Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2024.050791

ARTICLE

Instance Segmentation of Characters Recognized in Palmyrene Aramaic Inscriptions

Adéla Hamplová[1,*], Alexey Lyavdansky[2,*], Tomáš Novák[1], Ondřej Svojše[1], David Franc[1] and Arnošt Veselý[1]

[1]Czech University of Life Sciences in Prague (CULS), Faculty of Economics and Management, Department of Information Engineering, Prague, 16500, Czech Republic

[2]National Research University Higher School of Economics (HSE), Faculty of Humanities, Institute for Oriental and Classical Studies, Moscow, 101000, The Russian Federation

*Corresponding Authors: Adéla Hamplová. Email: hamplova@pef.czu.cz; Alexey Lyavdansky. Email: andurar@gmail.com

Received: 18 February 2024 Accepted: 12 April 2024 Published: 08 July 2024

ABSTRACT

This study presents a single-class and multi-class instance segmentation approach applied to ancient Palmyrene inscriptions, employing two state-of-the-art deep learning algorithms, namely YOLOv8 and Roboflow 3.0. The goal is to contribute to the preservation and understanding of historical texts, showcasing the potential of modern deep learning methods in archaeological research. Our research culminates in several key findings and scientific contributions. We comprehensively compare the performance of YOLOv8 and Roboflow 3.0 in the context of Palmyrene character segmentation—this comparative analysis mainly focuses on the strengths and weaknesses of each algorithm in this context. We also created and annotated an extensive dataset of Palmyrene inscriptions, a crucial resource for further research in the field. The dataset serves for training and evaluating the segmentation models. We employ comparative evaluation metrics to quantitatively assess the segmentation results, ensuring the reliability and reproducibility of our findings and we present custom visualization tools for predicted segmentation masks. Our study advances the state of the art in semi-automatic reading of Palmyrene inscriptions and establishes a benchmark for future research. The availability of the Palmyrene dataset and the insights into algorithm performance contribute to the broader understanding of historical text analysis.

KEYWORDS

Optical character recognition; instance segmentation; Palmyrene; ancient languages; computer vision

1 Introduction

Palmyra, known as Tadmur in Arabic, is an ancient city located in the Syrian desert. It is also an essential part of human history. Its archaeological significance lies not only in its physical ruins, but also in the inscriptions carved into the buildings and into the funerary stelae. These inscriptions represent a valuable repository of knowledge that records the Palmyrene dialect of Aramaic, its culture, and the records of ancient Palmyrene society. However, uncovering the secret written on these inscriptions poses challenges for the scientific community.

Deciphering and analyzing these historical texts have interested scientists, historians, and archaeologists for generations, and until now it has only ever been done by linguists, not by machines. Therefore, applying deep learning (DL) methods is a transformative force, making the work of linguists easier and allowing the non-scholarly public access to texts that would otherwise be incomprehensible to them. Deep learning algorithms, including deep neural networks, offer automation in letter classification and segmentation, which can be a potential solution to the complexity of the transcription of Palmyrene inscriptions.

Previous research [1] dealt with classifying Palmyrene characters from handwritten transcripts and photographs and their augmentation [2]. It addressed the classification in two ways. The first way is to divide a dataset of Palmyrene characters into squares that each contain one letter; the second way is to handwrite an EMNIST-like dataset using special software and a mouse pen tablet and then make both classifiers available in an Android mobile application and an online application [3], using a custom neural network which was chosen as the best performing from 10 different architectures. As photographs classification did not achieve satisfactory results initially, Generative Adversarial Networks (GAN) are employed to expand the classification dataset, improving the outcomes by 120%. The research plan for segmenting Palmyrene characters was presented at a conference in 2023 [4].

Based on deep learning principles, this study aims to evaluate and compare the performance of state-of-the-art DL instance segmentation algorithms in Palmyrene character segmentation. Through data collection in collaboration with several museums worldwide, photo analysis, pre-processing, manual review of published transcriptions, and custom annotation in the Roboflow annotation platform, the computing power of DL is employed to solve the unique challenges posed by transcriptions of ancient inscriptions.

2 Structure

The article is structured as follows:

Section 3—Related Work-presents other works that describe developing an ancient or alive language Optical Character Recognition (OCR) and comment on the proposed methods. It also presents other relevant studies that utilize instance segmentation and the research gap.

Section 4—Data Collection and Preparation-describes how the data were obtained from the museums, checks the published transliterations to indicate if they align with the photographs, and adds the transliterations to photos that did not have them available. It also describes the pre-processing and annotation process and its challenges and defines the number of classes to work with.

Section 5—Methodology-introduces two approaches-single-and multi-class segmentation-and algorithms used-YOLOv8 and Roboflow 3.0. It explains the advantages and disadvantages of each method and describes the training, hyperparameters, and evaluation metrics. It also presents the custom scripts developed for letter sorting and visualization.

Section 6—Results-includes individual network training, testing, quantitative metrics, and visualization of results.

Sections 7 through 9 discuss the results, next steps, and conclusion. At the end, statements, acknowledgments, and references are presented, followed by Appendices A and B that provide details of the models' training and testing.

3 Related Work

Developing OCR or Natural Language Processing (NLP) algorithms for languages lacking existing solutions is an essential part of preservation and making it easier to process documents in that given language. This applies to historical texts, such as Egyptian hieroglyphs [5], Sanskrit [6], and different types of cuneiform [7] and living languages.

For instance, a Turkish OCR system [8] employs commonly available OCR algorithms-CuneiForm Cognitive OpenOCR, GNU License Open-source Character Recogniton (GOCR), and Tesseract [9]-to handle a dataset consisting of scans and photos of Turkish texts. Another was designed for Icelandic to aid in digitizing the Fjolnir magazine, housing historical texts [10]. Character recognition has also been developed for Bangla [11], presenting unique challenges due to the variability of characters and the presence of ligatures (conjunctions of characters). Oni et al. [12] developed an OCR algorithm based on generated training data. They scanned images of Yoruba texts written in Latin script and reached 3.138% character error rate using the Times New Roman font.

There are comparative performance studies for or languages with many OCR systems available, either of the whole systems [13] or separate languages, such as Arabic [14].

Using instance segmentation algorithms for character detection can be effective in image-based tasks involving handwriting, as opposed to OCR for scanned text, where semantic segmentation is employed to separate text from background, e.g., in the case of Czechoslovak scanned documents [15]. Instance segmentation using Convolutional Neural Networks (CNNs) is applied to detect the boundaries of individual objects. It is usually used for other tasks, such as segmenting leaves in plants [16], cars in a parking lot [17], or ships and airplanes from satellite images [18].

Although instance segmentation algorithms are usually used for tasks other than letter segmentation, they also have a high potential to find letters in photographs. Instance segmentation can make it possible to recognize characters in different font styles and photographs of various quality if a large enough dataset is available, and it is not necessary to separate the text from the non-text part.

4 Data Collection and Preparation

4.1 Obtaining Data

Photographs of Palmyrene inscriptions were obtained from several private sources with their consent, from public online sources, and by taking photographs in the respective museums. The photographs of inscriptions originate from Arbeia Roman Fort and Museum[1], Archaeological Museum of Palmyra[2], The British Museum[3], Carlsberg Glyptotek[4], Hypogeum of Three Brothers[5], MET Museum[6], Musée du Louvre[7], Musei Vaticani[8], Museum of the American University, Beirut[9], The Getty Villa Museum[10], National Museum in Prague[11], Royal Ontario Museum[12], The Pushkin State Museum of Fine Arts[13] and The State Hermitage Museum[14].

[1] https://arbeiaromanfort.org.uk/.
[2] https://virtual-museum-syria.org/palmyra/.
[3] https://www.britishmuseum.org/.
[4] https://glyptoteket.dk/.
[5] https://archeologie.culture.gouv.fr/palmyre/en/mediatheque/hypogeum-three-brothers-palmyra-7.
[6] https://www.metmuseum.org/.
[7] https://www.louvre.fr/en.
[8] https://www.museivaticani.va/content/museivaticani/en.html.
[9] https://www.aub.edu.lb/museum_archeo/Pages/default.aspx.
[10] https://www.getty.edu/visit/villa.
[11] https://www.nm.cz/en.
[12] https://www.rom.on.ca/en.
[13] https://pushkinmuseum.art/?lang=en.
[14] https://www.hermitagemuseum.org/wps/portal/hermitage/.

4.2 Checking Transcriptions

Prior to the annotation, the collected photographs were checked. For each photo, the visible letters were checked. For some photographs, previously published transcriptions were available and edited to match the visible characters in the photographs. For those photographs that did not have transcripts available, transcripts were created.

4.3 Annotation and Pre-Processing

The annotation of the instance segmentation dataset was based on the checked and newly created transcriptions and was completed in the Roboflow annotation tool using 26 classes corresponding with the Palmyrene characters. Table 1 shows the complete list.

Table 1: Palmyrene character classes in multi-class segmentation

Class index	Class name	Transcription	Palmyrene
0	One	1	𐡱
1	Ten	10/100	𐡲
2	Twenty	20	𐡳
3	Aleph	ʔ	𐡠
4	Ayin	ʕ	𐡢
5	Beth	b	𐡡
6	Gimel	g	𐡣
7	He	h	𐡤
8	Heth	ḥ	𐡧
9	Kaph	k	𐡩
10	Lamedh	l	𐡪
11	Mem	m	𐡫
12	Nun	n	𐡬
13	Nun_final	n	𐡭
14	Pe	p	𐡮
15	Qoph	q	𐡰
16	Resh/daleth	r/d	𐡶𐡴
17	Right	>	𐡵
18	Sadhe	ṣ	𐡧
19	Samekh	s	𐡯
20	Shin	š	𐡱
21	Taw	t	𐡲
22	Teth	ṭ	𐡦
23	Waw	w	𐡥
24	Yodh	y	𐡩
25	Zayin	z	𐡦

The characters “left” 𐡶 and “right” 𐡷 are paratextual signs similar to punctuation marks. Traditionally, they are labeled “ivy leaf” and put either at the beginning or at the end of a line, or a whole text in Palmyrene Aramaic and Greek inscriptions. Generally, the “left” ivy leaf is used much more often than its right counterpart.

The character “left” 𐡶 was not present in any of the photographs, so it was excluded from the class list, but it was included in the classification dataset. The characters “resh” 𐡩 and “daleth” 𐡣 were combined into a single class because they are often written identically, with their distinction depending only on the context. Sometimes, “resh” 𐡩 is marked with a dot above. However, a segment must be a continuous object. Hence, the dot will make a separate segment. There are some dotted “resh” 𐡩 in the dataset, but they are a minority compared to the volume of those that are not dotted.

The same blending applies to characters “10” and “100”, “5” and “ayin”. In some visual variants, this also applies to the pair “mem” and “qoph” and the pair “heth” and “sadhe”, however, they were preserved as a separate class, as other visual variants are distinguishable.

5 Methodology

5.1 Instance Segmentation

Segmentation is the most intricate of the three computer vision tasks: classification, object detection, and segmentation [19]. It involves pixel-level classification, where pixels are grouped based on the selected class, revealing the precise boundaries of objects. There are two main types of segmentation: semantic segmentation, which clusters pixels belonging to the same class regardless of whether objects overlap, and instance segmentation, which identifies individual instances of objects within the same class. Instance segmentation determines the outlines of each instance based on factors such as shape, texture, brightness, and color [20].

During the training of an instance segmentation model, four types of losses are minimized in parallel, including box, segmentation, class, and distributional focal (box_loss, seg_loss, cls_loss, dfl_loss). The box loss measures the difference between predicted bounding box coordinates and the ground truth bounding box coordinates for each object instance, typically calculated as smooth L1 loss. The segmentation loss quantifies the difference between the predicted segmentation mask and the ground truth mask for each object instance. The class loss describes the variation between predicted class probabilities and the true class labels associated with each object instance. It is typically computed using a categorical cross-entropy loss function. The distributional focal loss is a modified version of the focal loss employed to solve the class imbalance problem. More information about the losses can be found in the literature [21].

This study uses two approaches to extract text from a photo using instance segmentation.

5.1.1 Single-Class Instance Segmentation

The first approach aims to segment letters regardless of their class and semantic meaning. Hence, the identified segments are ordered as text (right to left, top to bottom), and plotted one by one in the empty images (as described in Section 5.4, Custom Tools). These individual images are input for classification, and the classified features in the correct order form the entire text in the photo. This approach of looking for segments in only one class greatly increases the chance of finding more segments since the neural network only looks for one class.

5.1.2 Multi-Class Instance Segmentation

The second approach uses multi-class segmentation. Each letter is identified separately in the dataset, making it more accurate to find them and draw more correct segmentation masks. However, many letters are underrepresented in the dataset, so the segmentation algorithms do not find them and miss them entirely in the resulting text transcription. This problem will be solved through a significant dataset extension, which is currently in progress.

5.2 Selected Segmentation Models, Their Advantages and Disadvantages, Hyperparameters and Training

This study selected two instance segmentation algorithms. A comparison between YOLOv8 and Mask Region-based Convolutional Neural Network (R-CNN) was presented in 2023 [22] and showed that YOLOv8 performs better on selected images from fish-eye cameras. Like the images of the sandstone tablets with the Palmyra inscriptions, these images are of lower quality, and YOLOv8 can find more objects than the more accurate R-CNN. Roboflow Train was also chosen because this company offers dataset management, integrates an annotation tool, and offers data augmentation directly in the application. Thus, training directly in this particular application is relevant as the dataset was annotated there.

5.2.1 YOLOv8

YOLO, short for You Only Look Once, was released in 2016. It belongs to the category of one-shot detectors, which are generally less accurate but very fast, contrary to two-stage detectors, which are more accurate and slower [23]. YOLO has been under development for multiple years by Redmond et al. [24–26] until he decided to retreat from the research in fear of potential misuse by social media companies and the military; however, other teams took over his work. The first version of YOLO to incorporate instance segmentation was YOLOv5 in September 2022 [27]. It was developed by Glenn Jocher as an object detection algorithm [28]. The most contemporary version-YOLOv8, includes instance segmentation from January 2023 [29].

The main advantages of using YOLO are its training and inference speed, but it generally comes with lower accuracy.

The selected YOLOv8 instance segmentation model comprises 261 layers, 11800158 parameters, 11800142 gradients, and 42.7 GFLOPs. The complete architecture overview is indicated in Table A1 in Appendix A. The initial weights "yolov8n-seg.pt" are trained on the COCO dataset, and the transfer learning technique is used.

5.2.2 Roboflow 3.0 Instance Segmentation (Accurate)

Roboflow Train 3.0 is a model included in the Roboflow web application, released in July 2023 [30]. There are two options: fast or accurate training. However, the company has not publicly disclosed technical specifics about the structure and architecture. The main advantages are the simplicity of use and remote training, and the disadvantages are the lack of control over the model, as the only options the user can influence are the model type and providing a custom dataset with selected augmentation options.

5.3 Evaluation Metrics

Each Palmyrene text within a photo examines whether the correct number of characters is identified and whether the characters are correctly classified. Error analysis can be performed for

three main types of errors within the OCR transcription of a whole test set, and more derived metrics can be used. The following errors can occur when processing a test dataset:

- **Insertion Errors:** The system found a character where there was none. This study denotes the number of these errors as I.
- **Substitution Errors:** The system found the character in a certain location but misclassified it. This study denotes the number of such errors as S.
- **Deletion Errors:** There was a character at that location, but the system found no character at that location. This study denotes the number of these errors as D.
- **Total Levenshtein Distance:** The total number of errors that occurred during the processing of the test data set is:

$$TLD = I + D + S \tag{1}$$

where TLD is the Total Levenshtein Distance. The Levenshtein Distance, also known as the Edit-Distance algorithm, measures the number of characters that must be changed, added, or deleted in the predicted word so that it matches the true word [31]. Total Levenshtein distance does not apply to a word; it applies to the whole text.

- **Total Character Accuracy:** In addition to the Total Levenshtein Distance, the system's behavior will be evaluated using the Total Character Accuracy metric, which will rate the overall quality of the transcript. This study denotes the total number of letters as N and the Total Character Accuracy as TCA, where:

$$TCA = \frac{100 \cdot (N - S - D)}{N} \tag{2}$$

Thus, TCA determines the percentage of characters correctly found and correctly classified in the test dataset. The TCA value does not depend on the number of insertion errors I. Therefore, the value of the I parameter or the Total Levenshtein Distance that incorporates the I value must also be considered when evaluating the system's overall quality.

5.4 Custom Tools

The image is processed to text, as shown in Fig. 1.

5.4.1 Prediction Scripts

Due to the use of two segmentation methods and, thus, four different models, the characters in images are predicted in multiple ways. However, a 40% confidence score is always set as a threshold.

This study predicts using the stored model on the web server for single-class segmentation using YOLO. It obtains a list of identified segments labeled as "1" only (meaning a character). These are then sorted by the developed program from right to left, top to bottom (see Section 5.4.3 Sort), and after sorting, they are printed on a square image (Section 5.4.3 Draw), which is input to the classifier, classifying them in that order and outputs the resulting text.

For single-class segmentation using Roboflow, we use the Roboflow API snippet to predict the segments. The segments are then converted to YOLO format, and the subsequent procedure is identical to YOLO single-class segmentation.

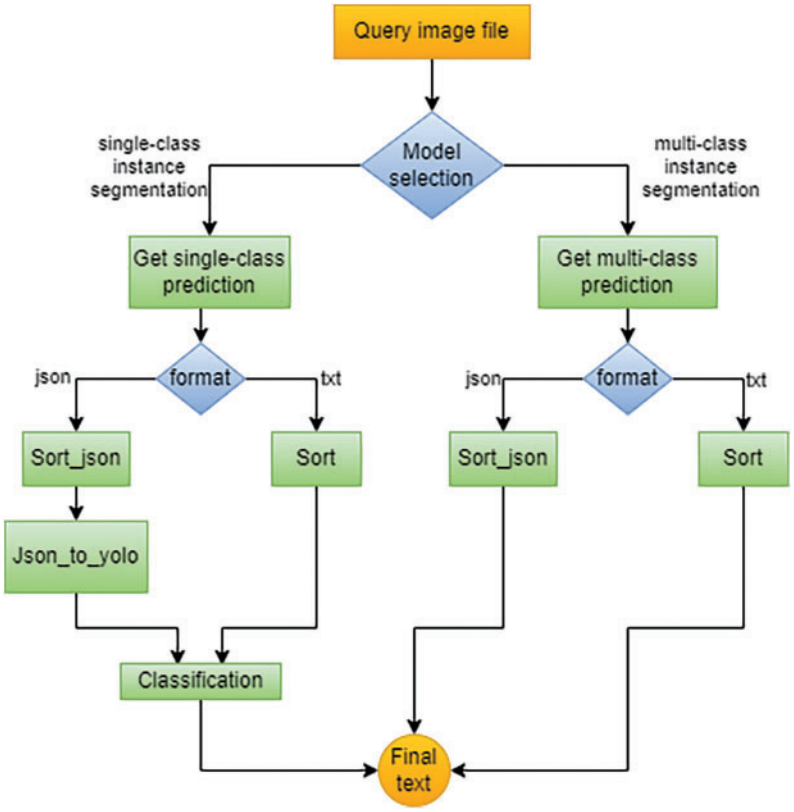

Figure 1: Palmyrene character instance segmentation-process flow diagram

Multi-class segmentation using YOLO uses a second model stored on the web server, the outputs are segments already assigned to the appropriate characters. These are further sorted, and the resulting text is obtained directly from the sorting tool. When using the last model, Roboflow multi-class segmentation, the predictions from the ".json" format are converted to the Yolo format. Then, the segments are sorted to produce the resulting text.

5.4.2 Detecting Rows and Sorting Letters

Classical line detection algorithms for scanned documents assume that the lines are straight, and in handwritten documents [32], line detection is performed in the original image before detecting separate letters. Another successful approach to detect lines in documents is to use Google Tesseract [33], but it does not support the Palmyrene language. The traditional algorithms assume text linearity and regularity, which is absent in the historical texts captured in the photographs of sandstone inscriptions. Such handwriting has considerable variability, which causes irregularities in spacing, angles of lines, and diverse styles, which were unique to each person. Palmyrene also uses irregular fonts in some cases. It was, therefore, necessary to address the issue in a specific manner.

The study cannot use either of the mentioned approaches because they are not intended to sort polygons already detected by YOLO or Roboflow Instance Segmentation. Since these polygons are restored from photographs, the rows in the images are ambiguous and not always straight.

At first, the algorithm in sort.py reads polygon information in YOLO instance segmentation format (the class index, points as x, y coordinate tuples, and confidence score). If the format is different, the variant sort_json.py is used, and subsequently, the output is converted for further processing by another custom script *json_to_yolo.py*. The sorting principle is as follows:

1. The average height $\overline{h}$ of the polygons (x_i,y_i) is determined (see Fig. 2).
2. Polygons are sorted in descending order according to their y_i coordinate.
3. Splitting into rows: For all pairs of polygons $((x_i, y_i), (x_{i+1}, y_{i+1}))$, $i = 1, \ldots, n–1$, we determine, whether

$$|y_{i+1} - y_i| > 0.5 \cdot \overline{h} \tag{3}$$

If the result of the inequality is true, y_i becomes the last polygon of the current row and y_{i+1} becomes the first polygon of the subsequent row.

4. Polygons (x_i, y_i), $i = 1, ..., n$, are arranged in each row based on the size of the x_i coordinate in descending order (the Palmyrene text is read from right to left).

Finally, the output text is printed, and the sorted polygons are saved to the file whose name was specified when the script was run.

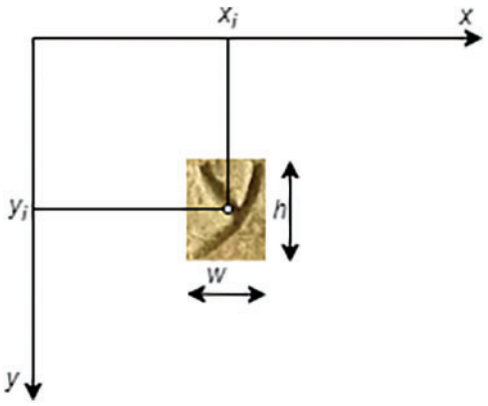

Figure 2: Coordinates (x_i, y_j)

5.4.3 Visualization Tools

Sort

The sorting tool includes plotting the polygons and class names in a plot, as depicted in Fig. 3.

Draw

draw.py visualizes separate polygons, which are printed into a black-and-white binary image in the correct order, which is an input to classification. The relative coordinates of the polygons (obtained by YOLO or converted to YOLO format from the ".json" format used by Roboflow 3.0) are scaled to match the size of the original image. The algorithm processes each polygon in the dataset. It scales the polygon's relative coordinates to fit the original image's size. Then, the polygons are drawn as white letters into a black image in the original polygon size. Subsequently, the polygons are cropped or stretched to a target size (80 × N or N × 80) based on the aspect ratio of the polygons and put in the center of a 100 × 100 black image, which is saved with a filename that indicates the polygon index.

The output of this tool is illustrated in Fig. 4. Then, the letters are classified using the classification prediction script, resulting in a final list of transcribed letters.

Figure 3: Plotted polygons from YOLO multi-class and single-class predictions of a photo of "Inv.1438/8582, Archaeological museum of Palmyra", generated by *sort.py* tool

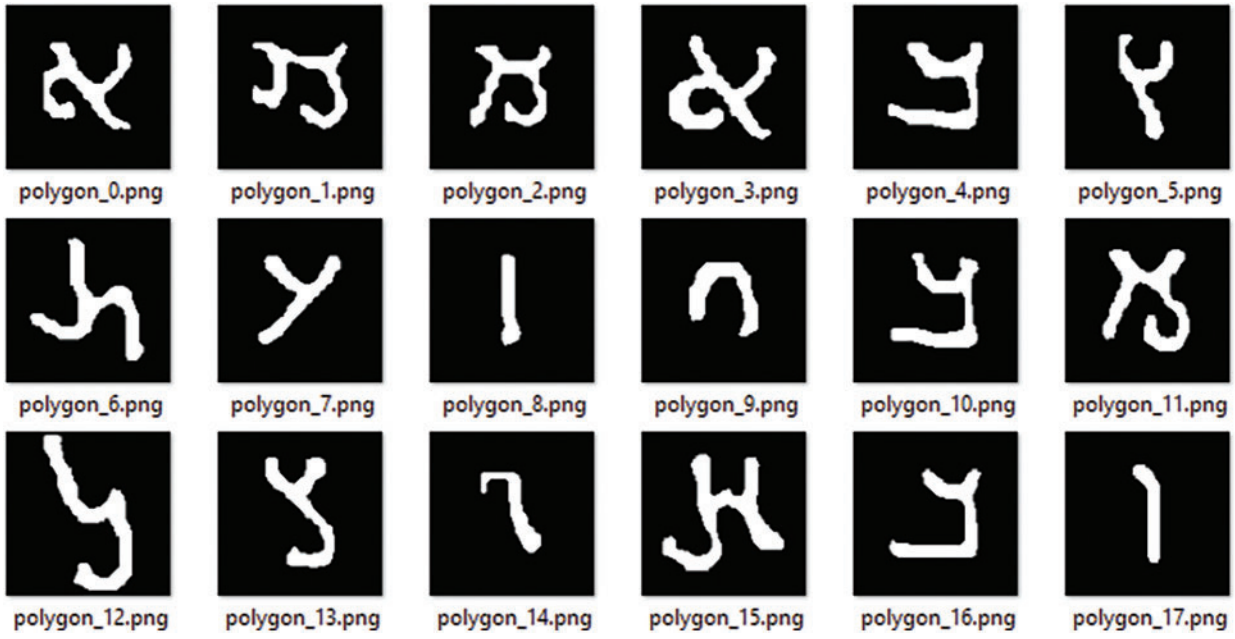

Figure 4: Plotted and sorted polygons for classification

6 Results

6.1 Training Results

The models were trained on the dataset consisting of 119 images with 3578 hand-annotated Palmyrene characters, resized to 920 × 920 pixels, and augmented to triple the dataset size using the following augmentation options:

- Grayscale: Apply to 50% of images
- Saturation: Between −60% and +60%
- Brightness: Between −11% and +11%
- Exposure: Between −11% and +11%
- Blur: Up to 1.25px

The value of the loss functions box_loss, seg_loss, cls_loss, and dfl_loss on the training set steadily decreases during the learning process.

The values of the four loss functions on the validation datasets oscillate, but their mean values also reduce. The smoothest decrease of the validation loss functions can be observed on the YOLOv8 multi-class model and the most random changes on the Roboflow 3.0 single-class model. The detailed training Figs. A1–A8 are provided in Appendix A. Table 2 lists the training results of each segmentation algorithm after the first and last epochs are rounded to 2 decimal places. There are 100 epochs for the YOLOv8 model and 120 epochs for the Roboflow model.

Table 2: Training of all models in the first and last epoch

Model		Roboflow 3.0 multi-class	YOLOv8 multi-class	Roboflow 3.0 single-class	YOLOv8 single-class
First epoch	box_loss	1.56	1.11	1.15	2.28
	seg_loss	2.91	2.12	2.21	4.63
	cls_loss	2.97	0.95	0.97	3.30
	dfl_loss	1.23	0.95	1.04	1.59
Last epoch	box_loss	0.31	0.68	0.62	0.84
	seg_loss	0.85	1.57	1.36	1.71
	cls_loss	0.30	0.48	0.39	0.48
	dfl_loss	0.14	0.85	0.86	0.87

6.2 Evaluation Results

The success of single-class segmentation with subsequent classification and multi-class segmentation was evaluated on six images with Palmyrene inscriptions with 216 characters. Each image was analyzed for errors specified in the Section 5. Only images with clear inscriptions were selected for the test, as the models did not perform well on lower-quality images. Tables A2–A7 include the original texts and comparisons to predictions available in Appendix B and a summary is present in Table 3. Original text in [brackets] indicates letters that are not visible in the photo but are part of the inscription. Errors in the predicted texts are labeled in the texts as follows:

Table 3: Overall evaluation of all models

	YOLO single-class	YOLO multi-class	Roboflow single-class	Roboflow multi-class
Insertion Errors	9	6	0	8
Deletion Errors	2	7	10	17
Substitution Errors	47	7	60	5
Total Character Accuracy	77.3%	93.5%	67.6%	89.8%
Total Levenshtein Distance	58	20	70	30

(1) insertion errors: **bold and underlined**, (2) substitution errors: **bold**, (3) deletion errors: **bold dash-**. All plotted figures with texts generated by the *sort.py* tool are available on GitHub [12].

7 Discussion

The results indicated that the Roboflow 3.0 multi-class model should be theoretically best performing as the losses in the last epoch are the least of all trained models. However, it ultimately achieves a Total Levenshtein Distance of 30 and a Total Character Accuracy of 89.8%, placing this model in second position. The subsequent tests showed that the YOLO multi-class instance

segmentation model performs best with the least Total Levenshtein Distance of only 20 and the highest Total Character Accuracy of 93.5%. The evaluation of both these models proved that using the multi-class segmentation method attains satisfactory results because the predicted segmentation mask shapes are very accurate.

However, the single-class instance segmentation method with consecutive classification is insufficient for practical use as the Total Character Accuracy reached only 77.3% for the YOLO single-class instance segmentation model, and its Total Levenshtein Distance was too high with the value of 58, due to a high number of misclassified characters. The Roboflow single-class model reached 67.6% Total Character Accuracy with a Total Levenshtein Distance of 70.

Although the classifier of handwritten Palmyrene characters, which was utilized to classify the predicted polygons created from instance segmentation masks, reached over 98% for classifying handwritten characters [1], the issue causing the misclassification can be the thickness of the lines, as the classifier was trained on artificially written characters with a fixed line thickness, which was significantly smaller. Sometimes, the predicted segmentation masks were very wide. Also, some of the predicted segments had incomplete shapes.

The best performing (YOLO multi-class) model was implemented in the web application ML-research [3] under the tab "Segmentation & Transcript".

The average accuracy of the OCR of Egyptian hieroglyphs was 66.64%, surpassing the state of art, which was 55.27% before that [9]. Arabic character recognition using Deep Belief Network (DBF) and Convolutional Deep Belief Network (CDBF) was 83.7% accuracy on the IFN/ENIT Database on a model that reached 97.4% accuracy during training [34]. A Holography graph neuron-based system (HoloGN) for handwritten Persian characters [35] was over 90% accuracy when using a dataset extracted from 500000 images of isolated Farsi characters written by hand by Iranian people, but only 45% on images downsized to 32×16 pixels due to memory use optimization when using feedforward Artificial Neural Network (ANN). By comparing this study's results to those of others in historical alphabets OCR, the proposed algorithm performed well with 93.5% accuracy when used on high-quality images of Palmyrene inscriptions.

8 Limitations and Next Steps

Some limitations can be encountered when using instance segmentation algorithms to identify Palmyrene characters in photographs. A possible problem arises from underrepresenting some characters in the training dataset. Although some letters such as "b", "d/r", "y" and "l" occur in almost every inscription, others such as "left", "right", "pe" and "samekh" appear quite rarely.

In the case of single-class instance segmentation, a limitation is the occasional inaccurate identification of polygons derived from the segmentation masks of letters, which can cause the character to be assigned to a different class than the one to which the letter belongs during subsequent classification. In order to address this problem, the polygons can be added to the training subset, and the handwritten character classifier can be retrained. This is subject to testing as it can bias the results of handwriting classification.

When choosing multi-class segmentation, there is a potentially higher risk of encountering deletion errors-missing some letters-especially for the Roboflow Instance Segmentation model. Since this type of segmentation expects very accurate character shapes presented to it during training, this can lead to missed letters in the recognized text when predicting texts in new images.

In the next steps of this research, the focus will be on integrating natural language processing (NLP) techniques to combine identified letters into words and sentences and to enable translation into other world languages. Developing an NLP module that interprets contextual relationships between characters requires collaboration with experts in the Palmyrene language. Continuous and dynamically updated expansion of the dataset by including photos of Palmyrene inscriptions with newly created transcriptions will ensure refinement of the current models and experiment with all possible data augmentation options. The study hopes to include the data in standard OCR training datasets, making it easily accessible for further experiments.

9 Conclusion

This study creates an instance segmentation model, which can identify and transcribe letters within high-quality photos of Palmyrene inscriptions with an accuracy of 93.5%, a significant step towards developing a Palmyrene OCR algorithm.

The development of tools capable of reading the characters and texts of dead languages has impactful sociological importance, as it links the past and the present. Inscriptions in dead languages carry information about important aspects of human history, in the case of Palmyrene Aramaic, recorded in the funerary, honorific, and dedicatory texts. By establishing OCR technology for this language, the potential for understanding ancient texts is expanded to a wider range of linguists, historians, archaeologists, museum keepers, and possibly even the non-scholarly public.

The final goal of humanists and linguists is to decipher the letters individually and understand the entire inscriptions and contextual meaning, which is not a simple objective that can be accomplished with a single computational task. However, this research is an essential step towards deciphering the texts in Palmyrene Aramaic, and the methodology used can be applied to the analysis and extraction of characters from other alphabets that do not use ligatures. The letters can be spatially separated from each other.

Acknowledgement: We would like to thank our grant agency for providing us with the funds necessary to perform this research. We would also like to thank our reviewers and editors for perfecting our manuscript.

Funding Statement: The results and knowledge included herein have been obtained owing to support from the following institutional grant. Internal grant agency of the Faculty of Economics and Management, Czech University of Life Sciences Prague, Grant No. 2023A0004 – "Text Segmentation Methods of Historical Alphabets in OCR Development". https://iga.pef.czu.cz/. Funds were granted to T. Novák, A. Hamplová, O. Svojše, and A. Veselý from the author team.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: A. Hamplová; data collection and annotation: A. Hamplová; inscription checking and text transcriptions: A. Lyavdansky; analysis and interpretation of results: A. Hamplová; web application updates: D. Franc, O. Svojše; draft manuscript preparation: A. Hamplová, A. Veselý, A. Lyavdansky; finance resource management: T. Novák; formatting references and the article structure in accordance with CMES template: A. Hamplová, T. Novák. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The latest dataset version is available at Roboflow [36]. The code that was developed in this research is available on GitHub [37]. By publishing the dataset and code related to our research, transparency and reproducibility are ensured.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Hamplová A, Franc D, Veselý A. An improved classifier and transliterator of handwritten Palmyrene letters to Latin. Neural Netw World. 2022;32:181–95. doi:10.14311/NNW.2022.32.011.
2. Franc D, Hamplová A, Svojše O. Augmenting historical alphabet datasets using generative adversarial networks. In: Data Sci Algorithms Syst Proc 6th Comput Methods Syst Softw 2022; 2023;2:132–41.
3. Hamplová A. Palmyrene translation tool. Available from: https://ml-research.pef.czu.cz. [Accessed 2021].
4. Hamplová A, Franc D, Pavlicek J. Character segmentation in the development of palmyrene aramaic OCR. In: Model-driven organizational and business agility; Zaragoza, Spain; 2023 Jun 12–13. p. 80–95.
5. Elnabawy R, Elias R, Salem M. Image based hieroglyphic character recognition. In: 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS); 2018; Las Palmas de Gran Canaria, Spain. p. 32–9.
6. Avadesh M, Goyal N. Optical character recognition for Sanskrit using convolution neural networks. In: 2018 13th IAPR International Workshop on Document Analysis Systems (DAS); 2018; Vienna, Austria. p. 447–52.
7. Gordin S, Gutherz G, Elazary A, Romach A, Jiménez E, Berant J, et al. Reading Akkadian cuneiform using natural language processing. PLoS One. 2020;15(10):1–16.
8. Karasu K, Bastan M. Turkish OCR on mobile and scanned document images. In: 2015 23nd Signal Processing and Communications Applications Conference (SIU); 2015; Malatya, Turkey. p. 2074–7.
9. Smith RW, Zanibbi R, Coüasnon B. History of the Tesseract OCR engine: what worked and what didn't. In: SPIE Optical Engineering + Applications; 2013; San Diego, California, USA.
10. Daoason JF, Bjarnadóttir K, Rúnarsson K. The journal fjolnir for everyone: the post-processing of historical OCR texts. In: LREC 2014, Ninth International Conference on Language Resources and Evaluation; 2014; Reykjavik, Iceland.
11. Rahaman A, Hasan MM, Shuvo MF, Ovi MAS, Rahman MM. Analysis on handwritten Bangla character recognition using ANN. In: 2014 International Conference on Informatics, Electronics & Vision (ICIEV); 2014; Dhaka, Bangladesh. p. 1–5.
12. Oni OJ, Asahiah F.O Computational modelling of an optical character recognition system for Yorùbá printed text images. Sci Afr. 2020;9:1–12.
13. Salah AB, Moreux JP, Ragot N, Paquet T. OCR performance prediction using cross-OCR alignment. In: 13th International Conference on Document Analysis and Recognition (ICDAR); 2015; Tunisia; p. 556–60.
14. Alghamdi MA, Alkhazi IS, Teahan WJ. Arabic OCR evaluation tool. In: 2016 7th International Conference on Computer Science and Information Technology (CSIT); 2016; Amman, Jordan. p. 1–6.
15. Gruber I, Hlaváč M, Hrúz M, Železný M. Semantic segmentation of historical documents via fully-convolutional neural network. In: Lecture Notes in Computer Science (LNAI). Cham, Linz, Austria: Springer; 2019. vol. 11658, p. 142–9. doi:10.1007/978-3-030-26061-3.
16. Yi J, Wu P, Liu B, Hoeppner DJ, Metaxas DN, Fan W. Object-guided instance segmentation for biological images. IEEE Trans Med Imag. 2021 Sep;40(9):2403–14. doi:10.1109/TMI.2021.3077285.
17. Berry T, Dronen N, Jackson B, Endres I. Parking lot instance segmentation from satellite imagery through associative embeddings. In: Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems; 2019; Chicago IL USA. p. 528–31.

18. Luo Y, Han J, Liu Z, Wang M, Xia G. An elliptic centerness for object instance segmentation in aerial images. J Remote Sens. 2022;2022:9809505. doi:10.34133/2022/980950.
19. Matsuzaka Y, Yashiro R. AI-based computer vision techniques and expert systems. AI. 2023;4(1):289–302. doi:10.3390/ai4010013.
20. Hafiz AM, Bhat GM. A survey on instance segmentation: state of the art. Int J Multimed Inf Retr. 2020;9(3):171–89. doi:10.1007/s13735-020-00195-x.
21. Zhang H, Wang Y, Dayoub F, Sünderhauf N. Reference for ultralytics/utils/loss.py. Available from: https://docs.ultralytics.com/reference/utils/loss. [Accessed 2024].
22. Telicko J, Jakvics A. Comparative analysis of YOLOv8 and Mack-RCNN for people counting on fish-eye images. In: 2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME); 2023; Tenerife, Canary Islands, Spain. p. 1–6.
23. Wang X, Zhi M, Pan Z, Wang X. Summary of object detection based on convolutional neural network. In: Eleventh International Conference on Graphics and Image Processing (ICGIP 2019); 2020; Hangzhou, China. vol. 2020, no. 31.
24. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016; Honolulu. p. 779–88.
25. Redmon J, Farhadi A. YOLO9000: better, faster, stronger. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017; Honolulu. p. 6517–25.
26. Redmon J, Farhadi A. YOLOv3: an incremental improvement. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2018; Honolulu.
27. Roboflow Team. What is YOLOv5 instance segmentation? Available from: https://roboflow.com/model/yolov5-instance-segmentation. [Accessed 2022].
28. Jocher G. YOLOv5 by ultralytics. Available from: https://zenodo.org/records/3908560. [Accessed 2020].
29. Jocher G. Ultralytics. Available from: https://github.com/ultralytics/ultralytics. [Accessed 2023].
30. Gallagher J. Announcing roboflow train 3.0. Available from: https://blog.roboflow.com/roboflow-train-3-0. [Accessed 2023].
31. Adjetey C, Adu-Manu K. Content-based image retrieval using tesseract OCR engine and levenshtein algorithm. Int J Adv Comput Sci Appl. 2021;12(7):666–75.
32. Li Y, Yefeng Z, Doermann D. Detecting text lines in handwritten documents. In: 18th International Conference on Pattern Recognition (ICPR'06); 2006; Hong Kong, China. p. 1030–3.
33. Bugayong VE, Flores VJ, Linsangan NB. Google tesseract: optical character recognition (OCR) on HDD/SSD labels using machine vision. In: 2022 14th International Conference on Computer and Automation Engineering (ICCAE); 2022; Brisbane, Australia. p. 56–60.
34. Elleuch M, Tagougui N, Kherallah M. Deep learning for feature extraction of arabic handwritten script. In: George A, Petkov N, editors. Lecture notes in computer science. Cham: Springer; 2015. vol. 9257, p. 371–82. doi:10.1007/978-3-319-23117-4.
35. Hajihashemi V, Arab Ameri MM, Alavi Gharahbagh A, Bastanfard A. A pattern recognition based holographic graph neuron for persian alphabet recognition. In: 2020 International Conference on Machine Vision and Image Processing (MVIP); 2020; Qom, Iran. p. 1–6.
36. Hamplová A. Palmyrene computer vision project. Available from: https://universe.roboflow.com/adela-hamplova/palmyrene-tutzu. [Accessed 2023].
37. Hamplová A. PalmyreneOCR. Available from: https://github.com/adelajelinkova/PalmyreneOCR. [Accessed 2024].

Appendix A—Training Details

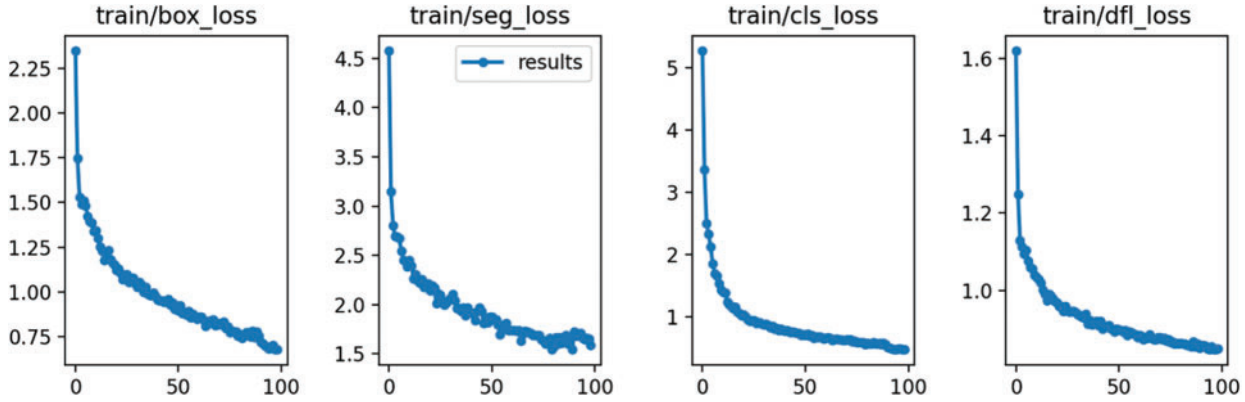

Figure A1: YOLOv8 multi-class training loss

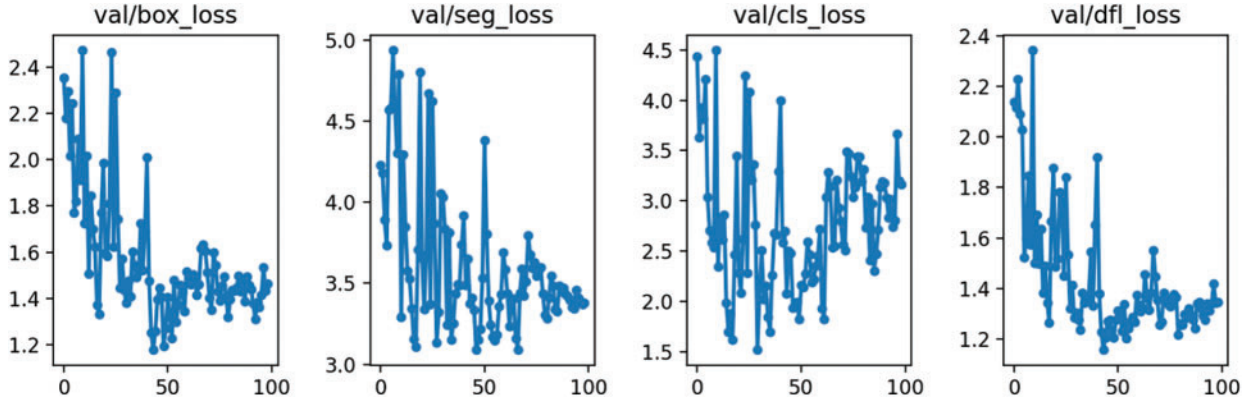

Figure A2: YOLOv8 multi-class validation loss

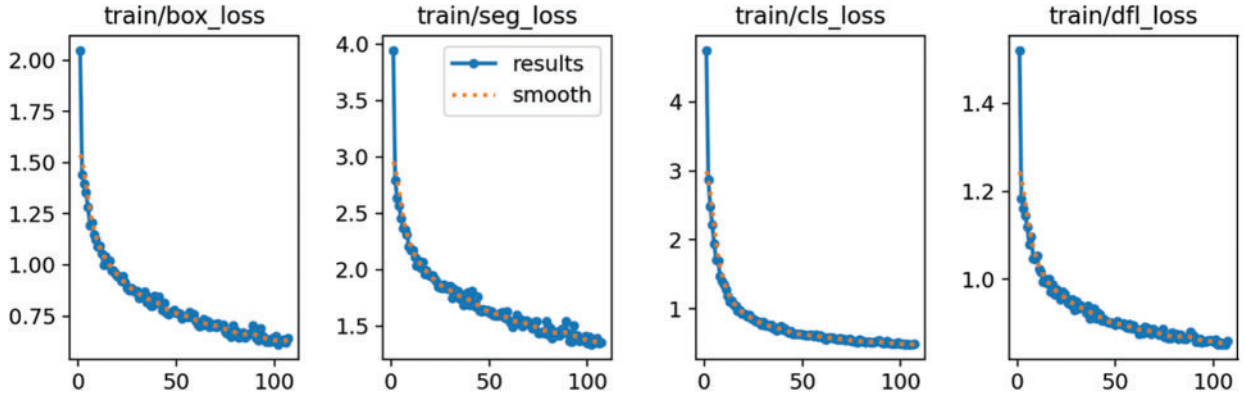

Figure A3: Roboflow 3.0 multi-class instance segmentation (accurate) training loss

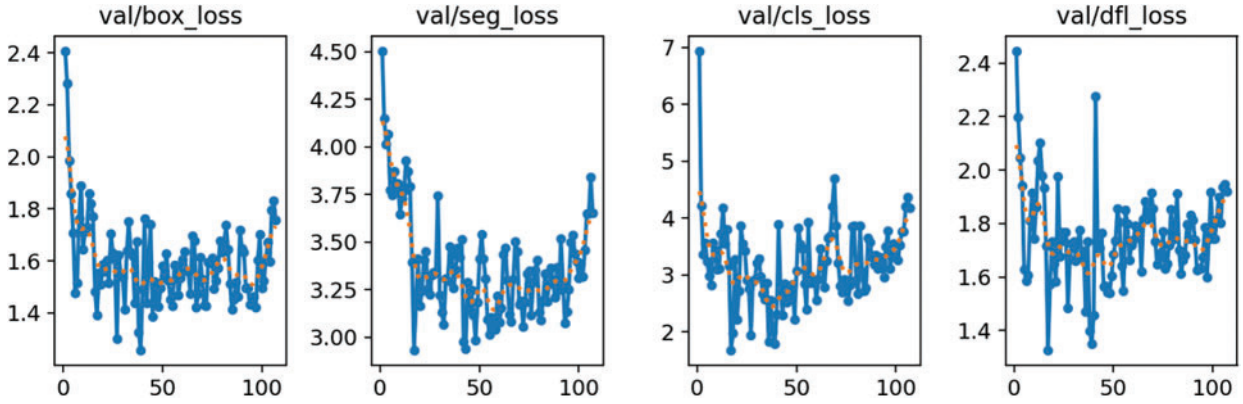

Figure A4: Roboflow 3.0 multi-class Instance segmentation (accurate) validation loss

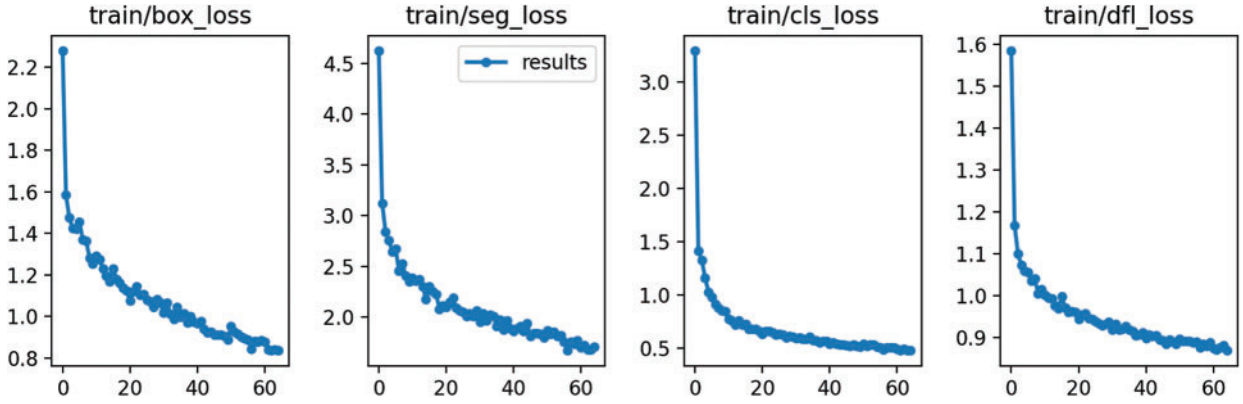

Figure A5: YOLOv8 single-class training loss

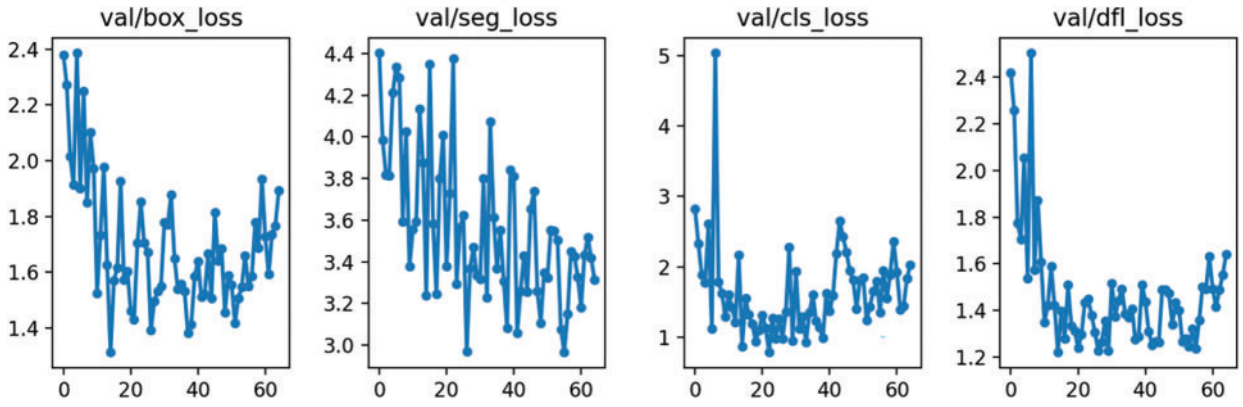

Figure A6: YOLOv8 single-class validation loss

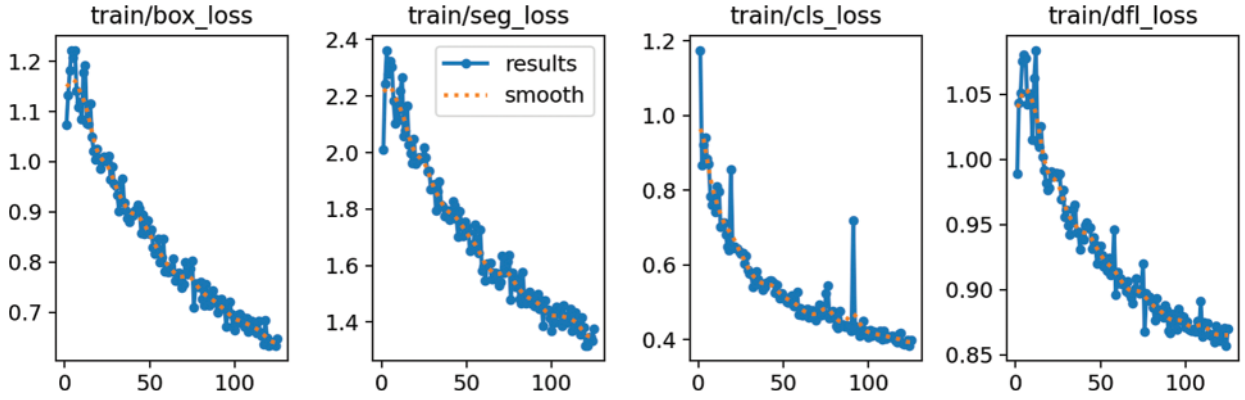

Figure A7: Roboflow 3.0 single-class Instance segmentation (accurate) training loss

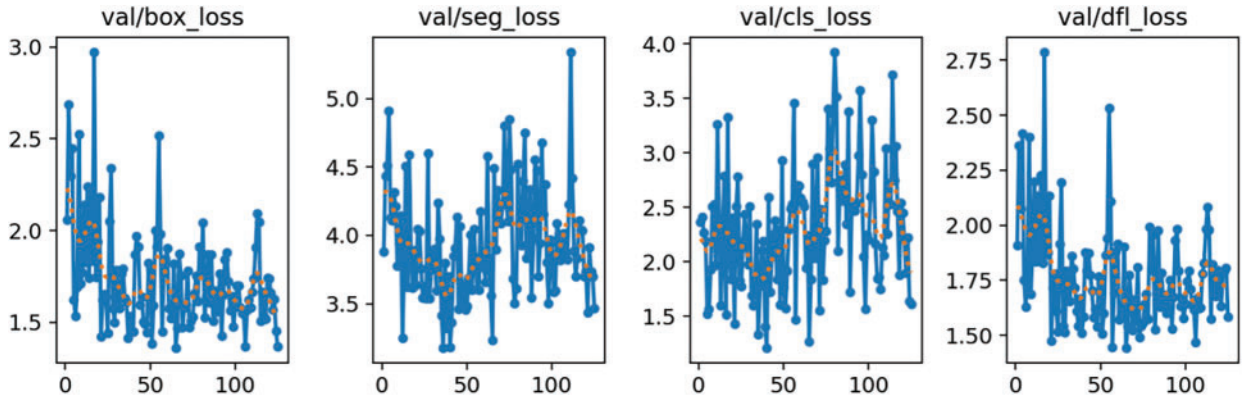

Figure A8: Roboflow 3.0 single-class Instance segmentation (accurate) validation loss

Table A1: YOLOv8 layers overview

Index	From	n	Params	Module	Arguments
0	−1	1	928	ultralytics.nn.modules.Conv	[3, 32, 3, 2]
1	−1	1	18560	ultralytics.nn.modules.Conv	[32, 64, 3, 2]
2	−1	1	29056	ultralytics.nn.modules.C2f	[64, 64, 1, True]
3	−1	1	73984	ultralytics.nn.modules.Conv	[64, 128, 3, 2]
4	−1	2	197632	ultralytics.nn.modules.C2f	[128, 128, 2, True]
5	−1	1	295424	ultralytics.nn.modules.Conv	[128, 256, 3, 2]
6	−1	2	788480	ultralytics.nn.modules.C2f	[256, 256, 2, True]
7	−1	1	1180672	ultralytics.nn.modules.Conv	[256, 512, 3, 2]
8	−1	1	1838080	ultralytics.nn.modules.C2f	[512, 512, 1, True]
9	−1	1	656896	ultralytics.nn.modules.SPPF	[512, 512, 5]
10	−1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, ‘nearest’]
11	[−1, 6]	1	0	ultralytics.nn.modules.Concat	[1]
12	−1	1	591360	ultralytics.nn.modules.C2f	[768, 256, 1]
13	−1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, ‘nearest’]

(Continued)

Table A1 (continued)

Index	From	n	Params	Module	Arguments
14	[−1, 4]	1	0	ultralytics.nn.modules.Concat	[1]
15	−1	1	148224	ultralytics.nn.modules.C2f	[384, 128, 1]
16	−1	1	147712	ultralytics.nn.modules.Conv	[128, 128, 3, 2]
17	[−1, 12]	1	0	ultralytics.nn.modules.Concat	[1]
18	−1	1	493056	ultralytics.nn.modules.C2f	[384, 256, 1]
19	−1	1	590336	ultralytics.nn.modules.Conv	[256, 256, 3, 2]
20	[−1, 9]	1	0	ultralytics.nn.modules.Concat	[1]
21	−1	1	1969152	ultralytics.nn.modules.C2f	[768, 512, 1]
22	[15, 18, 21]	1	2780606	ultralytics.nn.modules.Segment	[26, 32, 128, [128, 256, 512]]

Appendix B—Training Details

Table A2: Real texts and transcriptions of "Inv. 1438/8582, Archaeological museum of Palmyra", using all models

Real text	YOLO single-class	YOLO multi-class	Roboflow single-class	Roboflow multi-class
ʾqmʾ brt ʿzyb mlkw ḥbl	ʾ<mʾ br**ḥ** ʿzyb **ṣ**lkw **ṣ**b**z**	ʾqmʾ brt ʿzyb mlkw ḥbl	ʾ**y**mʾ br**ḥ** ʿzyb mlk**y** **ṣ**b**z**	ʾqmʾ brt **r**zyb mlkw ḥbl

Table A3: Real texts and transcriptions of "Inv. 88.AA.50, The Getty Villa Museum", using all models

Real text	YOLO single-class	YOLO multi-class	Roboflow single-class	Roboflow multi-class
mqy br mʿny	mqy br mʿn**-**	mqy br mʿny	m**y**y br m**-**ny	mqy br m**h̲**ʿny

Table A4: Real texts and transcriptions of "Inv. AO 2205, Musée du Louvre", using all models

Real text	YOLO single-class	YOLO multi-class	Roboflow single-class	Roboflow multi-class
nysn šnt [3] [100] 5 3 qbrʾ	ny**y**n šn**y** 5+1+1+1 qbrʾ ʿ**y** z**h**dbwl br	nysn šnt 5+1+1+1 **100+1** brʾ	nysn šn**ḥ** 5+1+1+1 **100**brʾ	nysn šnt 5+1+1+1 **100z1** brʾ
[d]y zbdbwl br [...]h br ʿtršwr bny kmrʾ dy lh wlbnwhy	**1 h**r ʿ**h**ršwr **y**ny k**n**mrʾ dy **th̲g** wlbn**y**h	y **1**bdbwl br h br ʿtršwr bny **n**kmrʾ dy lh wlbnw**ḥ**y	y zbdbwl br **y** br ʿ**ḥ**ršwr bn**-** kmrʾ dy **-**h wlbn**-**hy	y **1**zbdbwl br **-** br ʿtršwr bny kmrʾ dy lh wlbnw**-**y

Table A5: Real texts and transcriptions of "Inv. 95.28, The MET Museum", using all models

Real text	YOLO single-class	YOLO multi-class	Roboflow single-class	Roboflow multi-class
bṙyk šmh lʿlmʾ	bry**y**k š**ḥ**h lʿlm**mg**	bryk šmh**n** lʿlm**g**	**k**r**20**k šmh lʿlm**g**	bryk **h**mh lʿlmʾ
ṭbʾ wṙḥmnʾ ʿbd	**yk**ʾ **1n**ḥmnʾ ʿ**k**d	ṭbʾ wrḥmnʾ ʿ**r**bd	ṭ**k**ʾ **nrṣmlm** ʿ**kn**	**-**bʾ wrḥ–ʾ ʿbd
wmwdʾ ḥggw bṙ	w**w**m**20**wdʾ ḥggw **k**r **20**h**20y**kʾ bd yrḥ**20** dkʾ ʿl ḥyw**q**y	wmwdʾ ḥ**tr**w br	**nmnn**ʾ ḥgg**20 s**r **mhpk**ʾ **s**r **20nḥ20** dkʾ **1**l ḥ**2020**h**w ṣ20-**ʾ ʾb**n**h**n** **-**ʾḥ**20**h- b**20**rḥ q**ṣ20r** š**ṣ**t **n** 100 **wnww-**	wmwdʾ ḥ–w br
yhybʾ bṙ yṙḥy	**1n**ḥ**20**ʾ ʾbwhy wʾḥwhy byrḥ	yhybʾ br yrḥy		yhybʾ br **-**rḥy
dkʾ ʿl ḥywhy	q**lyw** šnt 5 100 **10 yyw–**	dkʾ**m** ʿ**-** ḥywhy		**-**kʾ ʿl ḥywh**-**
wḥyʾ ʾbwhy		wḥyʾ ʾbwhʾy		wḥyʾ ʾbwhy
wʾḥwhy byṙḥ		wʾḥwhy byrḥ		wʾḥ**ḥ**why byrḥ
qnyn šnt 5.100		qny**w** šnt 5 **r** 100		q–**r** š– 5 100
+40+3		20—**-**		20+20+1–

Table A6: Real texts and transcriptions of "Inv. 98.19.4, The MET Museum", using all models

Real text	YOLO single-class	YOLO multi-class	Roboflow single-class	Roboflow multi-class
ḥbl [ʿ]gʾ [br] zbdʿth	ʿb**t h**ʾ zbd**dq**h	ḥbl gʾ zbdʿth	**t**b**-** gʾ zbd**dq**h	**ṣ**bl gʾ zbd**d**tʾh

Table A7: Real texts and transcriptions of "Inv. 125024, The British Museum", using all models

Real text	YOLO single-class	YOLO multi-class	Roboflow single-class	Roboflow multi-class
ʾqmʾ brt ḥbzy ḥbl	ʾq**hz** br**ḥ** ḥb**np kwṣ**	ʾqm**-** brt ḥb**-**y ḥbl	**ḥ**qm**-** br**ḥ** ḥb**1**y **kwṣ**	ʾqm**q-** brt ḥbzy ḥbl **r**