jupyter Notebooks*[41]. The neural network and data ingestion pipeline are programmed in Google's *TensorFlow* framework.[42] The Google Colab environment has the advantage of optionally enable execution on specifically built hardware which decreases the runtime of the training by a factor of at least two orders of magnitude.

---

[39] <https://colab.research.google.com> [accessed: 5/22/2020].
[40] <https://www.python.org>, version 3.6 [accessed: 5/22/2020].
[41] <https://jupyter.org> [accessed: 5/22/2020].
[42] <https://www.tensorflow.org> (version 2) [accessed: 5/22/2020], see Abadi et al. 2016.

For a proper classification task, the individual datasets (*8k* and *16k*) are randomly divided into two subsets: a training set with 80 % of the dataset and a testing set with 20 % of the dataset. This guarantees that no image from the training set is part of the dataset which is used to report the classification performance. In addition, to ensure that all labels are part of the training and testing set, the random split is done in a stratified fashion: the distribution of labels from the full dataset are forced to be the same for the individual training and testing splits.

3.2.3. *Convolutional neural network and experiment*

The convolutional neural network[43] is built similar to *VGG19*[44] but with fewer layers. It consists of two convolutional blocks with each constructed from two consecutive convolutional layers with 32 filters in the first block and 64 filters in the second block. Both have a filter size of 3×3. They are each followed by a maximum pooling layer of size 2×2 and a dropout layer with a dropout rate set to 0.25.

The classification block following the convolutional blocks is made from two dense layers with 256 nodes each and again is followed by a dropout layer with a dropout rate of 0.5. All activation functions for the trainable layers (convolutional and dense) are set to ReLU, with the exception of the very last layer, the classification layer, which is a dense layer with 61 output nodes (resp. the number of classes for the experiment) and the softmax activation function.

As this classification task is multi-class, the loss function is the categorical cross-entropy which is optimized by the Adam optimizer[45] with *Keras*' default parameters and a learning rate of 0.001. The number of epochs is set to a maximum of 1000 which is not reached because early-stopping is applied that stops the training process when the generalization error increases again after decreasing during training. Due to the mandatory input size of 256×256 pixels, the original images are downsized without retaining the aspect ratio.

To virtually increase the number of training examples and to stabilize generalization, random input augmentation is done during the training phase. There are many methods available for randomly augment an image; we apply three:

1. Random Rotation: Each input image is rotated by an individually randomly chosen angle between −35 and 35 degrees.
2. Random Cropping: Each input image is increased in height and width by 6 pixels, and then a random square of the original image size (256×256) is chosen.

---

[43] Bermeitinger 2019.
[44] Simonyan – Zisserman 2014.
[45] Kingma – Ba 2017.

3. Random Brightness: The brightness of each input image is randomly decreased or increased by a randomly chosen value between −50% and 50%.

After the training process, evaluation is done by predicting the images contained in the testing set. The prediction results are compared with the ground truth data. We apply two different performance metrics: Accuracy and F1-Score.

The accuracy is computed by adding all correct classifications and dividing by the number of items in the testing dataset. This results in a fraction yielding the percentage value of correct classifications. The character error rate (CER) is the percentage of wrongly classified items, meaning the opposite of the accuracy.

The dataset is unbalanced, so the accuracy can't be fully trusted, hence we use the F1-Score for a more reliable measurement. It is the harmonic mean between *precision* and *recall*. As precision and recall are class-specific, we use their mean values over all classes resulting in a number between 0 and 1, with 1 being the best value.

3.3. Results

The mean validation accuracy for dataset 8k is 0.85 which is analogous to a character error rate of 15 %, while the value of 16k (0.71) results in a character error rate of 29 % (table 1). However, since we have an unbalanced distribution of items per label, it is better to evaluate the model by using the mean F1-score.

Table 1. Evaluation results of the model

Dataset	Validation Accuracy	Mean F1-Score	CER
8k	0.85	0.84	15%
16k	0.71	0.68	29%

It is not surprising, that high recognition rates were achieved for graphemes which have a simple structure and do not show large variances in form classes (table 2, fig. 5). In many cases, these graphemes have a tall narrow shape as general appearance like the foot (AKU-D540, Gardiner D58), the flowering reed (AKU-L190, Gardiner M17), the piece of cloth (AKU-R400, Gardiner S29), the windpipe and heart (AKU-F340, Gardiner F35), and the enclosure (AKU-N080, Gardiner O6). In addition to that, the vast majority of low broad signs like the ripple of water (AKU-M430, Gardiner N35), the basket with a handle (AKU-U290, Gardiner V31), the door bolt (AKU-N360, Gardiner O34), and the whip (AKU-U220, Gardiner V22) are recognized correctly. Furthermore, the stool (AKU-P030, Gardiner Q3), the walking legs (AKU-D500, Gardiner D54), and the scribe's kit (AKU-X010, Gardiner Y3) achieve good recognition results.

FIG. 5. NORMALIZED CONFUSION MATRIX FOR DATASET 16K ROUNDED OFF TO TWO DECIMAL DIGITS

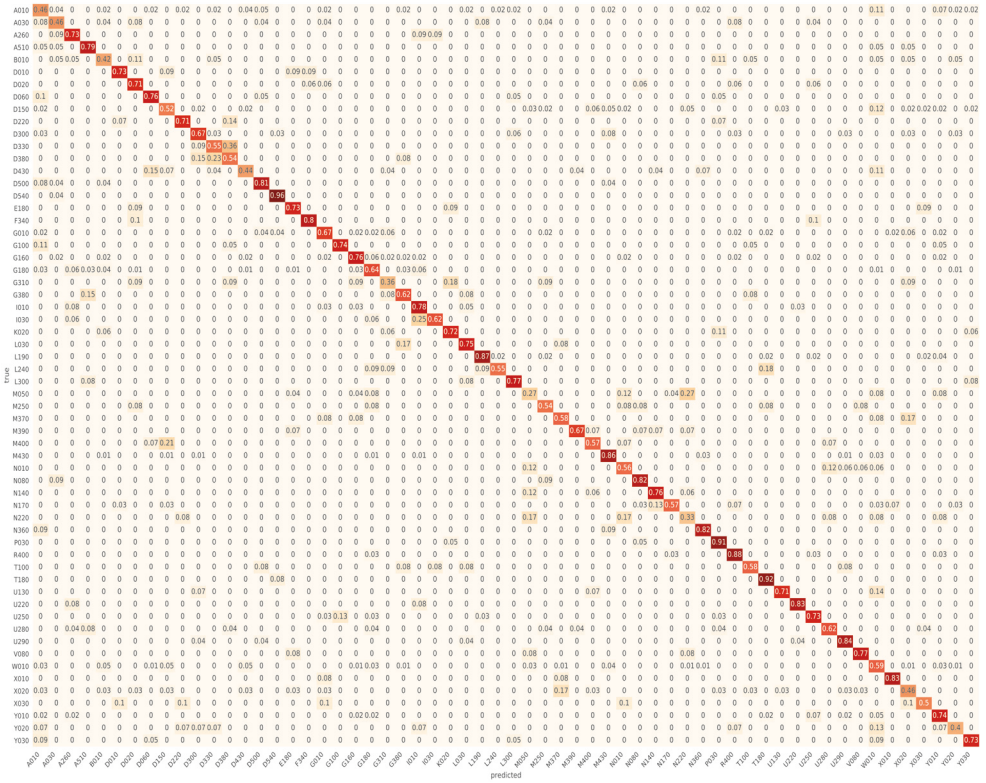


TABLE 2. EXAMPLES OF GRAPHEMES WITH A HIGH CLASSIFICATION ACCURACY

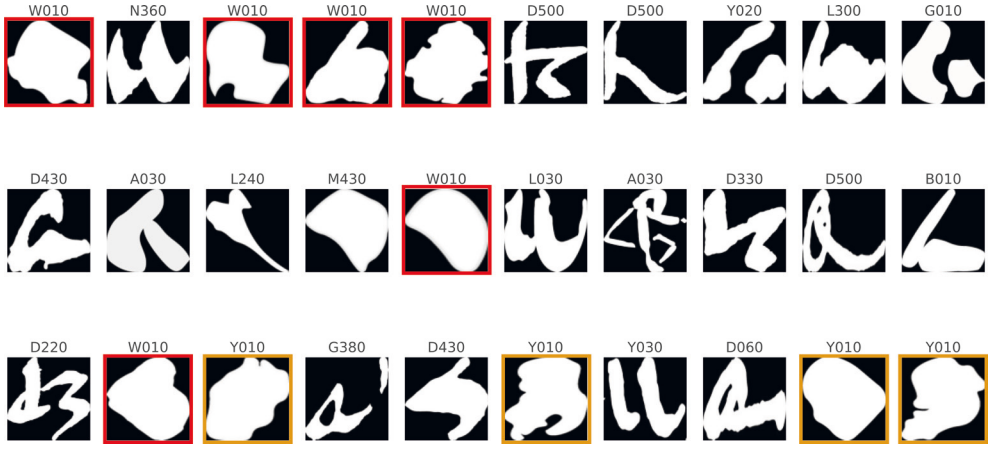
Label	Precision	Recall	F1-Score
D500	0.70	0.81	0.75
D540	0.85	0.96	0.90
F340	0.80	0.80	0.80
L190	0.92	0.87	0.89
M430	0.84	0.86	0.85
N080	0.64	0.82	0.72
N360	0.60	0.82	0.69
P030	0.96	0.87	0.77
R400	0.76	0.88	0.82
T180	0.65	0.92	0.76
U220	0.83	0.83	0.83
U290	0.75	0.84	0.79
X010	0.77	0.83	0.80

In contrast to this, there are some graphemes that show lower classification accuracy. Looking at the confusion matrix, there is clearly a cluster of misclassification visible that contains the graphemes AKU-D300, AKU-D330, and AKU-D380, which is not surprising because of the visual similarity that is shared by these (fig. 5). Figure 6 shows members of class AKU-D330 (𓂧, *rdi*, *ḏi*) together with their "wrongly" predicted labels. It is obvious that these hieratic forms are indistinguishable from those belonging to AKU-D380 (𓂨, *nḫt*) without looking at the context. The predicted labels for AKU-D330 are either AKU-D380 or AKU-D300—all three depict a forearm with or without a short stroke at the end.

Fig. 6. Members of D330 with misclassified labels

Another observation can be made concerning the confusion between two members of different categories. The low F1-score for grapheme AKU-A010 (Gardiner A1) might appear disappointing at first, but considering the signs in detail, it is evident that the visual appearance of many of the misclassified signs is typical for other graphemes (fig. 7). In this case, there is a strong overlapping between the dot-like forms of AKU-A010 and the forms of AKU-W010 (Gardiner X1) resp. AKU-Y010 (Gardiner Z1).

Fig. 7. Members of AKU-A010 (Gardiner A1) with misclassified labels. Highlighted are the dot-like forms that lead to a classification as different signs (red: Gardiner X1, orange: Gardiner Z1)



A further cluster of confusion of specific graphemes is visible within the matrix. It consists of AKU-A010 (Gardiner A1), AKU-A030 (Gardiner A2), and AKU-B010 (Gardiner B1). The confusion between members of these graphemes is quite obvious since they depict all seated human beings. In addition to that, a similar confusion can be observed between small round signs (see table 3) like AKU-M050 (Gardiner N5), AKU-N220 (Gardiner O50), AKU-N010 (Gardiner O49), and AKU-V080 (Gardiner W24), hence for this group of graphemes, the individual F1-Score is very low. However, the lower values of accuracy do not necessarily show evidence of a failure of the model.

Table 3. Examples of graphemes with a low classification accuracy

Label	Precision	Recall	F1-Score
A010	0.54	0.46	0.50
A030	0.55	0.46	0.50
B010	0.36	0.42	0.39
D430	0.55	0.44	0.49
G310	0.33	0.36	0.35
M050	0.33	0.26	0.29
N220	0.21	0.31	0.25
Y020	0.50	0.40	0.44

4. Discussion and conclusion

We have shown that due to the early stage of OCR applications for hieratic, good results at character level can be achieved by a small convolutional neural network. The values of the character error rates describe an average mean of the model's accuracy above the whole datasets and represent successful training. However, a closer look reveals a wide disparity in the values of the individual labels. This cannot only be traced back to the fact that the input data has an unbalanced distribution. Moreover, the recognition process performed by artificial neural networks may replicate results produced by human perception to a certain degree. A convincing example of this is the confusion between the dot-like shapes of AKU-A010 (Gardiner A1) and AKU-W010 (Gardiner X1). Without context, it is almost impossible—even for experts—to classify these indistinguishable shapes correctly. Convolutional neural networks can provide an investigation tool to analyze the structure of the shape system of hieratic forms. They can help us to get a better understanding of the relationship between form classes respectively hieratic signs in general.

For future stages, the other steps of the OCR pipeline should be developed respectively improved. The model created by the *Digital Lahun Papyri* project provides promising solutions for many steps of the recognition process—especially

for the task of detecting possible signs on a papyrus.[46] Other specific jobs like the binarization and segmentation of text and background can also be done with the help of artificial neural networks, such as a modified version of the *U-Net* architecture, that is specialized for segmentation.[47]

In addition to that, our own model should achieve higher accuracy by providing more training data and by using deeper networks. However, the results produced by HTR models promise a high accuracy for ancient Egyptian hieratic, but the lack of annotated material on word level is a major issue that cannot be solved without significant human resource requirements. One possible extension for error correction and accuracy improvement is the use of probabilistic models like the one Nederhof and Rahman presented.[48] Finally, it could be examined whether these models profit from new methods of line detection and segmentation.[49]

5. References

Abadi, M. et al. (2016). "TensorFlow: A system for large-scale machine learning". In *12th USENIX Symposium on Operating Systems Design and Implementation*, OSDI 16, Savannah: 265–283. <https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf> [accessed: 4/27/2020].

Arrivault, D. – Richard, N. – Bouyer, P. (2005). "A fuzzy hierarchical attributed graph approach for handwritten Egyptian hieroglyphs description and matching". In *Eighth International Conference on Document Analysis and Recognition*, ICDAR'05, Seoul: vol. 2, 898 – 902 (doi: 10.1109/ICDAR.2005.11).

Barucci, A. et al. (2021). "A Deep Learning Approach to Ancient Egyptian Hieroglyphs Classification". In *IEEE Access* 9: 123438–123447. https://doi.org/10.1109/ACCESS.2021.3110082.

Bermeitinger, B. (2019). *AKU Classification*. <https://gitlab.com/ds-unisg/aku> [accessed: 4/27/2020].

Dhali, M. A. – Wit, J. W. de – Schomaker, L. (2019). "BiNet: Degraded-Manuscript Binarization in Diverse Document Textures and Layouts using Deep Encoder-Decoder Networks", *ArXiv*, <http://hdl.handle.net/11370/1785a3de-3886-4c10-a011-521748333498> [accessed: 4/27/2020].

Elnabawy, R. – Elias, R. – Salem, M. (2018). "Image Based Hieroglyphic Character Recognition". In *2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. Las Palmas de Gran Canaria: 32–39 (doi: 10.1109/SITIS.2018.00016).

---

[46] Haliassos et al. 2020: 131–136.

[47] Dhali et al. 2019.

[48] Nederhof – Rahman 2017.

[49] Grüning et al. 2019. For a usable application see <https://github.com/TobiasGruening/ARU-Net> [accessed: 5/22/2020].

Franken, M. (2013). *Automatic Egyptian Hieroglyph Recognition by Retrieving Images as Texts*. Unpublished PhD dissertation, University of Amsterdam. <https://esc.fnwi.uva.nl/thesis/centraal/files/f1329601818.pdf> [accessed: 4/27/2020].

Franzini, G. et al. (2018). "Attributing Authorship in the Noisy Digitized Correspondence of Jacob and Wilhelm Grimm", *Frontiers in Digital Humanities* 5 (doi: 10.3389/fdigh.2018.00004).

Gardiner, A. H. (1957). *Egyptian Grammar. Being an Introduction to the Study of Hieroglyphs*. Oxford.

Graefe, E. (1999). "Projekt eines Programms zur teilautomatischen Umsetzung Hieratisch geschriebener Texte in Standardhieroglyphen und Erfassung von Texten in Transkription inklusive Zeichencode". In S. Grunert – I. Hafemann (eds), *Textcorpus und Wörterbuch. Aspekte zur ägyptischen Lexikographie*. Probleme der Ägyptologie 14, Leiden: 229–234.

Grüning, T. et al. (2019). "A two-stage method for text line detection in historical documents", *International Journal on Document Analysis and Recognition* 22: 285–302 (doi: 10.1007/s10032-019-00332-1).

Gülden, S. A. (2018). "Paläographien und Hieratogramme – digitale Herausforderungen". In S. A. Gülden – U. Verhoeven – K. van der Moezel (eds), *Ägyptologische „Binsen"-Weisheiten III. Formen und Funktionen von Zeichenliste und Paläographie. Akten der internationalen und interdisziplinären Tagung in der Akademie der Wissenschaften und der Literatur, Mainz im April 2016*, Abhandlungen der Akademie der Wissenschaften und der Literatur in Mainz – Geistes- und Sozialwissenschaftliche Klasse Einzelveröffentlichung 15, Stuttgart: 83–109. <http://nbn-resolving.de/urn:nbn:de:hebis:77-publ-598075> [accessed: 4/27/2020].

Gülden, S. A. – Verhoeven, U. (2017). "A new long-term digital project on Hieratic and cursive hieroglyphs". In M. C. Guidotti – G. Rosati (eds), *Proceedings of the XI International Congress of Egyptologists. Florence Egyptian Museum Florence, 23–30 August 2015*, Archaeopress Egyptology 19, Oxford: 671–675.

Gülden, S. A. – Verhoeven, U. – Krause, C. (2020). "Digital Palaeography of Hieratic". In D. Laboury – V. Davies (eds), *The Oxford Handbook of Egyptian Epigraphy and Paleography*, Oxford: 634–646. (doi: 10.1093/oxfordhb/9780190604653.013.42).

Haliassos, A. et al. (2020). "Classification and Detection of Symbols in Ancient Papyri". In F. Liarokapis et al. (eds), *Visual Computing for Cultural Heritage*, Springer Series on Cultural Computing, Cham: 121–140.

Hawk, B. W. – Karaisl, A. – White, N. (2019). "Modelling Medieval Hands: Practical OCR for Caroline Minuscule", *Digital Humanities Quarterly* 13. <http://www.digitalhumanities.org/dhq/vol/13/1/000412/000412.html> [accessed: 4/27/2020].

Kingma, D. P. – Ba, J. (2017). "Adam: A Method for Stochastic Optimization", *arXiv*:1412.6980 [cs]. <http://arxiv.org/abs/1412.6980> [accessed: 3/19/2020].

Miyagawa, S. et al. (2019). "Optical character recognition of typeset Coptic text with neural networks", *Digital Scholarship in the Humanities* 34 (Supplement_1): i135–i141. (doi: 10.1093/llc/fqz023).

Van der Moezel, K. (2018). "On signs, lists and standardisation". In S. A. Gülden – U. Verhoeven – K. van der Moezel (eds), *Ägyptologische „Binsen"-Weisheiten III. Formen und Funktionen von Zeichenliste und Paläographie. Akten der internationalen und interdisziplinären Tagung in der Akademie der Wissenschaften und der Literatur | Mainz im April 2016*, Abhandlungen der Akademie der Wissenschaften und der Literatur in Mainz – Geistes- und Sozialwissenschaftliche Klasse Einzelveröffentlichung 15, Stuttgart: 51–81. <http://nbn-resolving.de/urn:nbn:de:hebis:77-publ-598087>. [accessed: 4/27/2020]

Möller, G. (1927a). *Hieratische Paläographie. Die aegyptische Buchschrift in ihrer Entwicklung von der fünften Dynastie bis zur römischen Kaiserzeit: I. Bis zum Beginn der achtzehnten Dynastie* (2nd ed.). Leipzig.

Möller, G. (1927b). *Hieratische Paläographie. Die aegyptische Buchschrift in ihrer Entwicklung von der fünften Dynastie bis zur römischen Kaiserzeit: II. Von der Zeit Thutmosis' III bis zum Ende der einundzwanzigsten Dynastie* (2nd ed.). Leipzig.

Möller, G. (1936a). *Hieratische Paläographie. Die aegyptische Buchschrift in ihrer Entwicklung von der fünften Dynastie bis zur römischen Kaiserzeit. Ergänzungsheft zu Band I und II*. Leipzig.

Möller, G. (1936b). *Hieratische Paläographie. Die aegyptische Buchschrift in ihrer Entwicklung von der fünften Dynastie bis zur römischen Kaiserzeit: III. Von der zweiundzwanzigsten Dynastie bis zum dritten Jahrhundert nach Chr*. (2nd ed.). Leipzig.

Nederhof, M.-J. (2016). "OCR of hand-written transcriptions of hieroglyphic text". In M. Berti – F. Naether (eds), *Altertumswissenschaften in a Digital Age. Egyptology, Papyrology and beyond. Proceedings of a conference and workshop in Leipzig, November 4–6, 2015*. Leipzig. <https://nbn-resolving.org/urn:nbn:de:bsz:15-qucosa-201704> [accessed: 4/27/2020].

Nederhof, M.-J. – Rahman, F. (2017). "A probabilistic model of Ancient Egyptian writing", *Journal of Language Modelling* 5: 131–163 (doi: 10.15398/jlm.v5i1.150).

Quirke, S.G.J. (2010). "Agendas for Digital Palaeography in an Archaeological Context: Egypt 1800 BC". In F. Fischer – C. Fritze – G. Vogeler (eds), *Kodikologie und Paläographie im digitalen Zeitalter 2*. Schriften des Instituts für Dokumentologie und Editorik 3, Norderstedt: 279–294. <http://nbn-resolving.de/urn:nbn:de:hbz:38-43548> [accessed: 4/27/2020].

Reul, C. et al. (2019). "OCR4all—An Open-Source Tool Providing a (Semi-) Automatic OCR Workflow for Historical Printings", *Applied Sciences* 9: 4853. (doi: 10.3390/app9224853).

Rosmorduc, S. (2015). "Computational Linguistics in Egyptology". In J. Stauder-Porchet – A. Stauder – W. Wendrich (eds), *UCLA Encyclopedia of Egyptology*. Los Angeles. <https://escholarship.org/uc/item/0fk4n4gv> [accessed: 4/27/2020].

Selinger, P. (2003). "Potrace: a polygon-based tracing algorithm". <http://potrace.sourceforge.net/potrace.pdf> [accessed: 4/27/2020].

Shomaker, L. (2019). "Lifelong learning for text retrieval and recognition in historical handwritten document collections". <https://arxiv.org/abs/1912.05156> [accessed: 4/27/2020].

Simonyan, K. – Zisserman, A. (2014). "Very Deep Convolutional Networks for Large-Scale Image Recognition", *Computing Research Repository abs/1409.1556*. <http://arxiv.org/abs/1409.1556 > [accessed: 6/20/2017].

Strange, C. et al. (2014). "Mining for the Meanings of a Murder: The Impact of OCR Quality on the Use of Digitized Historical Newspapers", *Digital Humanities Quarterly* 8. <http://www.digitalhumanities.org/dhq/vol/8/1/000168/000168.html> [accessed: 4/27/2020].

Verhoeven, U. (2015). "Stand und Aufgaben der Erforschung des Hieratischen und der Kursivhieroglyphen". In U. Verhoeven (ed.), *Ägyptologische „Binsen"-Weisheiten I–II. Neue Forschungen und Methoden der Hieratistik*. Abhandlungen der Akademie der Wissenschaften und der Literatur in Mainz – Geistes- und Sozialwissenschaftliche Klasse Einzelveröffentlichung 14, Stuttgart: 23–63. <http://nbn-resolving.de/urn:nbn:de:hebis:77-publ-547544> [accessed: 4/27/2020].

van der Walt, S. et al. (2014). "scikit-image: image processing in Python", *PeerJ* 2: e453. (doi: 10.7717/peerj.453).

World Wide Web Consortium (2011). *Scalable Vector Graphics (SVG) 1.1* (2nd ed.). <https://www.w3.org/TR/SVG11/> [accessed: 6/4/2020].