



DEPARTMENT OF COMPUTER SCIENCE

Ancient Languages, Modern Tools: Applying Deep Learning to Linear B

Olivia Green

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science by advanced study in Computer Science (conversion) in the Faculty of Engineering.

Submission Date: 19/09/2023

Abstract

Dating back to 1400 B.C., Linear B is the earliest known written form of Ancient Greek. In recent years, a number of ancient languages have benefitted from the potential offered by computer vision to automate manual, labour intensive tasks such as transcription and translation, and to generate new insights and perspectives into long debated areas of research. Linear B, however, has largely remained untouched in this activity. A number of challenges inhibit the application of computer vision to Linear B: it suffers from a limited data pool, dispersed over multiple sources, and a significant portion of the data available is of poor quality. This paper intends to address these challenges and presents a new dataset of Linear B's symbols, the largest and most extensive of its kind, for the purpose of facilitating further research into this field. Using this dataset, this paper also presents an image classification model based on YOLOv8 for Linear B's symbols, which achieves an accuracy of 99.47%, and is a first step towards handwritten character recognition tooling capable of automating the transcription of Linear B's tablets.

Acknowledgements

I would like to extend my deepest gratitude to my supervisor, Dr Majid Mirmehdi, whose expertise and guidance has been invaluable to me throughout the completion of this thesis. Your support has been a continuous source of motivation to me, and has enabled me to turn a lifelong interest into a research project that I have taken great joy from.

I would also like to thank Dr Sion Hannuna, for whose support and (endless!) patience I am beyond grateful. Your continuous encouragement and guidance has helped me through not only the toughest parts of this thesis, but also the completion of this Masters as a whole. I can't express my appreciation enough.

Author's Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

SIGNED: *Olivia Green*

DATE: 19/09/2023

Contents

Abstract	ii
Acknowledgements.....	iii
Author's Declaration	iv
Contents	v
1 Introduction.....	1
1.1 Background and Motivation	1
1.2 Aims and Objectives	2
2 Background and Context.....	4
2.1 Linear B	4
2.1.1 <i>The Tablets, Culture and Scribes</i>	<i>4</i>
2.1.2 <i>The Structure and Symbols of Linear B.....</i>	<i>5</i>
2.2 Artificial Intelligence	7
2.2.1 <i>Computer Vision</i>	<i>7</i>
2.2.2 <i>Convolutional Neural Networks</i>	<i>8</i>
2.2.3 <i>Handwritten Character Recognition</i>	<i>9</i>
2.3 Related Work.....	10
2.3.1 <i>Applications of HCR to Ancient Languages</i>	<i>10</i>
2.3.2 <i>Applications of HCR to Linear B.....</i>	<i>12</i>
3 YOLOv8	13
3.1 Model Architecture	13
3.2 Benefits.....	15
4 Data Selection & Collection	17
4.1 Scope.....	17
4.2 Data Selection and Sources.....	18
4.3 Data Collection	20
4.4 Final Dataset	20
4.5 Data Preparation	21
5 Project Execution	24
5.1 Setup & Tools.....	24
5.2 Training the Model	24
5.2.1 <i>Pre-Trained Model with Data Augmentation Applied.....</i>	<i>24</i>

5.2.2	<i>Un-Trained Model with Data Augmentation Applied</i>	26
5.2.3	<i>Pre-Trained Model without Data Augmentation Applied</i>	27
5.2.4	<i>Un-Trained Model without Data Augmentation Applied</i>	28
5.3	Summary of Results	29
6	Critical Evaluation	30
6.1	Review of Model Results	30
6.2	Dataset Creation and Quality	34
6.3	Testing with Tablet Data	34
7	Conclusion	36
7.1	Improvements and Further Work	36
7.1.1	<i>Expansion of the Dataset</i>	36
7.1.2	<i>Research Tooling</i>	37
7.1.3	<i>Scribal Hand Detection</i>	37
7.1.4	<i>Fully Automated Tablet Transliteration and Translation</i>	37
7.2	Summary of Outcomes	38
	Appendix A: Code for Model	39
	Appendix B: Class Data	40
	Appendix C: Architectural Diagrams	41
	Appendix D: Summary of Test Results	42
	References	45

Table of Figures

Figure 1: Linear B tablet captured with modern RTI technology [9]	2
Figure 2: Linear B tablet (historic photograph) [11]	2
Figure 3: Linear B's Syllabary [14]	6
Figure 4: Example a neural network [17]	8
Figure 5: Example of convolutional neural network	9
Figure 6: YOLOv8 Classification Model Architecture	14
Figure 7: Photograph KN 1221 [56]	19
Figure 8: Facsimile of KN 1221 [56]	19
Figure 9: Graph showing class distribution of Linear B dataset	21
Figure 10: Symbol prior to cleaning. Adapted from [56]	22
Figure 11: Symbol post cleaning. Adapted from [56]	22
Figure 12: Example augmentations applied during training. Adapted from [56].	25
Figure 13: Model loss and accuracy curves on a pre-trained instance of YOLOv8 with data augmentation applied during training	26
Figure 14: Model loss and accuracy curves on an un-trained instance of YOLOv8 with data augmentation applied during training	27
Figure 15: Model loss and accuracy curves on a pre-trained instance of YOLOv8 with no data augmentation applied during training	28
Figure 16: Model loss and accuracy curves on an un-trained instance of YOLOv8 with no data augmentation applied during training	29
Figure 17: RangeKing's diagram of YOLOv8's Object Detection model [36]	41

1 Introduction

1.1 Background and Motivation

Over the last decade, artificial intelligence (AI) has revolutionized the study of ancient languages. Its power has been harnessed to automate translation of Egyptian hieroglyphs [1], predict missing words from damaged Akkadian tablets [2], and even crack the code of undeciphered ancient languages [3]. It has demonstrated significant potential as a tool to not only support scholars through manual, labor-intensive tasks, but also to generate its own new insights into a past often shrouded with mystery. One field of study, however, which has so far received limited treatment [4] in this rapidly expanding area of scholarship is the ancient script known as Linear B. It is this script that will be the focus of this thesis.

Linear B is the earliest known written form of ancient Greek. It first arose in roughly 1400 B.C. in the Late Bronze Age, before abruptly disappearing 200 years later with the mysterious collapse of Mycenaean civilization in Greece [5]. Lost from memory for over 3000 years, its tablets were finally recovered at the beginning of the 20th century from the ruins of Knossos in Crete, by explorer and archaeologist Sir Arthur Evans. After decades of scholarship and the cumulative effort of academics including Alice Kober and Emmett L. Bennett Jr [6], Linear B was finally deciphered by Michael Ventris in 1953, ushering in a new era of study for early Greek language and civilization.

The study of Linear B brings with it a number of challenges which may explain its limited treatment so far. In comparison with its more famous siblings, Ancient Greek and Latin, it is a niche area of scholarship. It suffers from a limited data pool, with only circa 5000 tablets that remain to us today [7, p. 12], and a significant number of these being damaged or badly burnt [8, p. 21]. That image data which is available is highly variable in quality, ranging from a small number of state of the art images [9] to others that are barely legible. The spectrum of this quality is demonstrated in Figures 1 and 2. Image formats are also inconsistent (for example, some tablets are only available as drawings rather than their real life counterparts, whose originals have been lost [10, p. 76]). All these combined make any incursion into the field of AI more challenging.



Figure 1: Linear B tablet captured with modern RTI technology [9]

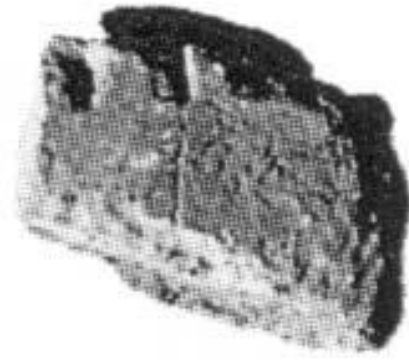


Figure 2: Linear B tablet (historic photograph) [11]

Yet in spite of this, it is a worthwhile exercise. Scholars still face all the typical challenges of analyzing ancient texts, exacerbated all the more by the issues highlighted above. Transliterating, translating and digitalizing tablets is time-consuming and laborious work – what if parts, or even all of this process could be automated? Where words or symbols are missing from damaged tablets, scholars rely on intuition and years of advanced study to fill in the gaps - what if statistical patterns in Linear B's data could be harnessed to aid their efforts? Could these patterns also be used to tell us more about the different authors that produced these tablets?

The Linear B tablets are our remaining written window into the minds of the Mycenaean people. Every step forward that we take in understanding Linear B is a step forward in understanding the Myceneans – the forefathers of a Hellenistic culture that still dominates European philosophy and thought to this day.

1.2 Aims and Objectives

Although there are ongoing efforts to create a complete digital collection of all of Linear B's tablets and their transcriptions [12], there is none available at present. Disparate source material adds unnecessary complexity to the study of Linear B and is a hurdle to those without an extensive background in the field. One impediment to the creation of a comprehensive digital corpus is the time required for tablet transcription. Transcription is an arduous manual process, requiring time and expertise from a limited pool of scholars with knowledge of Linear B.

In the context of these challenges, this thesis aims to take a first step towards automated comprehension and digitalisation of these tablets. It has two objectives: First, the creation of a quality dataset of Linear B's symbols; second, the creation of an image classification model (using this dataset) capable of accurately identifying and labelling Linear B's symbols. In meeting these objectives, a dataset of 7890 images is presented, made up of 60 different symbols from Linear B's corpus. 4 different image classification models, built on YOLOv8, are then constructed and compared, with the most successful achieving an accuracy rate of 99.47%

The remainder of this thesis is set out as follows. Chapter 2 will provide an overview of Artificial Intelligence and Linear B, and review the current state of scholarship in this area. Chapter 3 will detail and review the chosen model for this project, YOLOv8. Chapter 4 will cover the scope and creation of the dataset. Finally, the model's training and results will be described and evaluated in Chapters 5 and 6, before future work and project conclusions are set out in Chapter 7.

2 Background and Context

2.1 Linear B

2.1.1 The Tablets, Culture and Scribes

Unlike some other ancient cultures, we have no evidence of the Myceneans using any material but clay for writing Linear B (although it would be hasty to draw too many conclusions from this, given the susceptibility of less durable materials to the passage of time). This clay was shaped into tablets, inscribed with a stylus, and then left to dry in the sun as opposed to being fired, demonstrating that the Myceneans didn't write for permanence; the practice seems to have been to pulverize the old clay tablets on a yearly basis, and then re-use the material for next year's writing [8, p. 21]. It was only by sheer accident that any have been preserved for us today. In roughly the late 13th century B.C., a mysterious pattern of fires destroyed the palaces where these tablets were kept, baking and so preserving them [13, p. 21].

There are two main tablet groups: the "palm leaf" style, small, rectangular tablets covering a single record and so only one or two lines long; and the "page" style, larger tablets covering multiple records across many lines [13, p. 22]. The size of tablets varies significantly. Labels, sealings and ceramic vessels also feature, but these are not common and only very small numbers remain [8, p. 36]. The majority of tablets come from sites at Pylos and Knossos in Greece, although small quantities have also been found at a number of other sites including Mycenae, Thebes and Tiryns [8, p. 35].

Linear B's scribes seem to have had a significant pre-occupation with legibility, conveniently for Linear B's later audiences. They are often highly methodical about the way that they lay out the tablets, carving lines into the tablets and spacing out words with dividers [13, p. 21]. Nevertheless, as with any writing system where multiple authors exist, each scribe has their own peculiarities. Some scribes' writing is intricate and ornate, while others' is efficient and crude [10, p. 36]. Symbols don't always have uniform sizing across a given tablet. Identifying these styles is an important investigative tool for scholars, as it can help to find patterns and fill in the gaps where tablets are damaged or signs are written illegibly [13].

What did the Mycenaeans use Linear B for? At first glance, this might seem like a redundant question, if we consider it in the context of how we use writing today in modern English. We write letters, poetry, records, novels, laws – our uses of writing are infinite and varied. The Mycenaean psyche, however, seemed to view writing as a purely administrative exercise. The tablets are almost always records of goods, ownership or transactions, and they appear only in official governmental archives. Other forms of expression in writing are conspicuous in their absence, and there is no evidence of personal use – not even graffiti. This is a peculiar feature of Mycenaean culture, and puts it in direct contrast to other ancient societies where writing was used freely in a variety of contexts. This aspect of Linear B makes it fascinating in its monotony - administrative records don't make for particularly enthralling reading, but why did the Myceneans choose to limit themselves so thoroughly in their written expression? Perhaps other forms of expression did exist, but have simply been lost to us; perhaps there are as yet unknown cultural reasons for the Myceneans' unusual relationship with writing. Sadly, the answer is not clear [13, pp. 20-21].

2.1.2 The Structure and Symbols of Linear B

Unlike the English alphabet, Linear B is a syllabic writing system. This means that its symbols refer to combinations of sounds – typically a pairing of a vowel and a consonant, such as 'ta', 'ze' or 'ka'. These combination sounds are known as **syllabograms**, and there are 73 in total with known values, and an additional 14 which are thought to be syllabograms, but are currently untranslatable (they appear infrequently and may denote more complex syllables than the typical consonant-vowel pair) [13, p. 17]. These are shown below in Figure 3.

In addition to syllabograms, which can be thought of as the basic building blocks of Linear B, there is a further symbol group known as **ideograms**. There are roughly 170 of these, and they essentially function as shorthand words – that is, a quick way of writing out a whole word without spelling it phonetically using syllabograms. Interestingly, these ideograms do not often appear in isolation - scribes would typically spell out the word they were writing phonetically, and then write the corresponding ideogram after it. For example, a scribe writing the word "bronze" would first write *ka-ko*, then follow it with the ideogram for bronze, in a practice known as "double writing" [13, p. 18].

Numbers followed a decimal system, and were formed out of combination of 5 following symbols: 10,000, 1000, 100, 10 and 1, with the largest component number starting on the left. The final symbol groups included **weights and measures** (for marking out quantities of specific goods or items; similar to how we might say “a litre of water”) and very limited **punctuation** (many scribes would choose to mark word endings with a word-divider – a vertical stroke which is confusingly identical to the symbol for 1).

Syllabic signs					Special/unknown signs					
										
a	e	i	o	u	a ₂ (ha)	a ₃ (ai)	au			
										
da	de	di	do	du	dwe	dwo	nwa			
										
ja	je		jo		pu ₂ (phu)	pte	ra ₂ (rya)			
										
ka	ke	ki	ko	ku	ra ₃ (rai)	ro ₂ (ryo)	ta ₂ (tya)			
										
ma	me	mi	mo	mu	twe	two				
					Unknown / Doubtful values  18  19  20  34  47  49  pa ₃ ?  63  swi?  ju?  zu?  swa?  83  86  89					
na	ne	ni	no	nu						
										
pa	pe	pi	po	pu						
										
qa	qe	qi	qo							
										
ra	re	ri	ro	ru						
										
sa	se	si	so	su						
										
ta	te	ti	to	tu						
										
wa	we	wi	wo							
										
za	ze		zo							
Units of measurement										
										
Punctuation										
										
word separator	word separator	check mark								

Figure 3: Linear B's Syllabary [14]

It is important to stress that while Linear B is deciphered, in the broader sense of the word, there are elements of the language that still elude our understanding. As referenced above, a number of its symbols still have unknown values, and some of its vocabulary is untranslatable even after transcription [8, p. 21]. This is of particular relevance when defining the scope of this project, and will be discussed further in Chapter 4.

2.2 Artificial Intelligence

Artificial Intelligence (AI) refers to systems that can simulate human inference to solve complex problems or perform defined tasks. Machine learning is the process of teaching systems to acquire this human-like inference through exposure to large quantities of data, and is a field within the broader area of AI [15].

Learning can be conducted in either a supervised or unsupervised manner, depending on the problem at hand. In supervised learning, a model is provided with inputs (such as an image) labelled with a target value (in other words, label). The aim is to encourage the model to learn input features so that it can make predictions about the input that align with its target value [16, p. 191]. In unsupervised learning, data is fed into a model without any corresponding labels, with the goal that it will detect patterns in that data [16, p. 205].

2.2.1 *Computer Vision*

Computer vision is a subset of AI which teaches computers to interpret and make predictions on visual inputs such as images or videos. There are a variety of different recognition tasks contained within computer vision, but the most common include: Classification (assigning entire images to a predicted class, for example, cat or dog); detection (identifying specific locations in an image or video where objects might appear); and segmentation (separating out or “segmenting” visual inputs into their component objects) [16].

A number of different models can be employed for tackling computer vision problems, including “classical” approaches such as logistic regression or decision forests, but the most commonly used and successful models today are based on deep learning methods [16, pp. 189-190]. Deep learning is a field of machine learning which has developed significantly over the past decade thanks to rapid technological innovation and an increasing pool of training data [15]. It relies on a specific model architecture known as a neural network, inspired by

biological neurons in the human brain. The basic structure of a neural network is outlined below in Figure 4.

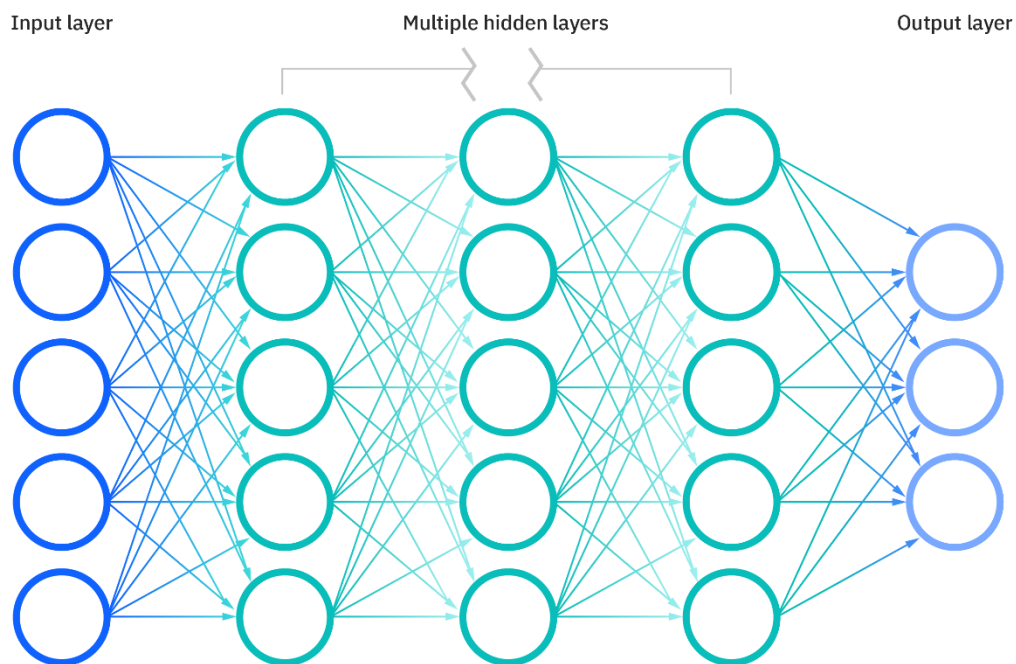


Figure 4: Example a neural network [17]

A neural network is made up of a number of interconnected artificial neurons (or nodes), which are essentially single computational units, with one or many inputs and a single output. Nodes act as switches; if the computations performed on their inputs exceed a certain limit, they generate an output which they pass on to other nodes in the chain [18]. Networks of nodes are arranged into layers: an input layer, which takes the model's inputs (for example, an image); hidden layers, which consist of multiple layers of interconnected nodes; and the output layer, which generates the model's prediction.

There many types of neural network, but for image classification problems (such as the one tackled in this thesis), convolutional neural networks (CNNs) are the most popular models, typically producing the highest rates of accuracy [16].

2.2.2 Convolutional Neural Networks

Taking inspiration from the brain's visual cortex, which is responsible for processing images, convolutional neural networks add two new layer types to their architecture [18]. The first is

the convolutional layer. Nodes in a convolutional layer have a defined range of sight; instead of having access to all pixels in a given image, they observe only a specific subset. Following convolutional layers will, in sequence, only have access to a specific group of neurons in the layer before them. As such, these layers are referred to as partially connected. These layer types are effective because they shift the network's initial focus to low-level features of an image, before subsequently arriving at the “bigger picture” view further down the line [18].

The second layer type is the pooling layer, which reduces the size the input image. The purpose of this is twofold: First, it reduces the computational cost of image processing; second, it helps the final model generalize better when making predictions on new images [18]. This means that, as the images are condensed and so too are the number of parameters that the model can learn from, the model is able to achieve a “big picture” view of a given image (and its subsequent class). This process helps to reduce a problem called overfitting – where a model learns the features of its training data too well and is unable to make accurate predictions when tested with new data [18].

CNNs can be structured in a number of ways, but a typical model might follow a pattern as shown in Figure 5.

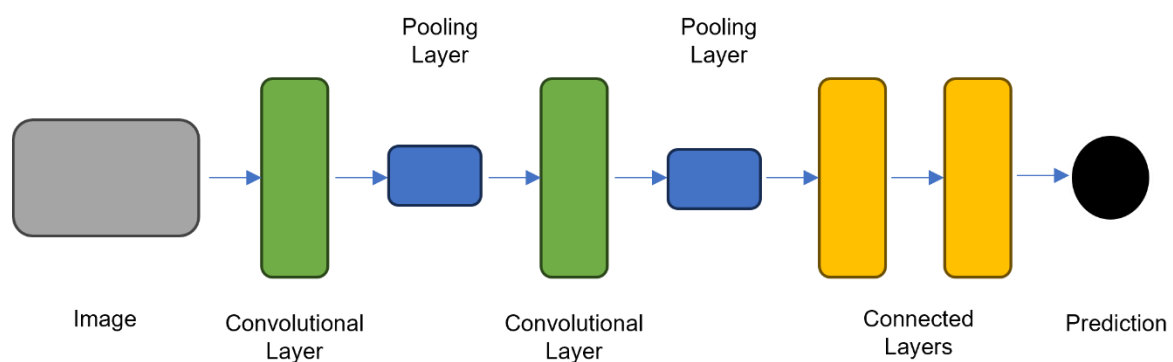


Figure 5: Example of convolutional neural network

2.2.3 Handwritten Character Recognition

Handwritten character recognition (HCR) is a field of computer science whose goal is to produce systems capable of identifying and digitalizing handmade text. It has an extensive history dating back to the 1960s, but the rise of deep learning over the last decade has seen an explosion of new research in HCR, and increasing accuracy rates of HRC systems [19]. While

the majority of research in this field still focuses on English and a handful of other widely spoken modern languages, deep learning has had a democratizing effect on HCR; the wholesale shift away from the mammoth, historic systems funded by corporations with sufficient backing to afford the complexity and computational power required, to accessible deep learning models (alongside a broadening pool of digital data and cheaper computation) has made it easier for the benefits of HCR to be applied to a broader range of languages [19].

Central to this movement has been a dataset called MNIST, which dates back to 1994 and was first presented by LeCun et al. in their seminal work on convolutional neural networks in the field of HCR [20]. MNIST comprises of 70,000 different images of handwritten numbers (0 through to 9) which are machine-ready (i.e. they do not require pre-processing before being fed into a model). It has fast become the “ground-zero” for testing and proving HCR models, and has contributed significantly to the evolution of successful models in this field [21] [22]. Recent research with MNIST focuses (for the most part) on convolutional neural networks, as both currently and historically these models have obtained the best rates of performance on MNIST [22]. Indeed, the highest rate of accuracy obtained to date (99.87%) was achieved using a CNN by Byerly et al., who used an architecture based off element-wise multiplication instead of matrix multiplication [23].

2.3 Related Work

This section will discuss the application of HCR tools and methodologies to the field of ancient languages, focusing particularly on applications involving deep learning. It will also provide a summary of related work with specific reference to Linear B.

2.3.1 Applications of HCR to Ancient Languages

Handwritten character recognition for ancient languages is, more often than not, a more complex problem to tackle when compared with their modern counterparts. Limited evidence, which can be of poor quality, and the lack of digitization of that evidence, can leave researchers with inadequate resources in their pursuit of creating performant tools in this space [4]. Nevertheless, a number of different techniques and approaches have seen great success in recent years, and following the broader trend within HCR, convolutional neural networks tend to outperform all other approaches and achieve the highest rates of accuracy [4].

It is impossible to take a “one size fits all” approach to solving the problem of HCR for ancient languages, as they vary in shape and form as much as modern languages do. Writing materials differ, ranging from papyrus to pottery to clay tablets, and some writing systems such as cuneiform (the oldest writing system in the world) densely pack symbols together across lines, making it extremely challenging to isolate and label these symbols for models to use during learning cycles [24]. Weak supervision (a collective term for a number of techniques which replace traditional approaches to labelling data with quicker, less accurate methods [25]) has seen success in approaching problems of this variety [24], but fortunately this is not a challenge which Linear B suffers from, so such approaches aren’t considered in significant detail in this thesis.

For languages that are more stylistically similar to Linear B, classification approaches which follow supervised styles of learning and which use CNNs have shown excellent results. For ancient Devanagari (an Indian script), Narang et al. achieved a classification accuracy of 93.73%, using convolutional neural network trained on 5484 samples (across 33 classes) [26]. For Ancient Greek, Swindall et al. took the innovative approach of using a crowd-sourced database of Greek characters, which generated large volumes of labelled character samples taken from old papyri [27]. They trained a number of different models on a total of 39,924 samples across 24 classes, with the best accuracy obtained on a ResNet model (92.73%). While the data volumes obtained are significantly more than could be hoped for on a corresponding Linear B dataset, a particularly interesting feature of their work was their comparison between traditional methods of HCR with CNNs. A version of Tesseract (which is a traditional OCR tool) modified for Ancient Greek was only able to obtain an accuracy of 11.5% on the same dataset, indicating just how much better suited CNNs are for this use case [27].

Symbol classification of ancient Egyptian hieroglyphs is a closely related problem space to Linear B, as, although the languages are etymologically different, there are certain visual similarities between the two scripts which make them disposed to comparison. Egyptian hieroglyphs, like Linear B’s symbols, are written as distinct from each other, are often complex and artistic in design, and variety is abundant [28]. Using a dataset of 3014 images of 40 different hieroglyphs for training, Barucci et al. compared the results of three common CNNs (ResNet-50, Inception-v3, and Xception) with their custom developed CNN, which they called

Glyphnet. Glyphnet has a more contained architecture when compared with the other models used, having a smaller number of filters, layers and parameters, and was designed to address the risk of overfitting [29]. While all models perform well, Glyphnet obtains the highest rates of accuracy (achieving 97.6%, compared to 94.5% for ResNet-50, 94.8% for Inception-v3, and 95.6% for Xception). Interestingly, all models perform better when models are trained blind without the application of transfer learning. Transfer learning is the process of applying a model previously trained on another dataset to a new problem, an approach which has been demonstrated to generally improve model accuracy and overcome challenges related to small datasets [30]. As such, the findings of Barucci et al. are unexpected. Less surprisingly, however, they show that augmentation significantly improved model performance. Data augmentations (such as random image crops or filters) are useful in scenarios where limited data is available (such as with this use case), as they can artificially increase the size and variance of a dataset, improving model performance [31]. Their findings demonstrate that this approach may well support a limited dataset of Linear B symbols, although this will be discussed further in later chapters.

2.3.2 Applications of HCR to Linear B

While no other attempts at automated recognition of Linear B's symbols have been identified during the course of this thesis, there has been one related investigation by Srivatsan et al. into the automatic detection of different scribal hands in Linear B [32], in which the authors propose a fully unsupervised neural network which aims to identify stylistic similarities between different symbols. While the aims of the study were different, the work involved production of the first (and only other, as far as the present author is aware) digital dataset of Linear B's glyphs, totaling 4,134 different images of transcriptions of symbols, that is to say, line drawings of symbols rather than photographic images of the original clay drawn symbols. As will be demonstrated in Chapter 4, the dataset produced by the present author represents a significant build on their efforts, being both created from a different source and also significantly larger (at roughly double the size).

3 YOLOv8

This section will provide to a detailed overview of YOLOv8, which is the model that will be used in this thesis. This will include a discussion its strengths and applicability to problem at hand, and a summary of its architecture.

YOLOv8 belongs to a family of models called YOLO (You Only Look Once), which date back to 2015 and are maintained by the company Ultralytics [33]. YOLOv8 is the latest of these models and was released in January 2023. Although its origins lie in object detection [34], it can perform a broad range of other vision-based tasks, including segmentation, tracking and classification. It is primarily built in Pytorch, a library used for building deep learning models [35].

3.1 Model Architecture

As the authors of YOLOv8 have yet to release an associated paper, inferences about its architecture have been made through alternative routes. A detailed architecture diagram for its object detection model has been produced and shared for Ultralytics review on YOLOv8's support pages by GitHub user RangeKing [36], which is contained within Appendix C. It is consistent with architectural descriptions published by Sharma et al [37] and Yang et al [38] in their recent publications on YOLOv8. A low level ONNX diagram for the classification model was also generated from the model's codebase and used for reference. These two diagrams, in combination with a review of the classification model's YAML config file and code have been used to generate a high level classification architecture diagram for YOLOv8. This diagram is shown in Figure 6.

YOLOv8's architecture consists primarily of a backbone and a head. The backbone refers to the body of the neural network itself, while the head produces the model's final output [39]. After images are processed following input, they are passed to the backbone, which contains a series of convolutional and C2f components whose purpose is to extract features from the images [40]. Convolutional layers, as described above in Chapter 2, are those which hone in on specific subsections of an input. The C2F component is a CSP (Cross Stage Partial) Bottleneck with additional convolutions, designed to increase model efficiency [41].

After inputs have been passed through the backbone, they are then passed into a classification function to generate class probabilities. This classification function consists of 4 different layers, which are built from various PyTorch components. Inputs are passed first to an additional convolutional layer. They are then passed to a 2D adaptive average pooling layer, the purpose of which is to reduce computational effort through down-sampling and to enhance the resistance of the output feature map to minor faults in calculation [42].

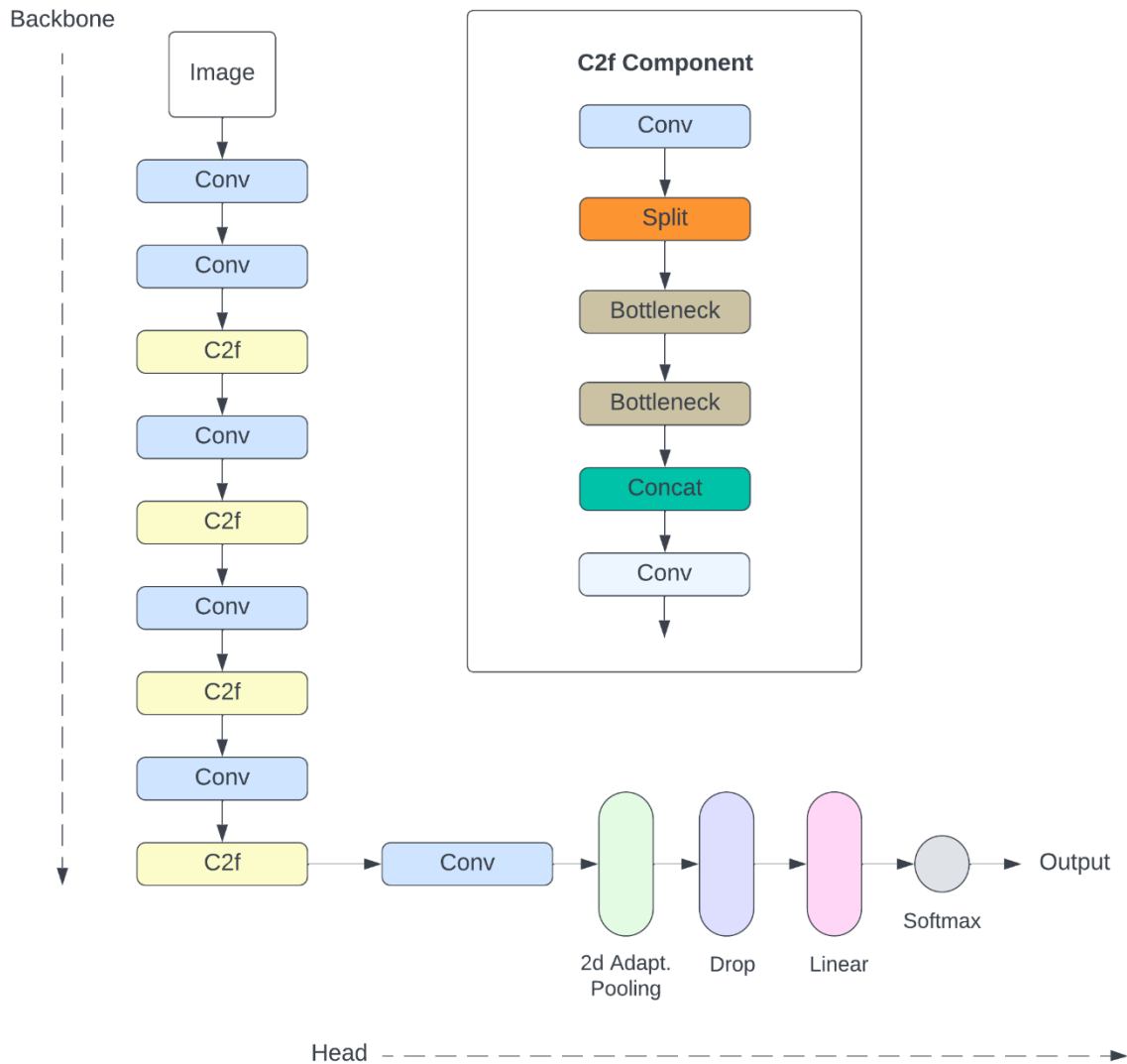


Figure 6: YOLOv8 Classification Model Architecture

Next, inputs pass through a dropout layer, which zeros randomized units of the input during the training cycle (a process which helps to reduce overfitting by encouraging the model to take a more generalized approach to learning input features) [43]. The final step is a linear layer, which generates a prediction on the input data through a calculation made with the

model's learnt weights and biases [44]. This prediction is then transformed into a class probability via Softmax, which is method for generating probabilities for multi-class classification tasks [45]. It is defined as:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

As the sum of all Softmax outputs equal one, the outputs are analogous to different probabilities. For example, given the task to generate probabilities that an input is an image of an apple, orange or pear, a Softmax function might produce the following output: 0.40 (apple); 0.30 (orange); 0.30 (pear). In practice, this means that a model has determined that the input is most likely to be an apple (as this probability is the largest of the 3 outputs).

3.2 Benefits

While there is not yet a significant amount of research available in respect of classification tasks performed on YOLOv8, initial results are extremely promising. One study, whose objective was to classify different kinds of leaves, achieved 100% accuracy using YOLOv8, surpassing all the other models the authors used for comparison (VGG16, VGG19, ResNet50, and YOLOv5) [46]. In the object detection arena, YOLOv8's results are much better documented, with a number of studies indicating impressive performance in this arena [47] [38], and given the architectural similarities between the object detection and classification models, worth referencing. These results are collectively indicative of a robust model which is capable of achieving high rates of accuracy in a range of problem spaces, making it a good starting choice for a model for the present use case.

Beyond its high performance, it boasts other features which make it an appealing. YOLOv8 provides an instance of its classification model that has been pre-trained on ImageNet data (an online repository of over 14 million images) [48], offering the potential to harness the capabilities of transfer learning. Although Barucci et al. have previously demonstrated that transfer learning decreased accuracy in the related problem space of classifying Egyptian Hieroglyphs [29], transfer learning would typically be expected to improve model accuracy, especially in respect of cases where models are trained on small datasets [30]. As such, it was

a desirable feature to test. YOLOv8 also offers a number of in-built augmentation features, which will be discussed in more detail in Chapter 5.

4 Data Selection & Collection

It is an increasingly accepted maxim of the AI community that the quality of the dataset, above all else, determines the success of the model [31]. As such, the creation of a dataset for Linear B's symbols was treated as a significant element of this thesis, and substantial effort was devoted to ensuring its quality.

This chapter will provide a detailed overview of the challenges, decision-making and processes involved in the creation of this dataset. It will consider questions of scope and feasibility, before moving on to cover the source material from which the data was created. Finally, it will provide an overview of the methods of collecting, labelling and storage, before providing a detailed summary of the final dataset.

4.1 Scope

As previously discussed, Linear B has an extensive and complex structure, with a variety of symbol groups and several hundred distinct symbols. While an ideal model would be able to handle any and all of these, restricted project time frames necessitated careful consideration of scope to ensure the feasibility of the project.

A number of symbols – and this is particularly the case with the ideograms – appear only a very limited number of times in the tablets that remain to us [49]. By itself, this is not necessarily an insurmountable challenge. Recent developments in one-shot learning show that models can provide reasonable levels of accuracy with extremely limited data [50]. In the case of Linear B, a more robust solution might be the generation of synthetic examples of symbols to achieve a more balanced dataset (shown to be one of the most effective forms of data augmentation) [31], but time constraints meant that this was not an option that could be explored in this thesis.

Relevant to the consideration of scope were challenges inherent to the nature of the language itself. For example, certain symbols have multiple meanings. The sign “,” can mean either a word divider or the number 1, while the sign “𐀓” is typically the syllabogram for “qi”, but also sometimes the ideogram for “sheep” [7, p. 15]. Context enables scholars to assign the appropriate label to these types of symbols during translation. Where a model has the benefit

of this context (in object detection models, for example), it too may be able to successfully differentiate in these cases [51]. For classification tasks, however, where the model does not have the benefit of context, differentiation is impossible.

Finally, despite the significant progress that has been made since Michael Ventris' breakthrough 70 years ago, Linear B's decipherment still remains a work in progress. 14 syllabograms and a number of ideograms still have no certain value and remain a matter of debate among scholars today [52]. As such, no meaningful class can be assigned to these symbols without invention.

Having given consideration to these constraints, the initial scope was limited to the core syllabograms (excluding those with complex or unknown values), which total 60 in number. While this meant that significant portion of the symbols were excluded, it ensured that Linear B's core and most common symbol group was represented.

4.2 Data Selection and Sources

While a number of different repositories of Linear B tablet imagery are available online, there does not exist one, unified set of high quality images of all of the tablets remaining today. Many of the photos available are old and of low quality, and so are challenging to read and work with [53]. While some recent efforts have been made to address this by applying techniques such as RTI photography and 3D scanning to a small number of tablets [9] [54], these efforts unfortunately do not go far enough to enable a classification dataset of sufficient size and quality to be able to be created. Estimates vary on the recommended number of samples per class to achieve optimal performance in classification tasks, but Shahinfar et al. demonstrate that at least 150 training samples per class are required for acceptable performance [55]. While the need for the creation of a quality digital corpus is recognized and efforts continue towards its creation, its current absence posed a challenge for the purposes of this thesis.

An alternative, significant source of images is found in facsimiles created by Sir Arthur Evans, the archaeologist responsible for first recovering Linear B's tablets from the ruins of Knossos. Published after his death in *Scripta Minoa II* [56], these are line drawings of circa 1500 of the original tablets found in Knossos, and are true-to-life copies capturing the full detail and stylistic variations (such as handwriting styles) of each tablet. Evans' degree of accuracy in

capturing the tablets is impressive [10, p. 46]), making these a reliable source of data. An example of one such facsimile, compared with its original tablet, is given below in Figures 7 and 8.



Figure 7: Photograph KN 1221 [56]

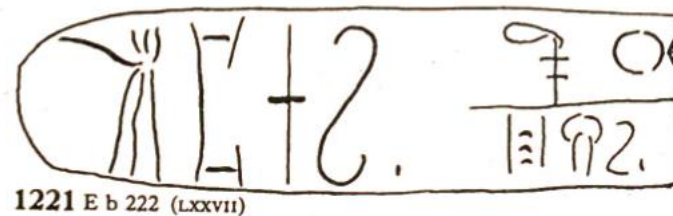


Figure 8: Facsimile of KN 1221 [56]

The facsimiles make an attractive source for a dataset for a number of reasons: They are clear, quality images from which it is straightforward to isolate specific symbols; their representations of the symbols are diverse and lifelike, making them meaningful examples for a model to learn from; finally, they are of a uniform image type and so easier to work with.

There are, however, two obvious disadvantages of building a dataset from the facsimiles. First, it is always preferable to work with “real thing” rather than copies. Copies inevitably lose some of the finer detail which may be relevant in the study of the tablets.

The second disadvantage is that using facsimiles limits the content of the dataset to only Knossian variations of the syllabograms. Tablets have been found in a number of locations across Greece, and an ideal dataset would contain full representation of symbols from all these locations. Linear B’s scribal styles do vary according to find site, as does sign usage (for example, certain syllabograms are used more commonly in Pylos tablets than they are in their Knossian counterparts) [57, p. 295]. Yet it is another interesting feature of Linear B that this

is not such significant concern as it might first appear, as Bennett outlines in his analysis on local differences in Linear B's script:

“...the Linear B script is remarkably uniform whether it is read in tablets from Knossos, or Pylos, or Mycenae, or Thebes. The texts recorded are similar in content and disposition. While personal names are not always the same, the vocabulary is shared in part, and the language is clearly identical. The same repertory of signs is used, now with fewer known exceptions, which could readily be attributed to accidents of preservation. And finally, the shapes of many signs, or of most, are markedly similar at the two principal sites.” [57, p. 295]

In addition to this, Knossos is also the largest source of Linear B tablets, with roughly 70% of all extant tablets originating from that location, and there is a significant variety in writing styles and by latest analysis, at least 43 different scribes [7, p. 199]. As such, it is theorized that a model trained on only Knossian variations of the script would still be able to generalize well on other sample from other locations.

4.3 Data Collection

Sir Arthur Evan's facsimiles are available in digital format via an online searchable corpus Linear B XYZ [58], which was used to collect the samples. Each tablet was manually reviewed to identify instances of the syllabograms in scope, and isolated screenshots were taken of these syllabograms. Given the volume of tablets and different symbol types, this was a painstaking process. Screenshots were stored in separate folders labelled according to class. Each image was labelled with its sound value and the tablet number the symbol originated from (for example, “KN 204 du”). This was to ensure traceability of the individual symbols, and reduce the possibility for any data duplication (as each symbol had a unique reference). Finally, each sample in the dataset was manually checked to remove any accidental duplication or miscategorized symbols.

4.4 Final Dataset

The final dataset included a total of 7890 samples, spread across 60 classes. The breakdown of samples per class is visualized in Figure 9 (exact class sizes are contained within Appendix B).

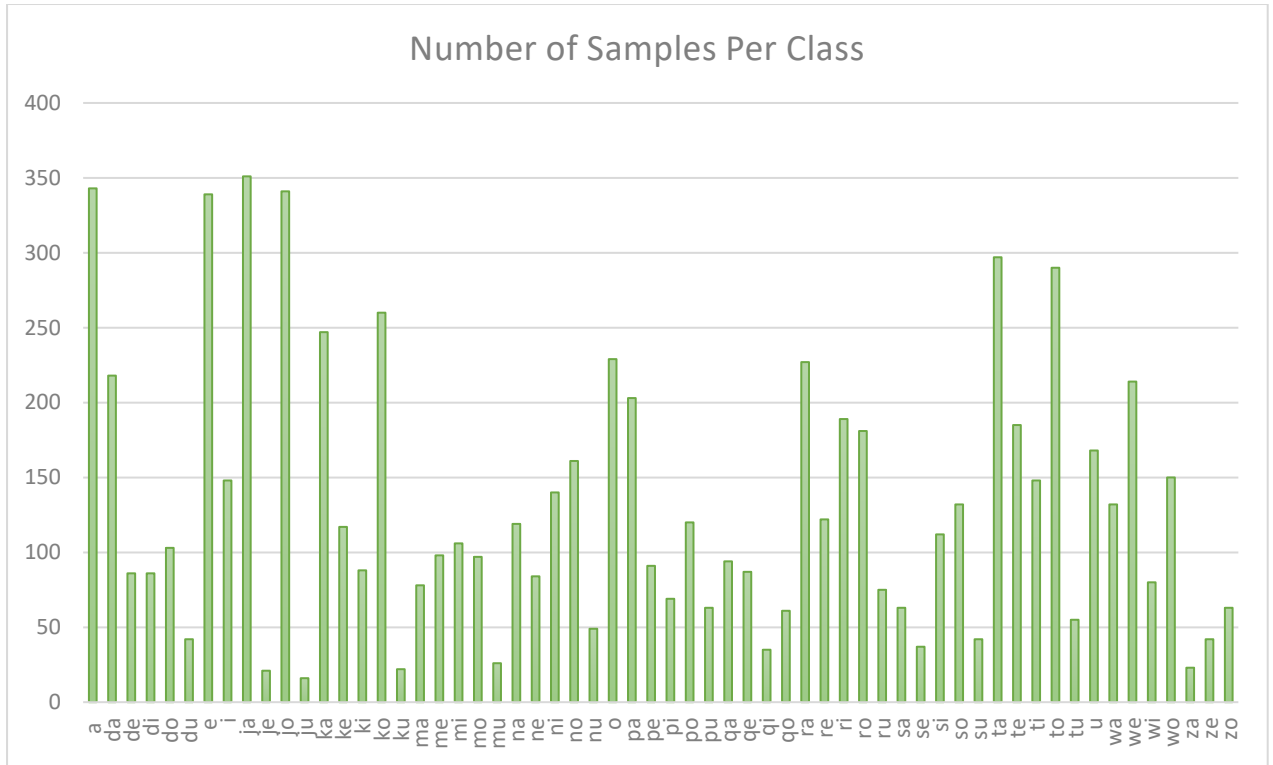


Figure 9: Graph showing class distribution of Linear B dataset

As shown, there was significant class imbalance in the dataset, meaning that the number of images per class was not equal [31]. The mean number of samples per class was 131, and many of the class sizes fell under the 150 mark recommend by Shahinfar et al. [55], which was a significant concern. The largest class (“ja”) contained 351 samples, while the smallest class (“ju”) contained just 16 samples.

4.5 Data Preparation

Given the relatively low volumes of data, ensuring the quality of the samples was paramount. To achieve this, noise (such as small scratches or elements of other symbols at the edge of the image) was removed where applicable from samples. Line dividers were left where symbols crossed over into these lines, to avoid impacting the integrity of the original symbol.

CNNs require a standardized image size for inputs and generally work better with a square image format [59]. White padding (identical in color to the sample background) was added to each sample to achieve a standardized square shape. Next, all samples were resized to a pixel size of 256x256. This resolution was large relative to the complexity of the actual data.

However, models trained on larger image sizes generally demonstrate improved performance (at the cost of increased computing resources and training time) [60]. As speed was not a particular concern for the purposes of this exercise, but low volumes of data (and the potential impact on accuracy) were, it was a worthwhile trade-off.

An example of an original sample, compared to a final sample (KN 914 a) with cleaning, white padding and resizing applied is provided below in Figures 10 and 11.



Figure 10: Symbol prior to cleaning.
Adapted from [56]



Figure 11: Symbol post cleaning. Adapted
from [56]

As a final preparatory step, the dataset was divided into training, validation and test sets. To avoid introducing any accidental bias into the dataset, a python package was used, which distributes examples randomly across the three sets [61]. A split ratio of 80:10:10 was chosen, leaving 6286 samples in the training set, 762 in the validation set, and 842 in the test set.

There were a number of additional data preparation tasks that were relevant, but not completed at this stage of the thesis. As noted above, the dataset suffered from significant class imbalance. Research demonstrates that class imbalance is detrimental to model performance in respect of convolutional neural networks [62]. The most common issue typically caused by class imbalance is over-prediction of dominant classes; in other words, smaller classes are classified as the dominant class (or classes) on the basis that the dominant class is statistically more likely to be an “accurate” prediction [63]. There are a number of ways that this issue can be addressed. For convolutional neural networks (such as YoloV8), the method that has proven most successful is oversampling (adding data to minority classes

either through duplication or the generation of additional synthetic examples), but other methods such as under-sampling (removing data from the dominant classes to achieve a better balance) or model modification (for example, adjusting cost according to class) can also be effective depending on the learning scenario and data involved [62].

For this use case, under-sampling was not an appealing option given the relatively small volumes of data collected per class. Either oversampling or model modification (or a combination of both) would have been more appropriate, allowing for maximum preservation of original data. However, neither were explored, as it was preferable to conduct a first run on original data, to determine if class imbalance would indeed affect model performance for this use case; and, if it did, to understand the nature and extent of the issue. Taking this step first would allow for the crafting of a more tailored solution should performance be affected.

Finally, augmentation was not explored at this stage. YoloV8 has a number of configurable data augmentation styles which can be implemented during runtime and, as these are comprehensive, it was preferable (and more pragmatic) to use these rather than conducting any manual data augmentation. The nature of the augmentations applied will be discussed in detail in the next chapter on model execution.

5 Project Execution

This chapter will cover the execution of the model training process. A brief overview of the tooling used during training will be provided first, before moving on to a detailed discussion of the various experiments ran. Finally, a summary of the initial results will be provided. These results will be analysed in detail in Chapter 6.

5.1 Setup & Tools

The language used for this project was Python, and the model was developed and trained in Google Colab. The code for this can be found in Appendix A. The principle metric used to measure the accuracy of the model's predictions is Top 1 accuracy. This defines an accurate prediction as one where the model output with the highest probability aligns with the correct class label. Top 5 accuracy is also tracked, which measures accuracy by determining whether any of the model's top 5 predictions align with the correct class [64].

5.2 Training the Model

As discussed in Chapters 3 and 4, YOLOv8 offers both a pre-trained and un-trained instance of its model, and a variety of adjustable parameters with respect to data augmentation. To provide a broad range of results and potential insights, 4 experiments were conducted using a mix of both the pre-trained and un-trained instances of the YOLOv8 model, first with data augmentation applied during training, and then with no data augmentation applied. Given that the dataset was limited, it was theorized that the pre-trained model with data augmentation applied during training would produce the highest rates of accuracy (and therefore determine the upper limit for accuracy), and so it was this model that was tested first.

5.2.1 Pre-Trained Model with Data Augmentation Applied

This training cycle tested the instance of the YOLOv8 model pre-trained on ImageNet data. The standard set of data augmentations offered by YOLO were applied to the training data with no adaptations. Examples of augmentations applied during training can be found below in Figure 12.



Figure 12: Example augmentations applied during training. Adapted from [56].

The number of epochs was set to 30, while the default batch size initiated by YOLO was kept at 16. An epoch is defined as the number of complete cycles that the model takes through the training data set while it is being trained, while the batch size refers to the quantity of training samples passed through the model before the model's weights and bias are updated [65]. After each epoch, the model was tested with the validation dataset (separate from the training data and not seen previously by the model) and accuracy achieved on this dataset was recorded. Figure 13 shows the loss calculated during both the training and validation cycles, and tracks the Top 1 and Top 5 accuracy of each validation cycle.

As demonstrated, the model reached high rates of accuracy extremely quickly, and the final accuracy achieved on the validation set after 30 epochs was 99.47%. However, while the training loss reduced very quickly, the loss during the validation cycle comparatively higher and did not reduce below 1.55. Loss is defined as the difference between the model's predicted output and what the correct output should have been [66]. A fully correct prediction (for example, where the model produced a softmax probability of 1.0 that an output belonged to its correct class) would generate a loss of zero. An incorrect prediction, or a prediction that was correct but with a confidence of less than 1.0, would result in a loss

greater than zero. The more wrong the prediction, the greater this number ends up being. Higher rates of loss therefore would typically be followed by a lower rate of accuracy, but this effect was not observed, with overall accuracy being very high. The potential causes for this will be discussed in further detail in Chapter 6.

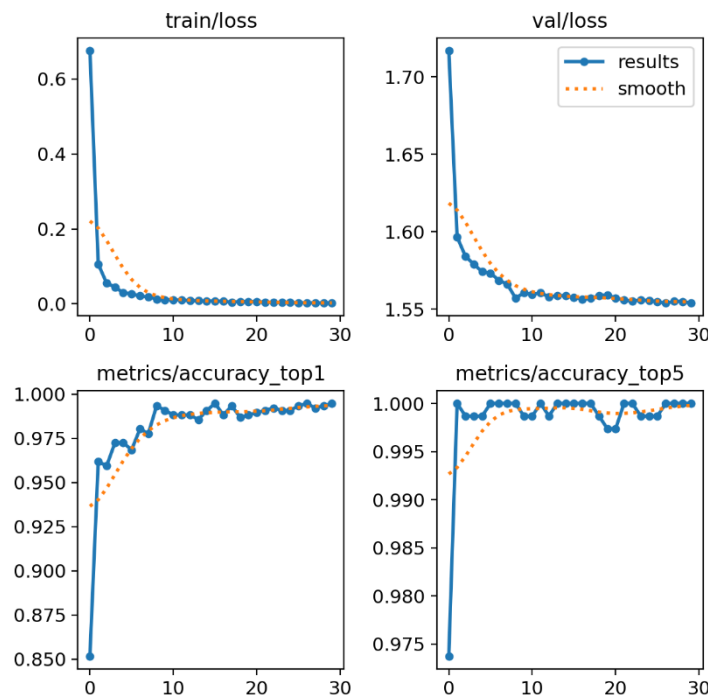


Figure 13: Model loss and accuracy curves on a pre-trained instance of YOLOv8 with data augmentation applied during training

5.2.2 Un-Trained Model with Data Augmentation Applied

This training cycle tested the un-trained instance of the YOLOv8 model. The standard set of data augmentations offered by YOLO were applied to the training data with no adaptations. As the previous training cycle had proved successful with 30 epochs, and to ensure the experiments were equalized, this training cycle was also set to 30 epochs. The batch size was kept at 16.

Similar to the previous training cycle, the model reached high rates of accuracy on the validation set quickly, although the curve is less dramatic than on the pre-trained model (which was not a surprising outcome given that this model did not have the benefit of transfer learning, and so would be expected to learn at a slower rate). However, a similar effect on the validation loss was observed. A final accuracy of 98.60% on the validation set was achieved.

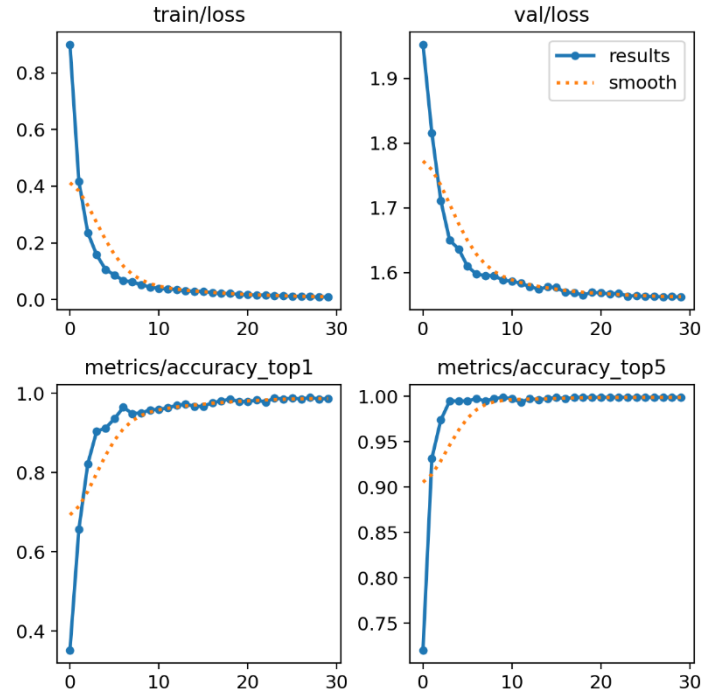


Figure 14: Model loss and accuracy curves on an un-trained instance of YOLOv8 with data augmentation applied during training

5.2.3 Pre-Trained Model without Data Augmentation Applied

This training cycle tested the instance of the YOLOv8 model pre-trained on ImageNet data. No augmentation was applied to the training data. As per the previous runs, the training cycle was set to 30 epochs. The batch size was kept at 16.

Similar effects to the previous training cycles in accuracy, speed at which accuracy was achieved and loss were observed. The final accuracy achieved on the validation set was 98.90%. Removal of augmentation from the training data seemed to have very little effect on the overall results, which was an unexpected result, given the original dataset was small. This will be discussed in more detail in Chapter 6.

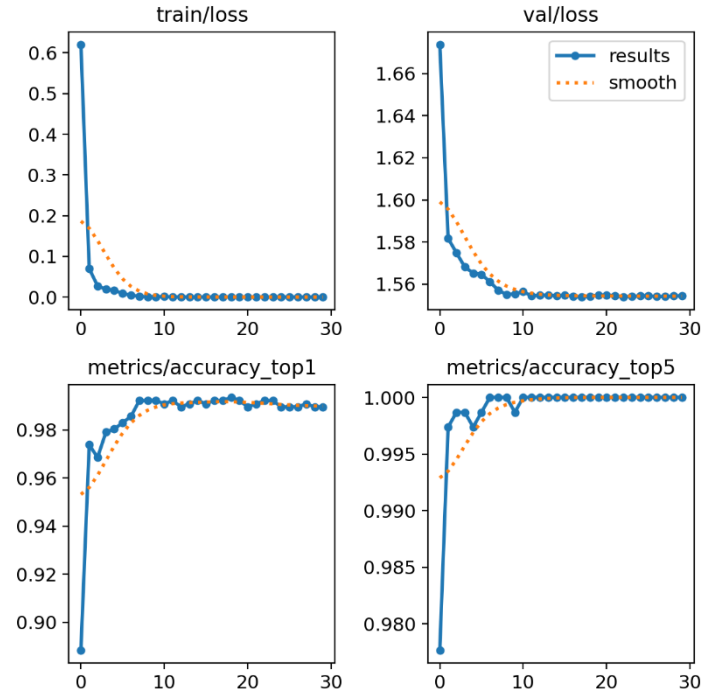


Figure 15: Model loss and accuracy curves on a pre-trained instance of YOLOv8 with no data augmentation applied during training

5.2.4 Un-Trained Model without Data Augmentation Applied

This training cycle tested the un-trained instance of the YOLOv8 model. No augmentation was applied to the training data. As per the previous runs, the training cycle was set to 30 epochs. The batch size was kept at 16.

Similar effects to the previous training cycles in accuracy, speed at which accuracy was achieved and loss were observed. The final accuracy achieved on the validation set was 98.70%. As with the previous example, removal of augmentation from the training data seemed to have very little effect on the overall results. In fact, this model marginally outperformed the other un-trained model with data augmentation applied during the training phase.

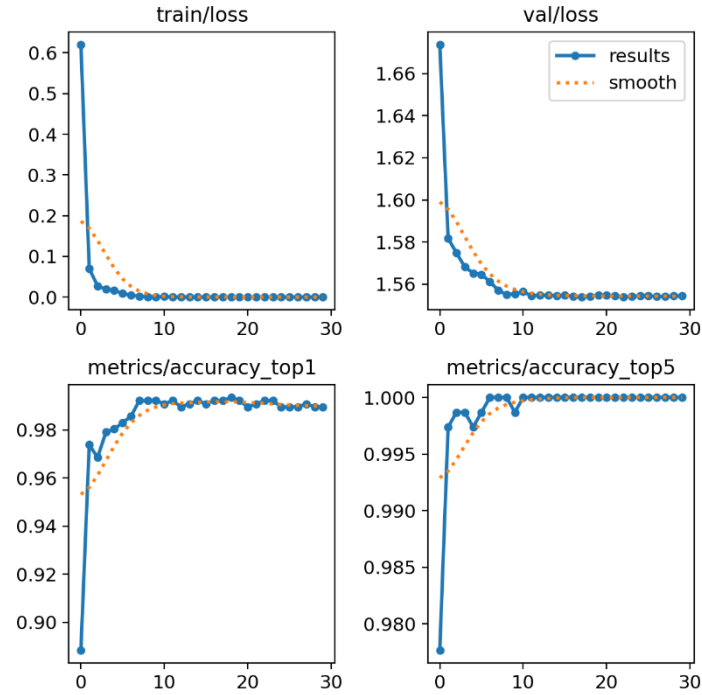


Figure 16: Model loss and accuracy curves on an un-trained instance of YOLOv8 with no data augmentation applied during training

5.3 Summary of Results

A high level comparative summary of results of the 4 different YOLOv8 models trained and tested on validation data is provided below in Table 1.

<i>Training Type</i>	<i>Final Training Loss</i>	<i>Final Validation Loss</i>	<i>Top 1 Accuracy</i>
<i>Pre-Trained & Un-augmented</i>	0.00008	1.5543	0.989
<i>Pre-Trained & Augmented</i>	0.00206	1.5541	0.9947
<i>Untrained & Augmented</i>	0.00878	1.5623	0.986
<i>Untrained & Un-augmented</i>	0.00032	1.5577	0.987

Table 1: Comparative summary of all models' final results

Across all models Top 1 accuracy was extremely high, with the best results being achieved on the pre-trained model with data augmentation applied during training. However, validation loss also remained consistently high across all models. These results, and additional testing of the model, will be evaluated in Chapter 6.

6 Critical Evaluation

6.1 Review of Model Results

Given the high levels of loss observed in training, it was necessary to test the model with new data to ensure that the model had not overfitted to the training data and was reporting misleading levels of accuracy. Furthermore, given the issues of class imbalance, it was also important to observe if there were any tendencies to over-predict any of the dominant classes. 842 test samples (containing examples from every class and as yet unseen by the model) had been reserved during the initial split of data and were used for this task. These samples were tested on all four of the trained models, and a low level analysis of each of the model's predictions was conducted.

Of all 842 samples, 60 images generated a prediction from one (or in some cases more than one) of the four models that was either a low confidence prediction or an incorrect prediction. A low confidence prediction is defined as one where the model correctly identified the class for the sample, but the probability it recorded was less than 0.90 (i.e. the model was less than 90% sure that its prediction was correct). An incorrect prediction is defined as one where the model's largest probability was for an incorrect class).

A comparative analysis of the samples was then produced to review the models' respective behaviours across each sample, evaluate performance and detect any patterns. The full analysis is contained within Appendix D, but a high level summary is provided below in Table 2.

The results demonstrated that all four models performed equally well on new data, as compared to performance during the training/validation stages. A comparison of the model accuracy on test data versus validation data indicates almost perfectly aligned performance, demonstrating that the model did not overfit when training. Overall, it can be concluded that the classification model produced is extremely successful at correctly identifying different Linear B symbols, with pre-trained models obtaining the highest rates of accuracy overall.

	Pre-Trained Model with no Data Augmentation	Pre-Trained Model with Data Augmentation	Un-Trained Model with Data Augmentation	Un-Trained Model with no Data Augmentation
Number of correct predictions defined as high confidence (>0.90)	830	816	798	818
Number of correct predictions defined as low confidence (<0.90)	4	14	27	14
Number of incorrect predictions	8	12	17	10
Model Accuracy on Test Data	99.05%	98.62%	97.99%	98.82%
Model Accuracy on Validation Data	98.90%	99.47%	98.60%	98.70%

Table 2: Comparative summary of all models' results on test data

There were also some common points of confusion across models with how they classified certain symbols. The symbols which were mis-categorized by all the models, along with a comparative example of the correct class, is shown below in Table 3.










Sample image used for testing	Class predicted by model	Correct class	Symbol of correct class	Symbol of predicted class
	i	qo		
	je	de		
	se	i		

Table 3: Symbols miscategorised across all models. Sample images adapted from [56].

These mis-categorizations are encouraging, as there is significant similarity between the incorrectly predicted symbols, and indicate that the models had learnt the symbol features well. The last sample's categorization in particular is debatable to the human eye.

One point of interest is that highest performing model on test data is the pre-trained model with no data augmentation applied during the training phase (which achieved accuracy of 99.05%). In addition to high rates of accuracy, it also demonstrates consistently higher rates of confidence in its accurate predictions (with only 4 correct predictions out of a total of 842 that could be defined as low confidence). This represents a small difference with the results demonstrated on validation data, where the highest performing model is the pre-trained model with data augmentation applied during the training phase.

The impact of transfer learning, however, on model performance is inconclusive. Improvements in accuracy obtained by it are very small, and the second highest accuracy on test data is in fact achieved by the un-trained model with no augmentation applied during training (out-performing the pre-trained model with data augmentation applied during training by 0.20%). Although this is a diversion from the findings of Barucci et al., who demonstrated that transfer learning actually led to significantly worse performance overall for classification of Egyptian Hieroglyphs (which are visually similar to Linear B) [29], the results nevertheless follow the general pattern of suggesting that transfer learning is not a key determinant of success in this problem space.

A review of the results also indicates that data augmentation appears to have been an unnecessary step (given the high levels of accuracy demonstrated when it wasn't applied), and was potentially even unhelpful to the model's ability to learn Linear B's symbols effectively. Typically, data augmentation would be expected to bolster model performance (especially in cases where training data is limited) [31]. However, the worst performing model was consistently the un-trained model with data augmentation applied (reporting an accuracy of 98.60% on validation data and 97.99% on test data). It is also the model that reports the highest rates of confusion on test data, producing more low confidence correct predictions than any other model type.

As with transfer learning, it is not possible to make definitive conclusions on the impact of augmentation, given that the highest performance achieved overall was also the pre-trained

model with data augmentation applied. It is possible however, that some of the augmentations applied during training may have been unconstructive (for example, the cropping augmentation). This is because certain Linear B symbols are very similar in shape, and the wrong kind of crop could make one almost identical to the other. Table 4 provides an example of this.

Sign	Meaning
𐀓	pa
𐀔	to

Table 4: Examples of similar Linear B symbols

Further testing would be required to determine which augmentations, if any, were least conducive to overall performance. It should be noted that overall accuracy across all models was still very high, indicating that any negative effect data augmentation did have cannot have been particularly significant.

A final point of discussion is the high levels of loss reported during the validation stage. Given the consistent rates all the models achieved on both validation and test data, it is not a particular concern, but still not an expected result. Higher rates of loss would typically mean a lower rate of accuracy, but there may be several reasons that this is not the case here. The first is that, as described above, YOLOv8 does not use any prediction thresholding to define accuracy at the validation stage. This means that a prediction classed as accurate can have a low probability associated with it (provided that probability is still the largest), meaning that accurate predictions could still generate significant amounts of loss.

The other potential cause of this is that YOLOv8 uses a cross entropy function to calculate loss. A completely incorrect prediction would therefore generate a loss of infinity (i.e. one where the model generates a softmax probability of 1.0 that the model is an incorrect class)

[67], meaning that any outliers in the dataset would significantly skew the overall loss upwards. Further testing would be required to verify this.

6.2 Dataset Creation and Quality

The results demonstrated above indicate that that Linear B dataset collected is fit for purpose and is of high quality. This meets the second objective of this thesis, which is the creation of a shareable classification dataset that can be used for further research and analysis.

However, given these levels of performance, the steps taken to clean the dataset (removing noise from the samples as discussed in Chapter 4) may have been unnecessary. With more time available, an attempt would have been made to create a further dataset out of the raw data (prior to any cleaning). This would have enabled a comparison to have been made on the performance on models on a “cleaned” dataset versus an “uncleaned” dataset, and to determine how well the model was able to identify symbols in a noisier environment.

6.3 Testing with Tablet Data

To further test the capabilities of the model, a small dataset of symbols pulled from photographic images of the tablets was created. The first group of tablets used for this purpose came from the Sir Arthur Evans archive (a collection of tablets from Knossos) [9]. The second came from the National Archeological Museum in Greece [68]. This second group represents an additional element of challenge for the model as the tablets originate from Mycenae. As discussed in Chapter 4, the training data only contains symbol variations from Knossos, and while symbol variations between regions are relatively minor, there are some local differences observed.

A total of 19 samples were created. Images were converted to greyscale prior to testing, but no other amendments were made. On a pre-trained model **with no** data augmentation applied during training, a 31.5% accuracy rate was achieved, with the model correctly classifying 6 symbols. On a pre-trained model **with** data augmentation applied during training, accuracy significantly increased, and 10 of the 19 symbols were classified correctly (an accuracy rate of 52.6%). Of the incorrect classifications that were made, many of these could be described as “intelligent” misclassifications, with the model predicting symbols that were very similar in shape to the correct symbols. A small subsection of the results achieved on tablet data is shown below in Table 5.

These results are promising and indicate that the model performs at a reasonable level in the face of a very different image type. The significant jump in accuracy achieved when using a model trained on augmented data may be because the additional noise and shading added to images as part of the augmentation process makes the model more resilient to the increased complexity of the tablet samples. Overall, it suggests that this model is the most robust in handling data types that are not aligned with the training data.








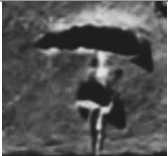


Sample image used for testing	Class predicted by model (with prediction confidence in %)	Correct class	Symbol of correct class	Symbol of predicted class (if class predicted is not correct class)
	da – 61%	da		Not applicable – prediction correct
	i – 42%	re		
	wa - 49%	wa		Not applicable – prediction correct
	pa – 75%	to		

Table 5: Subsection of results on tablet data. Images adapted from [9] and [68]

A further implication also arises from these results. This is that, as the model is able to use facsimile data to make reasonable inferences about tablet data, there may be inherent value in creating a mixed dataset of tablet data and facsimile data for any future work in this field. This approach of using a mixed dataset could help to tackle the challenges associated with the digital data available for Linear B – that low volumes and poor quality images make for a small, low quality dataset. Bolstering tablet data with facsimile data could add volume to datasets and improve the final results of any further research conducted.

7 Conclusion

To build on the initial findings presented in Chapters 5 and 6, a number of different avenues are available for further investigation. This section will provide an overview of what these potential avenues are, before concluding with a summary of the final outcomes of this thesis.

7.1 Improvements and Further Work

7.1.1 Expansion of the Dataset

The first potential improvement is the extension of the current dataset to include all of the symbols in the Linear B corpus. Some symbol types, such as the ideograms, were excluded from the original scope due to timing constraints. In some cases, symbol volumes (or lack thereof) also contributed to this decision; certain symbols only appear a limited number of times and thus gathering enough data to form a reasonably sized class was a significant concern. However, as the results of this experiment indicate, YOLOv8's classification model is able to attain a high level of accuracy even on very small class sizes. In cases where there are only one or two extant examples of particular symbols, synthetic data could be generated to address the paucity. Alternatively, models built for one-shot learning could be explored as an alternative to YOLOv8.

A further improvement that could be made to the dataset would be the inclusion of photographic images of symbols (from the original clay tablets). As discussed in Chapter 6, initial successes in testing the model with such images (as opposed to the symbols from facsimiles used to train the model), indicate that a model trained on the facsimile data alone is capable of making inferences about the original clay representations of the symbols. This is significant in that it shows that a training dataset made up of a mix of both facsimile data and photographic data could be effective. The facsimiles are thus an invaluable source of data that can be used to bolster a dataset, and to a certain extent mitigate the challenges associated with the poor quality images that make up significant portion of the digital evidence available for Linear B today.

7.1.2 Research Tooling

Additional work could also be done to produce an online interface through which interested parties could test the trained model on their own data samples, similar to that produced by Swindall et al. as part of their work on the classification of Ancient Greek symbols [27]. This would enable the model to be used as a research tool, particularly in the field classifying symbols that are partially represented or illegible due to damage or breakage in the tablets. Research has so far treated the identification of such symbols as a problem which sits within the remit of generative AI (training the models on large quantities of transliterated text from Linear B tablets, and using these trained models to generate predictions for missing or damaged symbols) [69] [70]. While this approach has significant value and has achieved positive results, treating such cases of indecipherable symbols as a problem for both computer vision and generative AI to collectively resolve would add an extra tool and a different perspective to the arsenal of a scholar grappling with these problems.

7.1.3 Scribal Hand Detection

There are also more immediate uses for the current dataset, with or without any further addition of extra classes or image types. In Chapter 2, the work of Srivatsan et al. using unsupervised learning to detect different scribal hands in Linear B was discussed [32]. While the present author's dataset is not labelled according to scribal hand, potential categorisations are available for each tablet [56]. As the originating tablet's reference is referenced in each image's file name in the dataset, it would be possible to relabel the dataset according to estimated scribal hand and thus build on the research of Srivatsan et al. Given that the current dataset is roughly twice the size of the one produced in the course of their work, it may be possible to generate a higher rate of accuracy.

7.1.4 Fully Automated Tablet Transliteration and Translation

Automated transliteration and translation is the process of extracting text from a physical medium (in Linear B's case, a tablet), presenting a digital version of that text in its original language, and then providing a corresponding translation in the viewer's native language. While this is a complex pipeline, involving a number of different technologies and processes, it would remove a significant element of burden from scholars who would otherwise be required conduct this process manually. Additional work towards a solution capable of this

would be a natural next step from the work on symbol classification presented here. This would however require a different approach and a different dataset, comprised of whole tablets rather than individual examples of symbols. An object segmentation model, such as has been used for cuneiform tablets [24], would also be required.

7.2 Summary of Outcomes

This thesis presented a new dataset of Linear B's symbols, totalling 7890 images across 60 different classes. To the best of the author's knowledge, this dataset is the largest of its kind for Linear B, and only one other dataset besides it exists. As such, this is a significant contribution to the field of applying AI to further Linear B's study, and it is hoped that this dataset can be used as a tool to support further research beyond the scope of the immediate work.

This thesis also presented a classification model based on YOLOv8, obtaining a remarkable accuracy of 99.47%. To the best of the author's knowledge, this is the first attempt of its kind for Linear B. Although the training dataset comprises of only facsimiles of the symbols, this model also demonstrated some success on photographic images, indicating an impressive ability to generalize well across very different types of symbol images.

Appendix A: Code for Model

The following code was written in Google Colab and was used for training the YOLOv8 classification models with standard augmentation applied during training. Separate adjustments to the YOLOv8 code base had to be made to switch off augmentation, which are not reflected below. This code was made with reference to training materials provided by GitHub user computervisioneng [71], whose informative training video on YOLOv8 was of great use during this thesis [72].

```
!unzip /content/TheodoraDataSplit-20230813T135509Z-001.zip

training_data = '/content/TheodoraDataSplit'

!pip install ultralytics
import os
from ultralytics import YOLO

model = YOLO("yolov8n-cls.pt") # pretrained model
results = model.train(data=training_data, epochs=30, imgsz=256)

model = YOLO("yolov8n-cls.yaml") # untrained model
results = model.train(data=training_data, epochs=30, imgsz=256)

from google.colab import drive
drive.mount('/content/gdrive')

!scp -r /content/runs '/content/gdrive/My Drive'

model = YOLO('/content/gdrive/MyDrive/Theodora/Pretrained &
Augmented/weights/last.pt')

import os
from os import listdir

# gets the path to test data
folder_dir = '/content/val'

# loops through all images in test folder and runs them through model
for subdir, dirs, files in os.walk(folder_dir):
    for file in files:
        result = model(os.path.join(subdir, file))

metrics = model.val()

metrics

!scp -r /content/runs/classify/val '/content/gdrive/My Drive/'
```

Appendix B: Class Data

CLASS	NUMBER OF IMAGES
A	343
DA	218
DE	86
DI	86
DO	103
DU	42
E	339
I	148
JA	351
JE	21
JO	341
JU	16
KA	247
KE	117
KI	88
KO	260
KU	22
MA	78
ME	98
MI	106
MO	97
MU	26
NA	119
NE	84
NI	140
NO	161
NU	49
O	229
PA	203
PE	91
PI	69
PO	120
PU	63
QA	94
QE	87
QI	35
QO	61
RA	227
RE	122
RI	189
RO	181
RU	75
SA	63
SE	37
SI	112
SO	132
SU	42
TA	297
TE	185
TI	148
TO	290
TU	55
U	168
WA	132
WE	214
WI	80
WO	150
ZA	23
ZE	42
ZO	63

Table 6 – Number of images per class

YOLOv8

Backbone
YOLOv8 Backbone (P5)

640×640×3

Conv k=3, s=2, p=1 0 P1

320×320×64×w

Conv k=3, s=2, p=1 1 P2

160×160×128×w

C2f shortcut=True, n=3×d 2 P3

160×160×128×w

Conv k=3, s=2, p=1 3 P4

80×80×256×w

C2f shortcut=True, n=6×d 4 P5

80×80×256×w Stride=8

Conv k=3, s=2, p=1 5 P6

40×40×512×w

C2f shortcut=True, n=6×d 6 P7

40×40×512×w Stride=16

Conv k=3, s=2, p=1 7 P8

20×20×512×w×r

C2f shortcut=True, n=3×d 8 P9

20×20×512×w×r Stride=32

SPPF 9

Note: height×width×channel

Head YOLOv8Head

C2f shortcut=?, n

80×80×256×w

Concat 14

80×80×256×w

Upsample 10

40×40×512×w

C2f shortcut=False, n=3×d 11 P3

40×40×512×w

Concat 12

40×40×512×w×(1+r)

Upsample 13

20×20×512×w

C2f shortcut=False, n=3×d 14 P4

20×20×512×w

Concat 15

20×20×512×w×r

Upsample 16

10×10×512×w

C2f shortcut=False, n=3×d 17 P5

10×10×512×w

Concat 18

10×10×512×w×(1+r)

Upsample 19

5×5×512×w

C2f shortcut=False, n=3×d 20 P6

5×5×512×w

Concat 21

5×5×512×w×(1+r)

Upsample 22

2.5×2.5×512×w

C2f shortcut=False, n=3×d 23 P7

2.5×2.5×512×w

Concat 24

2.5×2.5×512×w×(1+r)

Upsample 25

1.25×1.25×512×w

C2f shortcut=False, n=3×d 26 P8

1.25×1.25×512×w

Concat 27

1.25×1.25×512×w×(1+r)

Upsample 28

0.625×0.625×512×w

C2f shortcut=False, n=3×d 29 P9

0.625×0.625×512×w

Concat 30

0.625×0.625×512×w×(1+r)

Upsample 31

0.3125×0.3125×512×w

C2f shortcut=False, n=3×d 32 P10

0.3125×0.3125×512×w

Concat 33

0.3125×0.3125×512×w×(1+r)

Upsample 34

0.15625×0.15625×512×w

C2f shortcut=False, n=3×d 35 P11

0.15625×0.15625×512×w

Concat 36

0.15625×0.15625×512×w×(1+r)

Upsample 37

0.078125×0.078125×512×w

C2f shortcut=False, n=3×d 38 P12

0.078125×0.078125×512×w

Concat 39

0.078125×0.078125×512×w×(1+r)

Upsample 40

0.0390625×0.0390625×512×w

C2f shortcut=False, n=3×d 41 P13

0.0390625×0.0390625×512×w

Concat 42

0.0390625×0.0390625×512×w×(1+r)

Upsample 43

0.01953125×0.01953125×512×w

C2f shortcut=False, n=3×d 44 P14

0.01953125×0.01953125×512×w

Concat 45

0.01953125×0.01953125×512×w×(1+r)

Upsample 46

0.009765625×0.009765625×512×w

C2f shortcut=False, n=3×d 47 P15

0.009765625×0.009765625×512×w

Concat 48

0.009765625×0.009765625×512×w×(1+r)

Upsample 49

0.0048828125×0.0048828125×512×w

C2f shortcut=False, n=3×d 50 P16

0.0048828125×0.0048828125×512×w

Concat 51

0.0048828125×0.0048828125×512×w×(1+r)

Upsample 52

0.00244140625×0.00244140625×512×w

C2f shortcut=False, n=3×d 53 P17

0.00244140625×0.00244140625×512×w

Concat 54

0.00244140625×0.00244140625×512×w×(1+r)

Upsample 55

0.001220703125×0.001220703125×512×w

C2f shortcut=False, n=3×d 56 P18

0.001220703125×0.001220703125×512×w

Concat 57

0.001220703125×0.001220703125×512×w×(1+r)

Upsample 58

0.0006103515625×0.0006103515625×512×w

C2f shortcut=False, n=3×d 59 P19

0.0006103515625×0.0006103515625×512×w

Concat 60

0.0006103515625×0.0006103515625×512×w×(1+r)

Upsample 61

0.00030517578125×0.00030517578125×512×w

C2f shortcut=False, n=3×d 62 P20

0.00030517578125×0.00030517578125×512×w

Concat 63

0.00030517578125×0.00030517578125×512×w×(1+r)

Upsample 64

0.000152587890625×0.000152587890625×512×w

C2f shortcut=False, n=3×d 65 P21

0.000152587890625×0.000152587890625×512×w

Concat 66

0

41

Appendix D: Summary of Test Results

Tablet Number	Symbole	Image Ref	Pre-Trained & Unaugmented	Pre-Trained & Augmented	Un-trained & Augmented	Untrained & Unaugmented
29	de	29 de	ke 0.76, i 0.20, je 0.02, du 0.01, wo 0.00	ke 0.88, de 0.12, je 0.00, no 0.00, e 0.00	ke 0.98, de 0.02, je 0.00, du 0.00, e 0.00	je 0.80, ke 0.14, i 0.05, du 0.00, no 0.00
44	i	44 i	se 0.94, i 0.06, da 0.00, mu 0.00, ro 0.00	se 0.93, i 0.07, da 0.00, no 0.00, su 0.00	se 0.53, ne 0.44, i 0.01, no 0.01, su 0.00	se 0.98, re 0.01, di 0.00, ne 0.00, i 0.00
45	qo	45 qo	qo 0.68, ne 0.32, to 0.00, i 0.00, mu 0.00	to 0.66, ne 0.18, qo 0.11, za 0.01, ru 0.01	qo 0.91, i 0.04, to 0.04, mu 0.01, pa 0.00	qo 0.78, i 0.18, to 0.03, pa 0.00, mu 0.00
59	mu	59 mu	mu 0.65, u 0.32, se 0.03, ni 0.00, qo 0.00	u 0.98, mu 0.01, so 0.00, se 0.00, me 0.00	u 0.46, da 0.37, mu 0.06, qi 0.04, me 0.02	se 0.58, u 0.27, mu 0.09, da 0.06, su 0.00
61	ti	61 ti	ti 1.00, e 0.00, wo 0.00, za 0.00, je 0.00	ti 0.99, e 0.01, wo 0.00, pi 0.00, zo 0.00	ti 1.00, e 0.00, u 0.00, wo 0.00, je 0.00	e 0.57, ti 0.43, o 0.00, ku 0.00, wo 0.00
80	do	80 do	do 1.00, i 0.00, ri 0.00, mo 0.00, ja 0.00	do 0.83, ni 0.03, ku 0.03, jo 0.02, a 0.02	do 0.83, jo 0.08, na 0.03, so 0.01, u 0.01	do 1.00, di 0.00, ko 0.00, da 0.00, i 0.00
181	qo	181 qo	i 0.93, za 0.06, a 0.00, qo 0.00, mi 0.00	i 0.76, qo 0.14, ne 0.07, za 0.01, re 0.01	i 0.78, qo 0.10, za 0.07, ke 0.02, re 0.01	i 1.00, qo 0.00, za 0.00, no 0.00, a 0.00
192	na	192 na	na 1.00, ze 0.00, te 0.00, to 0.00, di 0.00	na 0.84, ze 0.13, do 0.01, te 0.01, ni 0.00	na 1.00, a 0.00, pa 0.00, to 0.00, di 0.00	na 1.00, te 0.00, tu 0.00, ri 0.00, jo 0.00
215	da	215 da	da 1.00, se 0.00, u 0.00, ta 0.00, ze 0.00	to 0.97, da 0.02, e 0.00, ro 0.00, wo 0.00	to 0.69, da 0.21, pa 0.06, ro 0.01, te 0.01	da 0.99, ro 0.00, su 0.00, u 0.00, to 0.00
247	pa	247 pa	pa 1.00, i 0.00, ro 0.00, qo 0.00, za 0.00	pa 0.99, ro 0.01, to 0.00, te 0.00, e 0.00	pa 0.50, ro 0.48, to 0.01, za 0.00, e 0.00	pa 1.00, ro 0.00, e 0.00, te 0.00, i 0.00
347	we	347 we	we 1.00, jo 0.00, ze 0.00, ri 0.00, ku 0.00	we 0.98, jo 0.01, ze 0.00, ku 0.00, wo 0.00	we 0.88, mo 0.04, ku 0.03, ra 0.02, do 0.01	we 0.81, wo 0.06, qe 0.03, ri 0.02, qi 0.02
358	qa	358 qa	qa 1.00, ni 0.00, ne 0.00, ru 0.00, qo 0.00	qa 0.99, qi 0.00, ru 0.00, ne 0.00, do 0.00	qa 0.96, do 0.03, qi 0.00, ni 0.00, qo 0.00	qa 0.86, qi 0.05, me 0.05, ru 0.02, ni 0.01
412	ru	412 ru	ru 0.94, ne 0.05, ko 0.01, mi 0.00, qi 0.00	ru 0.93, ne 0.07, a 0.00, ko 0.00, ka 0.00	ru 0.80, ne 0.19, ko 0.01, a 0.00, re 0.00	ru 0.92, so 0.04, a 0.03, ne 0.02, re 0.00
422	ko	422 ko	ko 0.99, qi 0.00, we 0.00, qe 0.00, ka 0.00	ko 1.00, qi 0.00, ne 0.00, ru 0.00, ka 0.00	ko 0.99, ki 0.00, tu 0.00, qe 0.00, qi 0.00	we 0.70, ko 0.16, ku 0.06, qi 0.03, jo 0.02
426	du	426 du	du 0.99, pu 0.01, pi 0.00, wa 0.00, ko 0.00	du 0.99, pu 0.01, no 0.00, ri 0.00, ju 0.00	du 0.74, pu 0.24, pi 0.01, ke 0.00, ju 0.00	du 1.00, pu 0.00, pi 0.00, je 0.00, ke 0.00
448	ta	448 ta	ta 1.00, ra 0.00, mi 0.00, wo 0.00, o 0.00	ta 0.98, ra 0.01, ri 0.00, jo 0.00, ja 0.00	ta 0.59, da 0.22, ra 0.10, su 0.03, o 0.02	ta 0.87, ra 0.10, mi 0.01, qe 0.01, we 0.00
465	ku	465 ku	ku 1.00, ka 0.00, ru 0.00, nu 0.00, so 0.00	ku 1.00, ko 0.00, me 0.00, do 0.00, a 0.00	ku 0.83, ka 0.14, qa 0.01, ko 0.01, u 0.00	ku 1.00, ki 0.00, ko 0.00, ka 0.00, qa 0.00
474	pu	474 pu	pu 1.00, wi 0.00, pi 0.00, wo 0.00, du 0.00	pu 1.00, ri 0.00, wo 0.00, wi 0.00, du 0.00	pu 0.83, wi 0.15, du 0.01, wo 0.01, ti 0.00	pu 0.99, du 0.00, wi 0.00, pi 0.00, wo 0.00
479	su	479 su	su 1.00, u 0.00, se 0.00, da 0.00, ja 0.00	su 0.97, u 0.02, se 0.01, da 0.00, ta 0.00	su 0.96, se 0.04, ja 0.00, u 0.00, da 0.00	se 0.59, su 0.32, u 0.08, nu 0.00, di 0.00
550	a	550 a 1	a 1.00, za 0.00, i 0.00, ri 0.00, u 0.00	a 0.97, za 0.01, ku 0.01, ra 0.00, ne 0.00	ma 0.37, a 0.27, ku 0.21, no 0.06, qa 0.03	a 1.00, ma 0.00, za 0.00, de 0.00, te 0.00

563	su	563 su	su 1.00, ja 0.00, ta 0.00, po 0.00, ra 0.00	su 0.98, ta 0.01, ja 0.01, po 0.00, a 0.00	su 0.81, ta 0.10, ja 0.06, po 0.02, a 0.00	su 1.00, u 0.00, ta 0.00, a 0.00, ja 0.00
571	ko	571 ko	ko 1.00, za 0.00, pu 0.00, wo 0.00, wa 0.00	ko 0.99, ru 0.00, wo 0.00, za 0.00, ri 0.00	ko 0.83, wo 0.16, qi 0.00, ri 0.00, pu 0.00	ko 1.00, qi 0.00, wo 0.00, je 0.00, za 0.00
584	wo	584 wo	wo 1.00, a 0.00, wa 0.00, ri 0.00, je 0.00	wo 1.00, ze 0.00, ri 0.00, e 0.00, ti 0.00	wo 0.99, e 0.00, ri 0.00, ki 0.00, wa 0.00	wo 0.81, e 0.16, ri 0.03, ze 0.00, pi 0.00
588	si	588 si	si 1.00, wi 0.00, wo 0.00, no 0.00, ka 0.00	si 1.00, wi 0.00, no 0.00, de 0.00, wo 0.00	si 1.00, wi 0.00, zo 0.00, se 0.00, no 0.00	si 0.70, wi 0.29, ma 0.01, ju 0.00, ka 0.00
600	so	600 so 1	so 1.00, po 0.00, mo 0.00, nu 0.00, a 0.00	so 0.93, po 0.07, a 0.00, u 0.00, su 0.00	so 0.78, po 0.22, ta 0.00, ki 0.00, a 0.00	so 1.00, po 0.00, a 0.00, ra 0.00, te 0.00
619	di	619 di 1	di 0.85, i 0.15, ne 0.00, no 0.00, na 0.00	di 0.99, i 0.01, na 0.00, jo 0.00, se 0.00	di 1.00, ne 0.00, jo 0.00, na 0.00, a 0.00	di 1.00, i 0.00, ne 0.00, wa 0.00, jo 0.00
620	ta	620 ta	we 0.49, ta 0.21, ra 0.09, da 0.08, u 0.07	ta 0.87, po 0.04, u 0.03, wo 0.02, we 0.02	da 0.53, ta 0.26, ra 0.05, su 0.03, po 0.02	ta 0.83, u 0.16, wo 0.01, we 0.00, da 0.00
627	wo	627 wo	wo 1.00, qi 0.00, je 0.00, mu 0.00, u 0.00	wo 1.00, qi 0.00, ju 0.00, je 0.00, ki 0.00	wo 0.64, ju 0.18, pu 0.11, du 0.03, ki 0.01	wo 0.98, ju 0.00, me 0.00, mu 0.00, ke 0.00
639	je	639 je	je 1.00, du 0.00, wo 0.00, ke 0.00, wa 0.00	je 1.00, ke 0.00, wo 0.00, ni 0.00, sa 0.00	je 0.80, o 0.14, wo 0.06, ke 0.00, du 0.00	je 0.99, ke 0.01, du 0.00, wo 0.00, mu 0.00
680	do	680 do	pe 0.94, do 0.05, ra 0.00, so 0.00, mi 0.00	pe 1.00, ra 0.00, do 0.00, ri 0.00, ze 0.00	do 0.84, pe 0.09, da 0.05, mo 0.01, ro 0.00	do 1.00, pe 0.00, jo 0.00, mo 0.00, ra 0.00
683	ke	683 ke	ke 1.00, de 0.00, je 0.00, ju 0.00, mu 0.00	ke 0.87, je 0.09, ni 0.03, de 0.01, me 0.00	ke 0.85, de 0.08, ni 0.06, je 0.01, o 0.00	ke 0.75, ni 0.13, de 0.13, ju 0.00, mu 0.00
685	sa	685 sa	sa 1.00, ni 0.00, je 0.00, ma 0.00, tu 0.00	sa 1.00, ni 0.00, je 0.00, ma 0.00, ke 0.00	sa 0.99, ni 0.01, je 0.00, qa 0.00, ma 0.00	ni 0.62, sa 0.38, qa 0.00, tu 0.00, ma 0.00
721	ke	721 ke	ke 0.99, je 0.01, o 0.00, e 0.00, mu 0.00	ke 0.83, o 0.15, e 0.01, je 0.01, mu 0.00	ke 0.86, o 0.13, je 0.00, nu 0.00, e 0.00	ke 0.91, je 0.06, nu 0.02, o 0.00, mi 0.00
744	sa	744 sa	sa 1.00, je 0.00, ma 0.00, ni 0.00, re 0.00	sa 1.00, ni 0.00, ma 0.00, je 0.00, mi 0.00	sa 0.88, ni 0.08, je 0.03, mi 0.01, de 0.00	ni 0.99, sa 0.01, ki 0.00, de 0.00, ma 0.00
768	ke	768 ke	ke 1.00, de 0.00, i 0.00, no 0.00, e 0.00	de 1.00, ke 0.00, za 0.00, mu 0.00, no 0.00	de 0.50, ke 0.50, je 0.00, mu 0.00, e 0.00	ke 0.99, de 0.00, qo 0.00, je 0.00, no 0.00
800	pe	800 pe	pe 1.00, mi 0.00, ze 0.00, ma 0.00, da 0.00	pe 0.85, mi 0.13, ma 0.01, sa 0.00, ze 0.00	pe 0.87, mi 0.07, ze 0.02, za 0.02, ri 0.02	pe 1.00, mi 0.00, so 0.00, ze 0.00, je 0.00
803	so	803 so	so 1.00, do 0.00, pe 0.00, qa 0.00, a 0.00	do 0.83, so 0.17, ku 0.00, pe 0.00, mi 0.00	so 0.92, do 0.07, mo 0.00, pe 0.00, jo 0.00	so 1.00, po 0.00, mo 0.00, qi 0.00, me 0.00
806	jo	806 jo	jo 0.99, na 0.01, qi 0.00, di 0.00, re 0.00	jo 0.84, na 0.16, wo 0.00, du 0.00, o 0.00	mu 0.64, du 0.16, qo 0.13, o 0.03, na 0.02	jo 0.85, mu 0.09, mo 0.01, qi 0.01, di 0.00
820	ke	820 ke	ke 1.00, i 0.00, de 0.00, no 0.00, je 0.00	de 0.98, ke 0.02, ni 0.00, ki 0.00, ma 0.00	ke 0.68, de 0.31, ni 0.00, ju 0.00, je 0.00	ke 0.98, mu 0.01, de 0.01, i 0.00, je 0.00
820	qe	820 qe 1	ka 0.99, qe 0.01, no 0.00, tu 0.00, ju 0.00	qe 0.89, ka 0.11, ku 0.00, du 0.00, ra 0.00	qe 0.96, ka 0.04, ru 0.00, qa 0.00, e 0.00	qe 0.79, ka 0.20, ku 0.01, qa 0.00, mo 0.00
831	ti	831 ti	ti 0.99, wo 0.00, si 0.00, pi 0.00, wi 0.00	ti 1.00, pi 0.00, wo 0.00, ze 0.00, ke 0.00	ti 0.99, pi 0.00, wi 0.00, ju 0.00, wo 0.00	ti 0.95, pi 0.04, pu 0.00, wi 0.00, wo 0.00
865	re	865 re 1	re 1.00, ne 0.00, ru 0.00, i 0.00, se 0.00	re 0.80, ne 0.19, i 0.00, se 0.00, di 0.00	re 0.96, ne 0.03, i 0.01, qo 0.00, se 0.00	re 1.00, i 0.00, ne 0.00, se 0.00, di 0.00
873	u	873 u	u 1.00, su 0.00, mu 0.00, da 0.00, se 0.00	u 0.96, po 0.02, ta 0.01, su 0.01, jo 0.00	u 0.86, su 0.06, po 0.04, me 0.01, so 0.01	u 0.99, na 0.01, su 0.00, mu 0.00, di 0.00

875	wi	875 wi 2	wi 0.99, e 0.01, pi 0.00, u 0.00, pu 0.00	e 0.57, wi 0.41, pe 0.02, u 0.00, wo 0.00	wi 0.43, e 0.38, pi 0.07, ti 0.06, si 0.04	wi 0.96, e 0.04, wo 0.00, pu 0.00, pi 0.00
891	re	891 re	re 1.00, i 0.00, ko 0.00, ri 0.00, qo 0.00	re 0.99, i 0.01, ko 0.00, ki 0.00, ne 0.00	re 0.84, i 0.07, ne 0.04, qo 0.02, mu 0.01	re 0.95, i 0.04, ta 0.00, mi 0.00, ne 0.00
902	mu	902 mu	mu 0.98, me 0.01, to 0.00, qi 0.00, na 0.00	mu 0.71, me 0.28, wo 0.00, o 0.00, mo 0.00	mu 0.85, qi 0.10, me 0.04, na 0.00, mo 0.00	mu 1.00, me 0.00, na 0.00, qo 0.00, qi 0.00
902	ne	902 ne	ru 0.89, ne 0.11, re 0.00, qi 0.00, to 0.00	ne 0.89, ru 0.11, qi 0.00, a 0.00, re 0.00	ne 1.00, ru 0.00, se 0.00, su 0.00, di 0.00	ne 0.88, ru 0.12, to 0.00, su 0.00, di 0.00
912	da	912 da	da 1.00, ra 0.00, po 0.00, ze 0.00, i 0.00	da 0.97, pa 0.01, to 0.01, ra 0.00, su 0.00	da 0.84, to 0.13, su 0.01, ze 0.00, pa 0.00	da 1.00, u 0.00, ro 0.00, su 0.00, to 0.00
912	ku	912 ku 3	mi 0.35, so 0.20, pe 0.14, tu 0.06, ko 0.06	tu 0.43, ku 0.33, so 0.18, ri 0.04, me 0.01	tu 0.66, ka 0.13, ku 0.08, so 0.06, mi 0.04	ra 0.76, mi 0.11, pe 0.04, jo 0.03, so 0.03
930	du	930 du	du 1.00, ju 0.00, wo 0.00, je 0.00, pu 0.00	wo 0.47, du 0.38, ju 0.14, qi 0.00, je 0.00	du 0.94, wo 0.03, ju 0.02, ke 0.01, na 0.00	du 1.00, jo 0.00, pu 0.00, pi 0.00, wa 0.00
946	du	946 du	du 1.00, pu 0.00, wa 0.00, jo 0.00, mo 0.00	du 0.99, ju 0.00, pu 0.00, wo 0.00, o 0.00	pu 0.77, du 0.22, ju 0.00, wo 0.00, wi 0.00	du 1.00, jo 0.00, ju 0.00, pu 0.00, ke 0.00
946	ni	946 ni	ni 1.00, de 0.00, a 0.00, ki 0.00, sa 0.00	ni 0.97, de 0.02, ki 0.01, a 0.00, me 0.00	ki 0.88, de 0.04, ni 0.03, a 0.02, ku 0.01	ni 0.51, de 0.26, a 0.20, ki 0.03, qi 0.00
990	ra	990 ra	ra 1.00, ze 0.00, da 0.00, pa 0.00, su 0.00	ra 1.00, ze 0.00, da 0.00, pe 0.00, te 0.00	ze 0.51, ra 0.47, pe 0.01, te 0.01, ri 0.00	ra 0.98, ze 0.01, te 0.00, ja 0.00, po 0.00
1250	ni	1250 ni	ni 0.81, sa 0.19, tu 0.00, de 0.00, ma 0.00	ni 1.00, ki 0.00, sa 0.00, ta 0.00, i 0.00	ni 0.65, sa 0.34, me 0.00, mi 0.00, ki 0.00	ni 1.00, sa 0.00, de 0.00, ki 0.00, tu 0.00
4401	jo	4401 jo 1	jo 1.00, mi 0.00, to 0.00, pe 0.00, na 0.00	jo 0.91, na 0.08, ki 0.00, ri 0.00, ko 0.00	jo 0.78, wa 0.17, ju 0.01, pu 0.01, ki 0.01	jo 1.00, mi 0.00, di 0.00, po 0.00, ta 0.00
4403	o	4403 o	o 1.00, ra 0.00, ta 0.00, nu 0.00, mu 0.00	o 0.70, ja 0.30, ra 0.00, ri 0.00, ta 0.00	ja 0.57, o 0.43, ta 0.00, ka 0.00, e 0.00	o 1.00, mi 0.00, ta 0.00, ku 0.00, ki 0.00
4412	o	4412 o	o 1.00, ju 0.00, ne 0.00, ma 0.00, de 0.00	o 0.95, ne 0.02, ja 0.01, ta 0.00, ma 0.00	ju 0.19, a 0.17, ja 0.15, ne 0.08, ka 0.08	o 0.83, ju 0.12, ti 0.01, ma 0.01, si 0.01
4413	ni	4413 ni 1	ni 1.00, sa 0.00, qa 0.00, a 0.00, ki 0.00	ni 1.00, sa 0.00, ki 0.00, a 0.00, ta 0.00	ni 0.63, sa 0.35, ki 0.01, a 0.00, mi 0.00	ni 1.00, sa 0.00, qa 0.00, tu 0.00, qi 0.00
4413	qe	4413 qe	qe 1.00, ka 0.00, ku 0.00, mo 0.00, we 0.00	qe 1.00, ka 0.00, ne 0.00, ja 0.00, nu 0.00	qe 1.00, ka 0.00, su 0.00, ja 0.00, we 0.00	qe 0.77, ka 0.23, we 0.00, ku 0.00, ja 0.00
4447	ze	4447 ze	ze 0.99, pe 0.00, we 0.00, ra 0.00, ja 0.00	ze 0.70, ra 0.29, nu 0.00, ma 0.00, o 0.00	ra 0.41, ze 0.37, pe 0.17, ku 0.03, mo 0.00	ze 0.96, pe 0.03, ra 0.00, ta 0.00, wo 0.00

Table 7: Table containing model results on test data

Note: Items highlighted in green are those where the model has made the correct prediction with high confidence (>0.90). Items highlighted in yellow are those where the model has made the correct prediction but with low confidence (<0.90). Items highlighted in red are those where the model has made the incorrect prediction

References

- [1] Google, “Fabricius,” 2020. [Online]. Available: <https://experiments.withgoogle.com/fabricius>. [Accessed 19 09 2023].
- [2] K. Lazar, B. Saret, A. Yehudai, W. Horowitz, N. Wasserman and G. Stanovsky, “Filling the Gaps in Ancient Akkadian Texts: A Masked Language Modelling Approach,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Punta Cana, Dominican Republic, Nov 2021. doi: 10.18653/v1/2021.emnlp-main.384.
- [3] J. Luo, Y. Cao and R. Barzilay, “Neural Decipherment via Minimum-Cost Flow: From Ugaritic to Linear B,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, Jul 2019. doi: 10.18653/v1/P19-1303.
- [4] T. Sommerschild, Y. Assael, J. Pavlopoulos, V. Stefanak, A. Senior, C. Dyer, J. Bodel, J. Prag, I. Androutsopoulos and N. d. Freitas, “Machine Learning for Ancient Languages: A Survey,” *Computational Linguistics*, p. 1–45, 2023. doi: https://doi.org/10.1162/coli_a_00481.
- [5] S. Deger-Jalkotzy, “Decline, Destruction, Aftermath,” in *The Cambridge Companion to the Aegean Bronze Age*, C. W. Shelmerdine, Ed., Cambridge, Cambridge University Press, 2008, p. 387–416.
- [6] J. Chadwick, *The Decipherment of Linear B*, 2nd ed., Cambridge: Cambridge University Press, 2014.
- [7] A. P. Judson, *The Undeciphered Signs of Linear B: Interpretation and Scribal Practices*, Cambridge : Cambridge University Press, 2020.
- [8] J. T. Hooker, *Linear B: An Introduction*, Bristol: Bloomsbury Academic, 1991.
- [9] Ashmolean Museum (Oxford University), “The Sir Arthur Evans Archive,” [Online]. Available: <https://sirarthurevans.ashmus.ox.ac.uk/collection/linearb/>. [Accessed 19 09 2023].
- [10] T. G. Palaima, “Scribes, scribal hands and palaeography,” in *A Companion to Linear B: Mycenaean Greek Texts and their World*, vol. II, Y. Duhoux and A. Morpurgo Davies, Eds., Louvain-La-Neuve, Peeters, 2014, pp. 33-136.
- [11] Hellenic Ministry of Culture, “LiBER - KN X 9831,” [Online]. Available: <https://liber.cnr.it/tablet/view/2121>. [Accessed 19 09 2023].
- [12] M. Del Frio, F. Di Filippo and F. Rougemont, “LiBER | Linear B Electronic Resources,” Institute of Heritage Science & Institute for Studies on the Mediterranean, [Online]. Available: <https://liber.cnr.it/index>. [Accessed 19 09 2023].
- [13] S. Ferrara, “Mycenaean Texts: The Linear B Tablets,” in *A Companion to the Ancient Greek Language*, E. J. Bakker, Ed., Chichester, John Wiley & Sons, 2010, pp. 11-24.

- [14] S. Ager, "Omniglot: the encyclopedia of writing systems and languages," [Online]. Available: <https://www.omniglot.com/writing/linearb.htm>. [Accessed 19 09 2023].
- [15] V. Lakshmanan, M. Görner and R. Gillard, Practical Machine Learning for Computer Vision: End-to-End Machine Learning for Images, O'Reilly Media, Inc. , 2021.
- [16] R. Szeliski, Computer Vision Algorithms and Applications, Cham: Springer, 2022.
- [17] IBM, "What are neural networks?," [Online]. Available: <https://www.ibm.com/topics/neural-networks>. [Accessed 19 09 2023].
- [18] A. Géron, Neural Networks and Deep Learning, O'Reilly Media, Inc., 2018.
- [19] J. Memon, M. Sami, R. A. Khan and M. Uddin, "Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR)," *IEEE Access*, vol. 8, pp. 142642-142668, 2020. doi: 10.1109/ACCESS.2020.3012542.
- [20] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278 - 2324, 1998. doi: 10.1109/5.726791.
- [21] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141-142, 2012. doi: 10.1109/MSP.2012.2211477.
- [22] A. Baldominos, Y. Saez and P. Isasi, "A Survey of Handwritten Character Recognition with MNIST and EMNIST," *Applied Sciences*, vol. 9, no. 15, p. 3169, 2019. doi: <https://doi.org/10.3390/app9153169>.
- [23] A. Byerly, T. Kalganova and I. Dear, "No routing needed between capsules," *Neurocomputing*, vol. 463, pp. 545-553, 2021. doi: <https://doi.org/10.1016/j.neucom.2021.08.064>.
- [24] T. Dencker, P. Klinkisch, S. M. Maul and B. Ommer, "Deep learning of cuneiform sign detection with weak supervision using transliteration alignment," *PLoS ONE*, vol. 15, no. 12, 2020. doi: <https://doi.org/10.1371/journal.pone.0243039>.
- [25] W. H. Tok, A. Bahree and S. Filipi, Practical Weak Supervision, O'Reilly Media, Inc., 2021.
- [26] S. R. Narang, M. Kumar and M. K. Jindal, "DeepNetDevanagari: a deep learning model for Devanagari ancient character recognition," *Multimedia Tools and Applications*, vol. 80, p. 20671–20686, 2021. doi: <https://doi.org/10.1007/s11042-021-10775-6>.
- [27] M. I. Swindall, G. Croisdale, C. C. Hunter, B. Keener, A. C. Williams, J. H. Brusuelas, N. Krevans, M. Selless, L. Fortson and J. F. Wallin, "Exploring Learning Approaches for Ancient Greek Character Recognition with Citizen Science Data," in *2021 IEEE 17th International Conference on eScience (eScience)*, Innsbruck, Austria, Sept 2021. doi: 10.1109/eScience51609.2021.00023.
- [28] P. Beylage, Middle Egyptian, University Park, PA: Pennsylvania State University Press, 2018.

- [29] A. Barucci, C. Cucci, M. Franci, M. Loschiavo and F. Argenti, "A Deep Learning Approach to Ancient Egyptian Hieroglyphs Classification," *IEEE Access*, vol. 9, pp. 123438-123447, 2021. doi: 10.1109/ACCESS.2021.3110082.
- [30] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang, W. Han, M. Huang, Q. Jin, Y. Lan, Y. Liu, Z. Liu, Z. Lu and X. Qiu, "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225-250, 2021. doi: <https://doi.org/10.1016/j.aiopen.2021.08.002>.
- [31] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1-48, Jul 2019. doi: <https://doi.org/10.1186/s40537-019-0197-0>.
- [32] N. Srivatsan, J. Vega, C. Skelton and T. Berg-Kirkpatrick, "Neural Representation Learning for Scribal Hands of Linear B," in *Document Analysis and Recognition – ICDAR 2021*, Lausanne, Switzerland, Sep 2021. doi: https://doi.org/10.1007/978-3-030-86159-9_23.
- [33] G. Jocher, A. Chaurasia and J. Qiu, *YOLO by Ultralytics*, 8.0.0 (2023), [Online]. Available: <https://github.com/ultralytics/ultralytics>.
- [34] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, May 2016. doi: 10.1109/CVPR.2016.91.
- [35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito and M. Rai, "PyTorch: an imperative style, high-performance deep learning library," in *NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Vancouver, Canada, Dec 2019. doi: <https://doi.org/10.48550/arXiv.1912.01703>.
- [36] RangeKing, "Brief summary of YOLOv8 model structure," 10 Jan 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics/issues/189>. [Accessed 19 09 2023].
- [37] N. Sharma, S. Baral, M. P. Paing and R. Chawuthai, "Parking Time Violation Tracking Using YOLOv8 and Tracking Algorithms," *Sensors*, vol. 13, p. 5843, Jun 2023. doi: <https://doi.org/10.3390/s23135843>.
- [38] G. Yang, J. Wang, Z. Nie, H. Yang and S. Yu, "A Lightweight YOLOv8 Tomato Detection Algorithm Combining Feature Enhancement and Attention," *Agronomy*, vol. 13, no. 7, p. 1824, Jul 2023. doi: <https://doi.org/10.3390/agronomy13071824>.
- [39] Ultralytics, "Ultralytics YOLOv5 Architecture," [Online]. Available: https://docs.ultralytics.com/yolov5/tutorials/architecture_description/. [Accessed 19 09 2023].
- [40] E. Soylu and T. Soylu, "A performance comparison of YOLOv8 models for traffic sign detection in the Robotaxi-full scale autonomous vehicle competition," *Multimedia Tools and Applications*, August 2023. doi: <https://doi.org/10.1007/s11042-023-16451-1>.

- [41] Ultralytics, "Reference for ultralytics/nn/modules/block.py," [Online]. Available: <https://docs.ultralytics.com/reference/nn/modules/block/>. [Accessed 19 09 2023].
- [42] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang and Y. Miao, "Review of Image Classification Algorithms Based on Convolutional Neural Networks," *Remote Sensing*, vol. 13, no. 22, p. 4712, Nov 2021. doi: <https://doi.org/10.3390/rs13224712>.
- [43] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," Jul 2012. doi: <https://doi.org/10.48550/arXiv.1207.0580>.
- [44] PyTorch Foundation, "Build the Neural Network," [Online]. Available: https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html. [Accessed 19 09 2023].
- [45] Google, "Multi-Class Neural Networks: Softmax," [Online]. Available: <https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax>. [Accessed 19 09 2023].
- [46] A. Islam, S. R. S. Raisa, N. H. Khan and A. I. Rifat, "A Deep Learning Approach for Classification and Segmentation of Leafy Vegetables and Diseases," in *2023 International Conference on Next-Generation Computing, IoT and Machine Learning (NCIM)*, Gazipur, Bangladesh, 2023. doi: 10.1109/NCIM59001.2023.10212506.
- [47] F. M. Talaat and H. ZainEldin, "An improved fire detection approach based on YOLO-v8 for smart cities," *Neural Computing and Applications*, vol. 35, no. 28, p. 20939–20954, 2023. doi: 10.1007/s00521-023-08809-1.
- [48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, Jun 2009. doi: 10.1109/CVPR.2009.5206848.
- [49] E. Salgarella, "A Note on the Linear A et B Ideogram AB 131/VIN(um) 'Wine' and Its Variants: References to Time Notation?," *Ktëma : Civilisations de l'Orient, de la Grèce et de Rome antiques*, vol. 45, pp. 161-172, 2020. doi: 10.3406/ktema.2020.2676.
- [50] G. Koch, R. Zemel and R. Salakhutdinov, "Siamese Neural Networks for One-shot Image Recognition," *ICML deep learning workshop*, vol. 2, 2015.
- [51] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun and A. Yuille, "The Role of Context for Object Detection and Semantic Segmentation in the Wild," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, Jun 2014. doi: 10.1109/CVPR.2014.119..
- [52] A. P. Judson, "The mystery of the Mycenaean 'labyrinth': the value of Linear B pu₂ and related signs," *Studi Micenei ed Egeo Anatolici*, vol. NS 3, pp. 53-72, 2018. doi: <https://doi.org/10.17863/CAM.7578>.

- [53] D. Nakassis, "Linear B in 3D," [Online]. Available: <https://mediterraneanworld.wordpress.com/2013/09/26/linear-b-in-3d/>. [Accessed 19 09 2023].
- [54] A. Greco, G. Flouda and E. Notti, "The PA-I-TO project," [Online]. Available: <https://www.paitoproject.it/en/home-2/>. [Accessed 19 09 2023].
- [55] S. Shahinfar, P. Meek and G. Falzon, "'How many images do I need?' Understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring," *Ecological Informatics*, vol. 57, May 2020. doi: <https://doi.org/10.48550/arXiv.2010.08186>.
- [56] A. J. Evans and J. L. Myres, *Scripta Minoa, the Written Documents of Minoan Crete, Vol. 2: With Special Reference to the Archives of Knossos*, Oxford: The Clarendon Press, 1952.
- [57] E. L. Bennett, "Some Local Differences in the Linear B Script," *Hesperia: The Journal of the American School of Classical Studies at Athens*, vol. 35, no. 4, pp. 295-309, Dec 1966. doi: <https://doi.org/10.2307/147560>.
- [58] "Linear B XYZ," [Online]. Available: <https://linearb.xyz/>. [Accessed 19 09 2023].
- [59] S. Ghosh, N. Das and M. Nasipuri, "Reshaping inputs for convolutional neural network: Some common and uncommon methods," *Pattern Recognition*, vol. 93, pp. 79-94, Sep 2019. doi: <https://doi.org/10.1016/j.patcog.2019.04.009>.
- [60] V. Thambawita, I. Strümke, S. A. Hicks, P. Halvorsen, S. Parasa and M. A. Riegler, "Impact of Image Resolution on Deep Learning Performance in Endoscopy Image Classification: An Experimental Study Using a Large Dataset of Endoscopic Images," *Diagnostics*, vol. 11, no. 12, p. 2183, Nov 2021. doi: [10.3390/diagnostics11122183](https://doi.org/10.3390/diagnostics11122183).
- [61] J. Filter, *split-folders 0.5.1*, (2022), Python Package Index, [Online]. Available: <https://pypi.org/project/split-folders/>.
- [62] M. Buda, A. Maki and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249-259, Oct 2018. doi: <https://doi.org/10.1016/j.neunet.2018.07.011>.
- [63] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, pp. 1-54, Mar 2019. doi: <https://doi.org/10.1186/s40537-019-0192-5>.
- [64] A. T. Dang, "Accuracy and Loss: Things to Know about The Top 1 and Top 5 Accuracy," 7 Jan 2021. [Online]. Available: <https://towardsdatascience.com/accuracy-and-loss-things-to-know-about-the-top-1-and-top-5-accuracy-1d6beb8f6df3>. [Accessed 19 09 2023].
- [65] J. Brownlee, "Difference Between a Batch and an Epoch in a Neural Network," 10 August 2022. [Online]. Available: <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/>. [Accessed 19 09 2023].

- [66] Google, "Descending into ML: Training and Loss," [Online]. Available: <https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss>. [Accessed 19 09 2023].
- [67] Y. Kang, "Multi-Class and Cross Entropy Loss," [Online]. Available: <https://sisyphus.gitbook.io/project/deep-learning-basics/basics/multi-class-and-cross-entropy-loss>. [Accessed 19 09 2023].
- [68] National Archeological Museum of Athens, "A versatile Mycenaean scribe," Feb 2021. [Online]. Available: https://www.namuseum.gr/en/monthly_artefact/a-versatile-mycenaean-scribe/. [Accessed 19 09 2023].
- [69] K. Papavassileiou, D. I. Kosmopoulos and G. Owens, "A Generative Model for the Mycenaean Linear B Script and Its Application in Infilling Text from Ancient Tablets," *Journal on Computing and Cultural Heritage*, vol. 16, no. 3, pp. 1-25, 2023. doi: 10.1145/3593431.
- [70] K. Papavassiliou, G. Owens and D. Kosmopoulos, "A Dataset of Mycenaean Linear B Sequences," in *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, Marseille, France, May 2020.
- [71] computervisioneng, "GitHub Profile: computervisioneng," [Online]. Available: <https://github.com/computervisioneng>. [Accessed 19 09 2023].
- [72] computervisioneng, "IMAGE CLASSIFICATION with Yolov8 custom dataset | Computer vision tutorial," [Online]. Available: <https://www.youtube.com/watch?v=ZeLg5rxLGLg>. [Accessed 19 09 2023].
- [73] G. Yang, J. Wang, Z. Nie, H. Yang and S. Yu, "A Lightweight YOLOv8 Tomato Detection Algorithm Combining Feature Enhancement and Attention," *Agronomy*, vol. 13, no. 7, p. 1824, Jul 2023.

