Machine Vision for Recognizing the Syllabograms of Linear-A Script

ARGYRO MAVRIDAKI[1], CHRISTOS SAMOLADAS[2], IOANNIS GIACHOS[2,*],
EVANGELOS PAPAKITSOS[2], ANASTASIOS KESIDIS[3], NIKOLAOS ZACHARIS[1]
[1]Department of Informatics and Computer Engineering,
University of West Attica,
Egaleo, Athens,
GREECE

[2]Department of Industrial Design and Production Engineering,
University of West Attica,
Egaleo, Athens,
GREECE

[3]Department of Surveying & Geoinformatics Engineering,
University of West Attica,
Egaleo, Athens,
GREECE

**Corresponding Author*

Abstract: - In the past, archaeologists have attempted to use various software tools to facilitate the study and digitization of ancient languages, with varying degrees of success. The aim of this study is to develop an object detection program that is trained to recognize the syllabary of Linear-A, a Minoan (Cretan) script of the Bronze Age. Specifically, this study will focus on the 'T' series, which consists of 6 syllabograms, to determine if the use of this particular program can feasibly produce acceptable results for our case's requirements. While an optical character recognition (OCR) program was initially suggested, its performance would be sub-optimal in our case (which consists mostly of clay tablet images), so it was decided to use an object detection program instead. YOLOv5 was chosen for its low hardware requirements, quick speed and high accuracy, while also being easy to use. The study concluded that the YOLOv5 program successfully recognized the syllabograms, with a satisfactory success rate. Nevertheless, there is always room for improvement in the results or the use of a newer version of the program.

Key-Words: - Linear-A Script, Bronze Age Crete, Clay Tablets, Machine Vision, Object Detection, YOLOv5.

Received: June 4, 2025. Revised: August 18, 2025. Accepted: September 13, 2025. Published: June 5, 2026.

1 Introduction

Linear-A was a Minoan script that was used between 1800 and 1450 BCE, by the Minoans of Crete. It was first discovered by Sir Arthur Evans in 1900. It consists of syllabograms (each letter/sign corresponds to a syllable) and ideograms (symbols portraying animals, plants, or everyday tools). Although 75 syllabograms have been discovered [1], it is assumed that even more exist, while it is still considered the oldest indisputable script of Europe, although undeciphered, [2]. Over 1427 inscriptions have been found, mainly from Crete and Mycenae [3], and some even beyond the Aegean. The overwhelming majority of Linear-A script was found on clay tablets.

The purpose of this study was to find a suitable program, capable of recognizing the characters of Linear-A script. Such a program will be later integrated into an application that will assist in the study and learning of Linear-A [4]. Therefore, only a handful of the signs needed to be tested to measure the effectiveness of the program, hence the "T" series (TA, TE, TI, TO, TU, TA_2) was chosen for the geometrical simplicity of its signs (Figure 1).

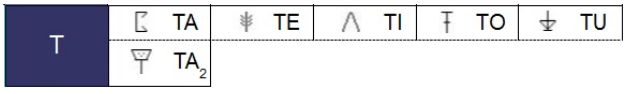

Fig. 1: The "T" series of Linear-A
Source: [4]

1.1 Related Works

The first step of this project was to gather images of the signs of Linear-A script, from relevant databases [5] and decide on the set to be used (Figure 1). Then, the next decision regarded the recognition software to be used. While the use of an optical character recognition (OCR) was first suggested, its capabilities would be limited due to the nature of our data, which are clay tablets, and thus its quirks (such as coloring or faded signs) would impede the success of the program. Ergo, the use of an OCR was abandoned in favor of an object detection program. There have been many past cases where detection programs were utilized to assist in the research of ancient scripts (especially after 2015). One such case was the use of Histogram of Oriented Gradients (HOG) in 2018, for the study of Hieroglyphs of the Egyptians, [6].

The first milestone for object detection was the creation of the "Viola-Jones algorithm", by Paul Viola and Michael Jones in 2001, [7]. It had a very basic function of recognizing people's faces in images. Later, in 2006, the "Histogram of Oriented Gradients" (or HOG) was created. Its main feature was that it divided the images into parts and repeated the recognition process for each individually, something which was also done by the programs that followed. In 2008, the "Deformable Part-based Model" (DPM) introduced a different approach to the problem, and instead of just looking for an object as a whole, it scanned the images for its distinct features (so instead of looking for a human, it tried to find arms, legs, a head, etc.). By combining its findings, it could deduce whether or not the sought-after object was present, [4].

In 2014, artificial intelligence started being used in this field. A new method, called "two-stage" recognition, separated the process into two parts, one where possible points of interest were found, and another where the program would search those areas, [8]. In 2015, the method "one-stage", which forgoes the search for points of interest, started. It could achieve quicker recognition times and with less processing power, but with less accurate results, [7]. In 2018, the "anchor-free" method started being used, which no longer used bounding boxes but instead saw the objects as points, [8].

2 Method and Results

2.1 You Only Look Once (YOLO)

The first YOLO version was introduced in 2015, [9]. It was the first object detector to utilize the one-stage detection method. Over the years, it received many updates, and new versions of it were introduced.

YOLOv5 is an open-source object detection program, developed in 2020 by Ultralytics. It was the first YOLO version that used PyTorch instead of the Darknet framework, making it more user-friendly. It has low hardware requirements coupled with high detection speed while staying on par with other object detection programs in terms of its accuracy, and therefore it was chosen for this study, [4].

During the training phase, YOLOv5 receives input in the form of images and the corresponding labels. The program will scan each image and try to find "patterns" (similarities) between the items inside the bounding boxes that have the same name, [10]. Training the system with all the images available is called an "epoch". The training process is repeated until a predefined number of epochs is reached. The data acquired from training are stored as a file, called "weights", which is a vital part of the recognition process.

During the recognition phase, YOLOv5 receives an image as input, which is then taken to the backbone, where the feature maps are created. These "maps" are, in essence, pixel grids, which are used to divide the image into smaller parts. The maps are then sent to the program's neck, where they will be combined to create three new maps, each with a different grid size (20×20, 40×40 and 80×80 pixels). The head uses these maps separately and scans each pixel in the grids for any feature that it was previously trained on using the data that is contained in the weights. Then it creates multi-dimensional arrays for each item that was recognized, with information that includes coordinates, class, and the percentage of confidence. Finally, all the data are filtered, and all the arrays that didn't pass the percentage of confidence threshold get discarded, and so we receive output only for the remaining items, [4].

2.2 LabelImg

In order for YOLOv5 to be trained, we need an image annotation program, which will be utilized to create the labels for each image that is used during training. We chose LabelImg (Figure 2), since it is open-source. To create a label, we need to draw a bounding box surrounding the sign we want to train the program to recognize, as well as to input the name of the sign. LabelImg will create a file for each image, which will include the coordinates of the bounding boxes and the associated name, as well as a file called "classes.txt", which includes the

names of all the classes we have. We need this for later.

The images will be distributed with a ratio of 80-20 to the YOLO folders named “train” and “val” respectively. If we have images with similar features (such as being a bit darker than the rest), they should not be exclusively put in the “train” or “val” folder, since doing so will confuse the algorithm. Instead, they should be distributed equally among the folders.

Another important thing that should be noted is the inclusion of images that don’t contain any item that we want to train our program to detect, but instead look alike. For example, the syllabogram TA (Figure 1) looks similar to the capital letter C. In order to avoid “false positives”, we will include images of the letter C, which will not be labeled and thus stay “null”. Such cases make up about 5% of the total number of images.

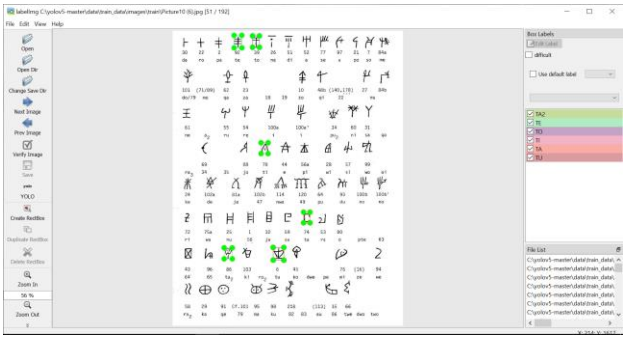

Fig. 2: The interface of LabelImg
Source: [4]

2.3 Training Process

The next step is the training process. We are using the model YOLOv5s, which is not only fast but also requires low processing power. We are starting with 150 epochs, and if the results are not satisfactory, we can increase the number for the next training. At the end of the training, a folder is created, which contains files with the graphical depiction of the training process (Figure 3 and Figure 4), a spreadsheet, and also a file with the weights.

The blue line is the result of each “epoch” separately, while the orange line is the average of the whole training process. In more detail:

- *Box loss*: It measures how successful the program is at covering an object with a box. The lower it is, the more successful the algorithm.
- *Objectness loss*: It measures the probability that a region, proposed by the algorithm, contains an object that we are looking for. The lower it is, the higher the probability.
- *Classification loss*: It measures how likely the algorithm is to give a wrong class to an object. The lower it is, the less likely a wrong class will be given to an object.
- *Precision*: It shows how accurate the algorithm’s predictions are.
- *Recall*: It shows the percentage that the algorithm managed to find all the positives in an image.
- *mAP*: It is showing us the average performance of the algorithm. It is essentially the average of the other metrics for different IoU (Intersection over union) (Figure 5). For IoU 0.5 we have mAP_0.5, while mAP_0.5:0.95 is for IoU from 0.5 to 0.95 with a step of 0.05.

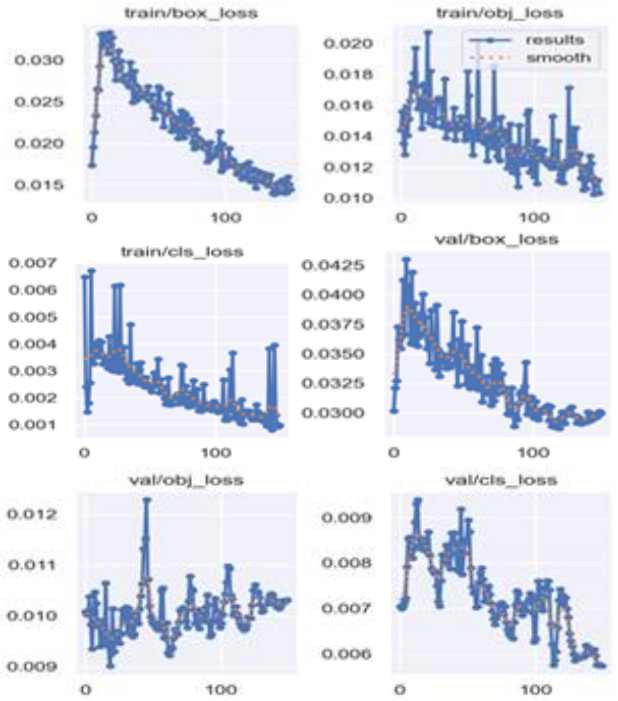

Fig. 3: The graphical depiction of the training process, which includes: box loss, objectness loss, classification loss for the data in the train folder (first row) and val folder (second row). Since they are all loss functions, the lower they are the better the results
Source: [4]

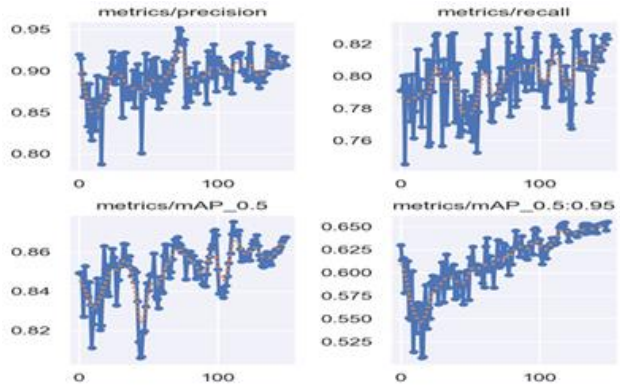

Fig. 4: The rest of the graphical depiction which includes: precision, recall (first row) and mAP (mean Average Precision) (second row)
Source: [4]

2.4 Detection Process

Now that we have the weights, we can proceed with the detection process. We will be using the weights that we received after the training and designating the location of the folder that contains the images set for detection. After the command is executed, a folder will be created, containing the modified photos with bounding boxes surrounding the recognized signs, showing the class name as well as the percentage of confidence that the recognition was correct, on a scale of 0 to 1 (Figure 6).

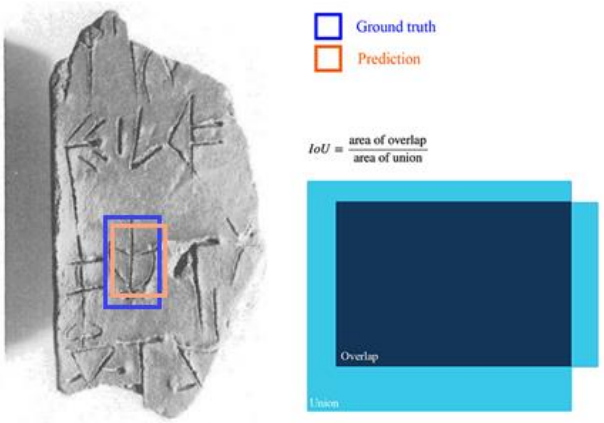

Fig. 5: IoU (Intersection over Union)
Source: [4]

There may be cases where not all the requested objects have been identified (Figure 7). We can repeat the training process with more data (photos) from the syllabogram that is not identified very successfully (i.e., TU). At the same time, we can also increase the number of epochs. The case of re-training the program, using the weights we got the first time, is not recommended in cases where only one of the requested objects does not have a satisfactory recognition rate, because it is possible to have "Overshooting" for the other data.

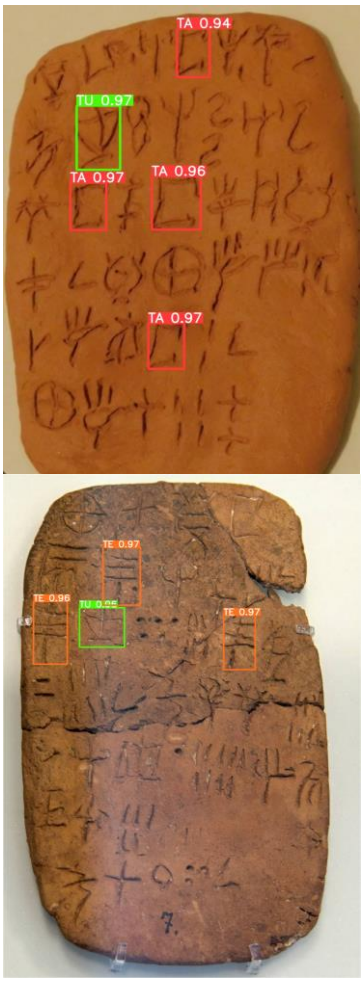

Fig. 6: The results of the program's recognition process of an image
Source: [4]

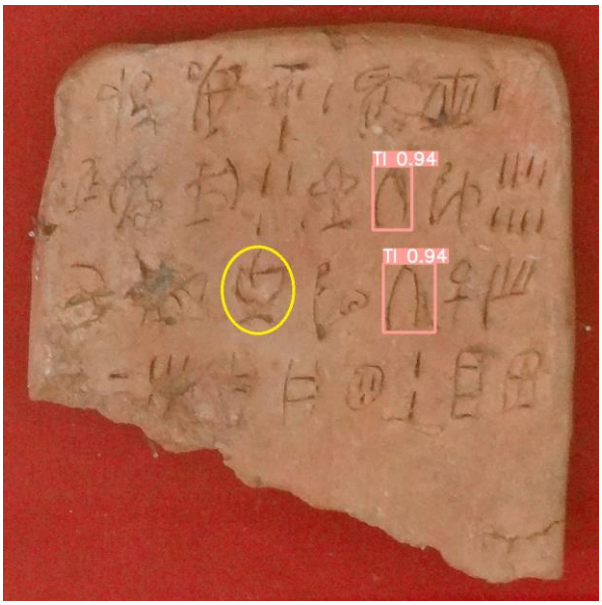

Fig. 7: The recognition of the TU sign (inside the yellow circle) was unsuccessful
Source: [4]

After testing the program on numerous images (Figure 4), it was concluded that an overall recognition rate of 92% was achieved, although the algorithm showed poor results when it came to blurry images or those that had cracks on the clay near a sign (Figure 7).

3 Conclusion

In this study, we have shown that object recognition programs can successfully identify figures on clay tablets. YOLOv5 is a suitable object recognition program for our intended application, meeting our requirements for high recognition rate, while needing low computational resources and still being supported by its developers. Additionally, there are some actions we can take to improve the recognition rate. To improve the program's training, we can provide it with more images that have irregularities, such as cracks. Moreover, using a larger model, like the YOLOv5l, can result in a more accurate outcome, provided hardware capabilities are not a concern.

The future research may follow three directions. The first one is to improve the training of the system, by providing more photographs of the signs from different sources and perspectives, and by increasing the number of epochs. The second one it to use more modern versions of the object detection program (e.g., YOLOv7), with improved performance. The final direction is to increase the number of signs processed and, eventually, develop a complete integrated software application that can be incorporated in a platform for studying and deciphering Linear-A script.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work the authors used Google Translate services in order to improve the readability and language of their manuscript. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References:

[1] Olivier J.P., Cretan Writing in the Second Millennium BC, *World Archaeology: Early Writing Systems*, Vol.17, No.3, 1986, pp. 377-389. https://doi.org/10.1080/00438243.1986.9979977.

[2] Christidis A.F., History of the Ancient Greek language (2nd reprint) [in Greek], *Institute of Modern Greek Studies*, 2010. ISBN-13: 978-960-231-113-4.

[3] Davis B., Introduction to the Aegean Pre-Alphabetic Scripts, *KUBABA*, Vol.1, 2010, pp. 38-61, [Online]. https://projectos.fcsh.unl.pt/kubaba/Davis_2010__Introduction_to_Aegean_pre-Alphabetic_Scripts.pdf (Accessed Date: January 10, 2026).

[4] Samoladas C., Development of an OCR software for the syllabograms of Cretan Protolinear Script [in Greek], *Diploma dissertation, Department of Industrial Design & Production Engineering, University of West Attica*, Greece, 2024. http://dx.doi.org/10.26265/polynoe-6131.

[5] Salgarella E. & Castellan S., "SigLA" The signs of Linear A: a paleographical database, 2020, [Online]. https://sigla.phis.me/ (Accessed Date: January 10, 2026).

[6] Hossam R., Abdel-Megeed M., Salem M. & Rimon E., Image Based Hieroglyphic Character Recognition, *The 14th International Conference on Signal Image Technology and Internet Based Systems (SITIS 2018)*, Las Palmas de Gran Canaria, Spain, 2018, pp. 32-39. https://doi.org/10.1109/SITIS.2018.00016.

[7] Zou Z., Chen K., Shi Z., Guo Y. & Ye J, Object Detection in 20 Years: A Survey, *Proceedings of the IEEE*, Vol.111, No.3, 2023, pp. 257-276. https://doi.org/10.1109/JPROC.2023.3238524

[8] Wang X., Hua Y., Mao Z., Hu G., Huang P., Hao K. & Jiao C, Small Object Detection Based on Deep Learning for Remote Sensing: A Comprehensive Review, *Remote Sensing*, Vol.15, No.13, 2023, 3265. https://doi.org/10.3390/rs15133265.

[9] Redmon J., Divvala S., Girshick R. & Farhadi A., You Only Look Once: Unified, Real-Time Object Detection, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779-788. https://doi.org/10.1109/CVPR.2016.91.

[10] Liu H., Chang F., Liu S., Meng D. & Shao W, SF-YOLOv5: A Lightweight Small Object Detection Algorithm Based on Improved Feature Fusion Mode, *Sensors*, Vol.22, No.15, 2022, 5817. https://doi.org/10.3390/s22155817

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

- Argyro Mavridaki surveyed the linguistic aspects and wrote the English text.
- Christos Samoladas created the software and carried out the simulations.
- Ioannis Giachos carried out the optimization of the software.
- Evangelos Papakitsos organized the experiments and assessed the linguistic methodology.
- Anastasios Kesidis assessed the machine vision aspects and the simulation results.
- Nikolaos Zacharis assessed the computing aspects and supervised the project.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0
https://creativecommons.org/licenses/by/4.0/deed.en_US