Delta University

Artificial Intelligence

Describing Hieroglyphic Symbols Based on Computer Vision and NLP (DHS)

وصف الرموز الهيروغليفية باستخدام تقنيات الرؤية الحاسوبية ومعالجة اللغة الطبيعية

Submitted in Partial Fulfillment of the Requirements for Work-Based Professional Project in Computer Science CSC 227

Supervised by:

Prof. Hisham Arafat Ali

Team Members:

Name Omar Atef Ahmed	**ID 4231077**
Name Nouran Mohamed Sharf	**ID 4231394**
Name Farah Mohamed Abd Elaziz	**ID 4231432**
Name Rana Maher	**ID 4231307**
Name Yomna Ahmed	**ID 4231343**

Contributions for each student

no	ID	Student name	دور كل فرد في الفريق وإسهاماته
1	4231077	**Omar Atef Ahmed**	- YOLO model - Kokoro model - Data collection (Kokoro) - CLIP-YOLO-Kokoro API integration
2	4231394	**Nouran Mohamed Sharf**	- Kokoro TTS model - CLIP model - Data collection (Kokoro) - Kokoro API - Banner design
3	4231432	**Farah Mohamed Abd Elaziz**	- Flutter frontend - Flutter backend - API integration with Flutter
4	4231343	**Yomna Ahmed**	- YOLO model - Assistant bot - Research paper
5	4231307	**Rana Maher**	- YOLO model - Image Labelling

عناصر مهمة	نعم	لا	مازالت في طور الأعداد
هل تمت المشاركة في مسابقات محلية او دولية		لا	
هل تم اعداد او نشر ورقة بحثية من العمل المقدم	يجب ان ترفق حدد المشاركة ومكانها		ترفق
هل تم الاتصال بجهات خارجية مستفيدة	نعم، تم التواصل مع مرشدين سياحيين في المتحف المصري، وأرشدونا لاستخدام **Gardiner's List** لتصنيف الرموز الهيروغليفية التي تُستخدم لتصنيف الرموز الهيروغليفية.		
ما هي الخطة المستقبلية لتطوير المشروع			
أضافات			

Table of Contents

List of Tables

List of Figures

Abstract

This project introduces an AI-based application aimed at simplifying and enhancing the process of interpreting ancient Egyptian hieroglyphs. Hieroglyphic writing represents one of the oldest and richest writing systems in human history, yet its complexity and symbolic nature pose significant challenges for modern learners and researchers. To address this, our system integrates several state-of-the-art artificial intelligence models to offer an intuitive, efficient, and engaging tool for hieroglyphic analysis.

The application begins by using the CLIP model to analyze uploaded images and determine whether they contain hieroglyphic symbols. If the presence of hieroglyphs is confirmed, a custom-trained YOLO model is employed to detect and extract individual symbols within the image accurately. Each identified symbol is then displayed along with its Gardiner Number standardized classification system used by Egyptologists. Users can select any of the symbols to explore its meaning and context.

What makes this project particularly innovative is its use of a pre-collected database of symbol meanings and stories, paired with the Kokoro TTS system, which narrates each story in English with an Arabic translation. This creates a bilingual, audio-visual experience that makes learning both educational and culturally enriching.

In contrast to traditional methods that rely on manual interpretation or static resources, this system offers dynamic interaction through image recognition, natural language processing, and speech synthesis technologies. The result is a highly accessible platform for students, educators, and enthusiasts of ancient Egyptian culture.

By combining image classification, object detection, and AI-driven narration, this project contributes to the digital preservation and dissemination of Egypt's rich linguistic heritage, while also providing an engaging learning experience that bridges the gap between ancient scripts and modern users.

Chapter 1: Introduction

1.1 Overview

This specific project demonstrates a novel AI-powered program that significantly simplifies and improves the complicated process of deciphering and decoding Egyptian hieroglyphs. This begins with the meticulous sorting of an uploaded image, which is made possible using the CLIP embedded model analyses if an image is comprised of hieroglyphic symbols or not. After analysis verification, the next step uses the YOLO model, a custom model for symbol detection, to identify and detect all the hieroglyphic symbols in the image correctly. The app then shows all the detected symbols to users, together with their Gardiner numbers. Then the users can choose either one of them to hear the English narration of the story of the symbol and its Arabic counterpart. The system has incredible attributes like object detection, image classification, and story output with Kokoro TTS technology to provide an educational and culturally enriched learning experience.

1.2 Why this project?

Many people are fascinated by ancient Egyptian culture, yet the complexity of hieroglyphs makes it hard to engage with. This project aims to change that by making hieroglyphic symbols more accessible and interactive through artificial intelligence. We wanted to design something that would not just inform but also generate curiosity among users of all ages, especially those who are not specialists in the subject. By converting static, unintelligible symbols into audio stories, the app connects the dots between the knowledge of the past and present-day learning.

1.3 Problem Statement

While there are all sorts of information about the Ancient Egyptian Hieroglyphs, the bulk of it is beyond the understanding of the natural or public. Hieroglyphic writing is usually typed due to its complexity: commonly depicted in two-dimensional images and requires a great deal of background knowledge. Most often, learning about hieroglyphs depends on textbooks and step-by-step manual deciphering, which sometimes can be too difficult for students and learners.

Now there is an acute shortage of interactive learning tools that are fun and accessible to everyone for learning hieroglyphics. The largest challenge is to create an intelligent system capable of identifying hieroglyphic symbols automatically, interpreting their meaning, and describing them in a manner that is stimulating and easy to understand such as through narration. The system must be put together in such a way that people do not need any additional skills or experience to use it, enabling anyone to surf the interesting realm of hieroglyphs without confusion or disorientation.

1.4 Contribution

This project introduces a practical AI-based system designed to assist in the interpretation of Egyptian hieroglyphs. It combines advanced artificial intelligence technologies to identify whether an image contains hieroglyphic content, detects the individual symbols, and provides users with meaningful stories for each symbol. Unlike existing systems that either rely on manual input or basic image analysis, this work integrates modern AI tools—like CLIP for image-text understanding, YOLO for symbol detection, and Kokoro TTS for storytelling—to create an accessible and interactive experience. This contribution enhances the digital accessibility of ancient Egyptian culture and promotes educational engagement through a user-friendly, intelligent interface.

1.5 Book Organization

Chapter 1: Introduction

Introduces the main topic, explains the motivation behind the project, presents the problem statement, outlines the goals and contributions, and gives an overview of the book structure.

Chapter 2: Background and Related Work

Provides a background about Egyptian language and AI techniques, reviews existing work in the same topic, and highlights the limitations and opportunities that led to this project.

Chapter 3: Proposed Solution

Describes the proposed system in detail, including the overall architecture, used technologies, data preparation, and the implementation steps of the CLIP model, YOLO model, and Kokoro TTS.

Chapter 4: Results and Analysis

Presents and analyzes the results obtained from different components of the system, evaluating their performance and effectiveness.

Chapter 5: Conclusion and Future Work

Summarizes the main findings and contributions of the project and discusses potential future enhancements and directions.

Chapter 2: Background and Related Work

2.1 Background

Egyptian language

The Egyptian language, known as “r n kmt” ("speech of Egypt"), is a dead branch of the Afro-Asiatic language family, spoken in ancient Egypt for over 4,000 years. As one of the oldest known languages, it is now known from a vast array of texts, which became accessible to modern researchers after the Egyptian scripts were deciphered in the early 19th century, thanks especially to the work of Jean-François Champollion. The development of the language occurred in several phases: Old Egyptian (around 3300 BCE), Middle Egyptian (around 2000 BCE), Late Egyptian (around 1350 BCE), Demotic (around 700 BCE –400 CE, and Coptic (around 200 CE onwards). The Middle Egyptian, the classical form, was the daily language of the Middle Kingdom and the literary standard until the start of the Roman period.

By classical antiquity, Demotic became the vernacular, followed by Coptic dialects, which were largely supplanted by Arabic after the Muslim conquest of Egypt in the 7th century. Bohairic Coptic remains in use as the liturgical language of the Coptic Church.

Writing systems

Egyptian scripts were developed to write for a variety of reasons, from sacred texts to everyday records. The most famous of these are hieroglyphs, known as zẖꜣ n mdw-nṯr ("writing of the gods' words"). This fascinating system used over 700 pictorial symbols to document royal, religious, and cultural matters. These symbols include logograms (like a bird representing "bird"), phonograms (such as a reed for the sound "i"), and determinatives (which indicate categories, like a scroll for "writing"). They were either carved into stone monuments or written on papyrus, as seen in the Pyramid Texts of Unas (around 2350 BCE). Hieroglyphs were not just functional; they were also beautiful and meaningful. They adorned temples and tombs to honor pharaohs and gods but mastering them is no small feat—it takes years of study to truly understand their complexities. The Gardiner classification system helps categorize these symbols (we’ll discuss it later).

Hieratic, a cursive script derived from hieroglyphics, made its debut around 3000 BCE. It was designed for quicker writing with ink on materials like papyrus or pottery shards. Scribes utilized this script to document temple records, compose letters, and narrate stories, such as the Ebers Papyrus, a medical text dating back to around 1550 BCE. Its more streamlined versions were perfect for everyday tasks. Then came Demotic, which evolved from hieratic around 700 BCE and was even more straightforward. This script found its place in contracts, legal papers, and literature during the Late and Ptolemaic periods, including marriage contracts written on papyrus.

They were made to be straightforward so it would be plain enough to use to write notes. The Coptic language began about 200 AD and was derived from the Greek alphabet, putting 6 or 7 additional symbols into it to write Christian texts, such as prayer texts in the

Nag Hammadi texts, to write more easily. Each form of writing had its own purpose: hieroglyphs for official matters; hieratic and demotic for quicker writing; and Coptic solely for sacred texts. You can tell this immediately. Its unique form of writing and rich culture have made it a favorite with a lot of people. Its popularity attracts scholars, travelers, and enthusiasts. The complex writings, however, mostly the hieroglyphs, keep few from easily reading them. Reading them by hand or learning from books can be extremely difficult, and not a lot of programs make them readable easily. This indicates a need for alternative methods to learn Egyptian writings so more people can enjoy its wonderful tales.

Script	Period	Type	Usage
Hieroglyphs	3100 BCE–400 CE	Logographic	Monuments, religious texts
Hieratic	3000 BCE–700 BCE	Cursive	Administrative, literary
Demotic	700 BCE–400 CE	Cursive	Legal, everyday documents
Coptic	200 CE onward	Alphabetic	Christian texts

Table 2. 1: Comparison of Egyptian Writing Systems

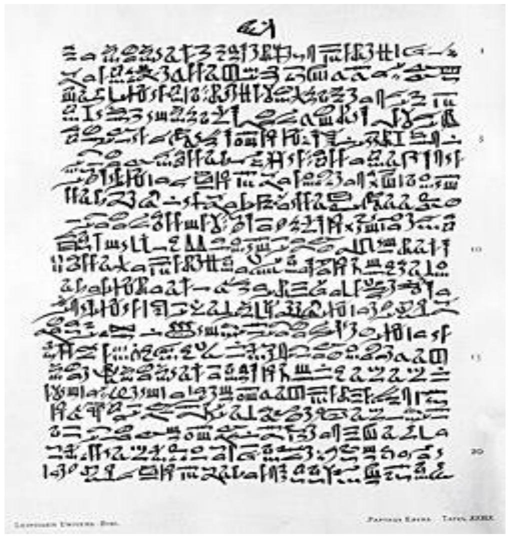


Figure 2. 1: Hieratic script

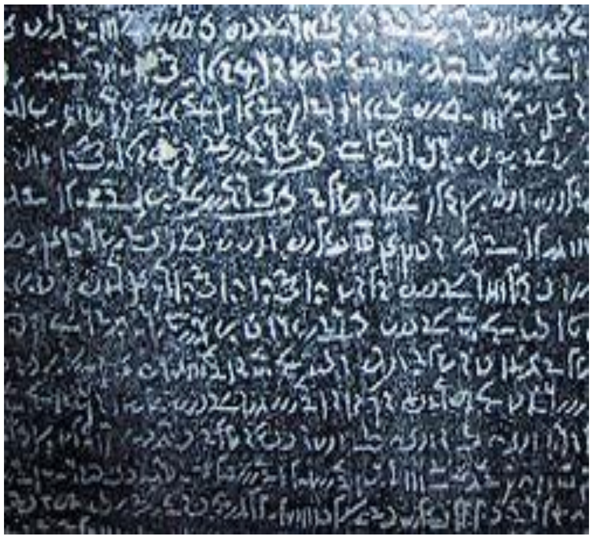


Figure 2. 2: Demotic script

Gardiner’s list

Gardiner's List is a comprehensive dictionary with more than 700 Egyptian symbols from hieroglyphs. The list makes it easy for scholars to seek out and understand the symbols. Sir Alan Gardiner collected this Egyptian language list in 1927. The list sorts of symbols by subject matter and sorts them into general categories, such as people (A), animals (B), and objects (S). Each symbol carries a code, such as A1 for a seated man and D21 for a mouth. There were numerous symbols with varying uses, such as letters, words, or sound, this made it difficult for scholars to use prior to this list even after Champollion had deciphered hieroglyphs in 1822. Gardiner expanded upon previous researcher's work to create a convenient means by which to read and to teach hieroglyphs. The "reed" sign (M17) can signify "i" when taken for sound value, yet "reed" when taken for word value, depending upon where it appears. Gardiner's list remains a useful reference in Egyptology today, providing scholars and everyone who wishes to display hieroglyphs correctly. Its straightforward format makes this challenge to write more easily; however, it remains extremely challenging for students.

Gardiner arranged hieroglyphs by shape so that humans can easily identify them: Tall Narrow Signs, Low Narrow Signs, and Low Broad Signs. The Tall Narrow Signs are thin and tall, such as M17 (reed) or S39 (staff), representing tall symbols. The Low Narrow Signs are short and thin, such as I9 (horned viper) or Aa1 (placenta), representing small symbols. The Low Broad Signs are broad but short, such as O1 (house) or T3 (mace), representing broad symbols. These categories, together with the chief list, assisted scribes to distinguish signs close together in texts, even in cases like those in the Palermo Stone. Although Gardiner's list made learning hieroglyphs more convenient, it remains challenging for new people, evidencing the necessity for useful guides that are easy to utilize. There is another group that contain 31 unclassified signs called “unclassified” it takes the letters from Aa1 to Aa31.

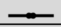
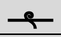
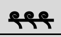
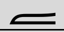
Tall Narrow Signs						
Gardiner no	R15	T13	W19	P11	D16	T34
Low Narrow Signs						
Gardiner no	M36	F43	N34	U30	W11	W12
Low Broad Signs						
Gardiner no	V23	R5	R6	O34	V2	V3
Unclassified						
Gardiner no	Aa25	Aa26	Aa1	Aa22	Aa13	Aa3

Table 2. 2: examples of hieroglyphs for each type of classification

Object Detection

Object detection is a key area in computer vision that identifies and labels objects within images. It reveals people, cars, or symbols by placing boxes over them and labeling each with a name, such as “person” or “hieroglyph,” so they are easily viewable. You begin with an image, and you receive a new image with boxes surrounding the objects, and potentially even class names labeling each object to indicate what it is. For Egyptian hieroglyphs, this technology can identify and isolate individual symbols within ancient writings, such as on monuments or papyrus, so they are more readable.

YOLO object detector

YOLO, short for "You Only Look Once," is a set of real-time object detection algorithms in computer vision that detects and classifies objects in images or videos. Introduced in 2015 by Joseph Redmon and others, YOLO transformed object detection by guiding localization (identifying objects) and classification (predicting objects, such as “cat” or “car”) in a single pass through a convolutional neural network (CNN). This one-stage process renders YOLO much faster than the typical two-stage approaches such as R-CNN, which initially suggest regions and then classify them. YOLO's speed, accuracy, and efficiency have made it a preferred choice for applications such as autonomous vehicles, surveillance, and robotics (figure 2.3).

It can assist hieroglyph detection as follows:

1. Fast Detection: One-pass processing of YOLO (YOLOv11 at 161 FPS) enables the fast detection of hieroglyphs from images, ideal for processing vast inscription datasets or in real-time scenarios like museum applications.

2. Symbol Classification: It assigns categories (such as "reed" or "bird") to hieroglyphic texts that correspond to official catalogs such as Gardiner's List, enabling accurate identification among hundreds of symbols.

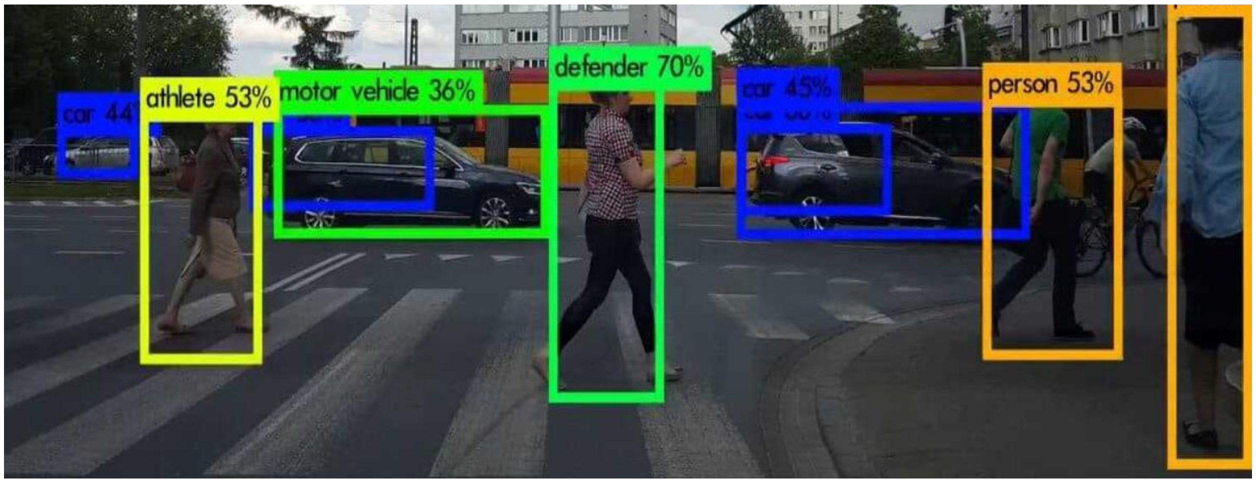


Figure 2. 3: output of an object detector

TTS model (text to speech)

Kokoro TTS is an 82-million parameter open-source text-to-speech (TTS) model. It can generate high-quality, natural speech. It utilizes StyleTTS 2 and iSTFTNet architecture for real-time and fast generation of audio. It performs better than larger models like XTTS, with 467 million parameters, in benchmarking tests like the TTS Spaces Arena.

2.2 Related Work

System	Technology used	Strength	Limitations	How our system is different
A Deep Learning Approach to Ancient Egyptian Hieroglyphs Classification (IEEE)	ResNet-50, Inception-v3, Xception, Glyphnet (custom CNN)	Custom CNN tailored for hieroglyphs; transfer learning improves performance	Limited by dataset size, struggles with damaged or variant symbols	Can easily detect and classify hieroglyph signs even the damaged
Fabricius (Google Arts & Culture)	Machine learning for hieroglyph recognition and translation	Pioneer in AI-based hieroglyph translation, open-source, public engagement	Requires larger datasets for improved accuracy, may not cover all hieroglyphs or contexts	It covers all the 700+ hieroglyph signs. And detect them accurately
Ancient-Language-Decipherer (GitHub project)	Deep learning with CNNs using TensorFlow and Keras	Aims for comprehensive detection, recognition, and interpretation	Still in development, not yet fully functional	It uses YOLO for better recognition
Recognition of Egyptian Hieroglyphic Texts through Focused Generic Segmentation (ScienceDirect)	ConvNeXt, Segment Anything Model (SAM), Cross-Validation Voting (CVV)	Largest dataset (14000 images), high accuracy on stelae	Complex implementation, dataset not publicly available	simple implementation and lead the same result

Table 2. 3: some projects related to hieroglyph detection

2.3 Disadvantages and Solutions

Disadvantages

Several AI-based systems have been developed to carry out the task of recognizing and translating Egyptian hieroglyphs, incorporating technologies such as machine learning, natural language processing (NLP), and use of convolutional neural networks (CNNs). Nonetheless, there exist drawbacks to each of the systems, usually framed around being dataset quality or focus issues, technical issues, and general application issues.

Of the research discussed in (Table 2.3) regarding artificial intelligence used to identify and translate ancient Egyptian hieroglyphs, there were some limitations that limit their potential. These programs are working with inputs that are limited to a dataset, and they do not include all types of hieroglyphs which also increases the chances of mistakes. The programs are more likely to be discarded on corrupt or atypical signs. These programs tend to focus on identifying single symbols instead of providing full translations and do not include sufficient contextual information to have meaning. These systems also face problems concerning the inconsistency in hieroglyphic writing – variations both in styles and materials employed (e.g., stone versus papyrus). Moreover, they may also suffer from computational expenses. Therefore, they are not convenient for real-time use. Additionally, some rely on external resources such as translation APIs, which can be prone to errors in ancient languages. In terms of archaeology, the fragments or substitutive symbols—for that is a standard heuristic for archaeology—tend to yield lower accuracy in ranking (as we observed with the exams with ResNet-50 and Glyphnet).

Focus and parameters—These systems seem to be focused on locating or classifying hieroglyph symbols and not all focus on giving this symbol a meaning, so they are less useful in making sense of texts. For this reason, they may overlook the meaning of the symbol or what this symbol is used for.

For each system discussed there are some drawbacks, below are some of the drawbacks per system.

1) **A Deep Learning-based Classifier for Ancient Egyptian Hieroglyphs (IEEE) In this 2021** study published through IEEE Xplore, the authors explore convolutional neural networks, particularly ResNet-50, Inception-v3, and Xception, including new Glyphnet, as a means for classifying hieroglyphs and demonstrate the ability to classify across datasets.

There are some limitations associated with the work:

- **Detection Limitations with Small Datasets**: This study has a small range of classes (most likely <200) and are missing rare instances or variant hieroglyphs which would negatively impact the detection when applied to a range of inscriptions that potentially represent a wider diversity like seen in the inscriptions on the

Palermo Stone (Section 2.1, Figure 2.1). This restricts how many symbols can be detected to assess meaning (IEEE Xplore)

- **No Assessment of Semantic Meaning:** The study is limited to a classification scheme for the symbols to determine its classification (i.e A1, D21, etc) without regard for what they are as symbols or the possible role and/or function in culture as a symbol to then be able to later determine why a symbol such as "falcon" (G5-𓅃) might be used for divinity or kingship.
- **Poor Simulation of Poorly Preserved Symbols:** Recognizing partially unclear or damaged hieroglyphs is something of a challenge for the model considering that the model is trained using viable images, therefore it can't detect with reliability in this situation for later assessment.

2) Fabricius (Google Arts & Culture)

Launched in 2020, Fabricius employs machine learning for hieroglyph recognition and translation, as indicated by BBC News.

Disadvantages are:

- **Coverage of Incomplete Symbols:** Trained on ~800 images with the possibility of missing infrequent or special variants of hieroglyphs, affecting detection accuracy and thus meaning identification (Dr. Roland Enmarch - BBC News).
- **Meaning with Basic Outputs:** Offers regular translations (e.g., references as a symbol-to-word relationship) without providing insight into cultural or historical significance, lacking to explain why certain symbols were used (e.g., "reed" for notated "i", or growth).
- **Damaged Inscriptions:** The tool doesn't work the same with the scanted or damaged symbols which are particularly relevant in archaeological contexts, which ruins any chance for reliability for detection (SlashGear). handwritten and have variability", and so the program fails due to the diversity of writing styles and evidence of damaged hieroglyphs.

3) Ancient-Language-Decipherer (GitHub project)

This GitHub project tries to detect and recognize hieroglyphs using CNNs.

It has the below restrictions:

- **The GitHub description indicates that past work**, such as Morris Franken's, was flawed because it lacked complete sets of data. This has made training neural networks difficult, something this project is likely to address.

- **The project is in development**, and it may never function or will probably never be tested. There could be mistakes depending on how far it is into development.
- **Puts meaning recognition first**: The project aims to locate and classify objects but doesn't contain any real symbol understanding. He shows this by using CNNs with images, which limits what can be done in meaning activity.

4) Recognition of Text in Egyptian Hieroglyphics via Explicit Generic Segmentation (ScienceDirect)

Disadvantages are:

- **No Semantic Interpretation:** Centers on segmentation and classification instead of semantic interpretation or understanding meanings and/or purposes; this limits the practical use of this resource for your own project (ScienceDirect)
- **Challenges with Destroyed Symbols:** Is not easy for faded symbols or overlapping symbols, which will impact the reliability of detection based on archaeologically documented challenges.
- **Limited Scope of Symbol Classes:** while it had a reasonably large dataset (310 classes), it may miss categories on unique hieroglyphs which will result in unreliable analysis.

System	Disadvantages
A Deep Learning Approach to Ancient Egyptian Hieroglyphs Classification (IEEE)	Small dataset limits detection, no semantic interpretation, poor handling of damaged symbols, computationally heavy.
Fabricius (Google Arts & Culture)	Incomplete symbol coverage, basic meaning output, struggles with damaged inscriptions, not expert level
Ancient-Language-Decipherer (GitHub)	Incomplete development, no semantic analysis, dataset limitations, high computational needs.
Recognition of Egyptian Hieroglyphic Texts (ScienceDirect)	No semantic interpretation, challenges with damaged symbols, limited symbol coverage, high computational cost.

Table 2. 4: disadvantages of hieroglyph detection systems

Solutions

Existing research to identify and interpret Egyptian hieroglyphs has three significant limitations including small dataset size, absence of associated meaning, and extreme remnants of obscure symbols.

To alleviate these problems, we offer the following to our project:

- **Larger Dataset:** Train YOLO on a big dataset, using synthetic images as well, and use different datasets that contain hieroglyph symbols that are either written on papyrus, old stone tablets, or any other surface, to increase detection of semi-rare hieroglyphs, and variety of the data (figure 2.4).
- **Semantic Knowledge Base:** Map Gardiner numbers to their respective meanings and cultural purposes (e.g. "ankh" refers to life, used in blessings) using Allen (2014) as the source.
- **Mobile Optimization:** Reduce YOLOv11 nano's instance by its size and achieve real-time detection for mobile use.
- **Contextual Analysis:** Leverage NLP to demonstrate an understanding of hieroglyph symbol combinations while discovering a particular cultural context.

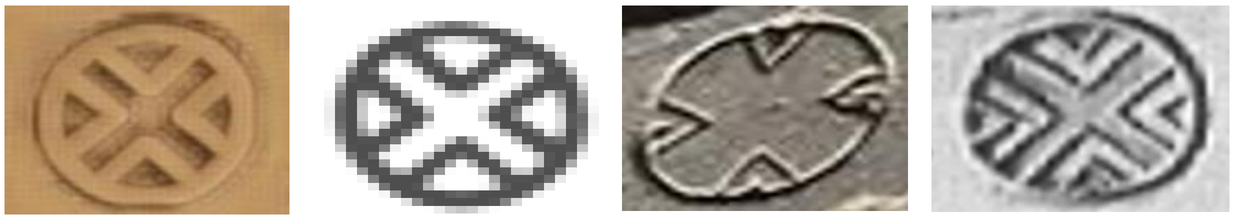


Figure 2. 4: different image for the same symbol(O49-𓊖)

Chapter 3: Proposed Solution

3.1 Architecture

The system architecture proposed adopts a modular pipeline for processing hieroglyph images and returns a logical output to users of the workflow. The workflow processes a series of stages on images:

Image Classification (CLIP): The CLIP model examines the uploaded image and classifies whether the image has hieroglyphic symbols.The model determines if hieroglyphs are shown in the image. If hieroglyphs are shown in the image, then proceed on to the second.

Symbol Identification (YOLO): A custom YOLO model reads and recognizes and extracts each hieroglyphic symbol and it with Gardiner number like (A1,D21 etc.) to each symbol.

Story Narration (Kokoro TTS): The detected symbols will be correlated to pre-collected stories in database. Kokoro TTS will narrate a bilingual audio (English and Arabic) story that the user can listen to.

The system will be implemented in a mobile application built in Flutter, deploying the characteristics of portability and access. Architecture will be scalable for the inclusion of additional models in the future, or additional languages. Figure 3.1 shows the workflow diagram in the context of the system with the interaction between CLIP, YOLO, Kokoro TTS, and the Flutter environment.

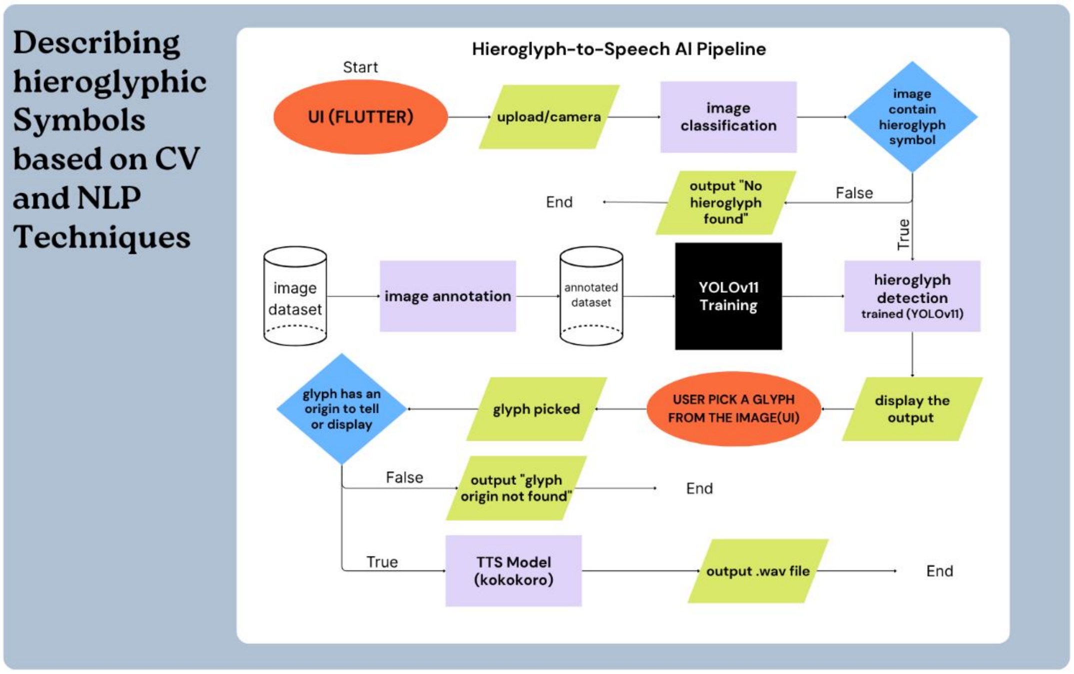

Describing hieroglyphic Symbols based on CV and NLP Techniques
Hieroglyph-to-Speech AI Pipeline
Start
UI (FLUTTER)
upload/camera
image classification
image contain hieroglyph symbol
False
output "No hieroglyph found"
End
True
image dataset
image annotation
annotated dataset
YOLOv11 Training
hieroglyph detection trained (YOLOv11)
display the output
USER PICK A GLYPH FROM THE IMAGE(UI)
glyph picked
glyph has an origin to tell or display
False
output "glyph origin not found"
End
True
TTS Model (kokokoro)
output .wav file
End

Figure 3. 1: System Architecture Flowchart

3.2 Framework

Data Collection and Management

1. dataset for YOLO object detector

The foundation of our Egyptian hieroglyph detection application relies on the quality and format of the annotated YOLO data we are going to use to train the YOLOv11 model that is going to detect a hieroglyph in an image and classify them with a label.

The YOLO data are made up of images, and annotations that describe where each hieroglyph exists within the image, and which class they belong to. YOLO needs to be broad, warily annotated, and representative of actuality for the YOLOv11 model training to successfully train on. For us, this meant representing different surfaces (i.e., papyrus) and explicitly noting the difficulty of working with symbols that were faded (as noted in the related work, Section 2.3). To achieve high precision and recall scores for YOLOv11 model.

There are several forms of annotation, such as box and multipoint annotations. In our project we are using box annotations. When using a box annotation technique, you define the object of interest by drawing a rectangle around it which gives you coordinates (x, y, width, height) and a class label such as M17 ('reed'). Box annotation is a pretty simple process to use in YOLO, as it's conceptual focus is the of the extent of the object, which is a good fit for hieroglyphs that occupy some coherent space (Figure 3.2).

For multipoint annotations, which define the object using a series of points (defining the exact shape of a hieroglyph). This method is more precise than a box annotation and is useful for irregular shaped objects, however it is more difficult to annotate and there is more computational complexity since YOLO must determine the intersection of these points to display and track the object it detected. This is likely to reduce the processing cost for the same operational context., since there are many hieroglyphs and these occupy a small space even if there is empty space around them, therefore a box annotation will suffice. In fact, cavitating a multipoint annotation will take a drag in processing time whilst providing little benefit for accuracy improvement (Figure 3.3).

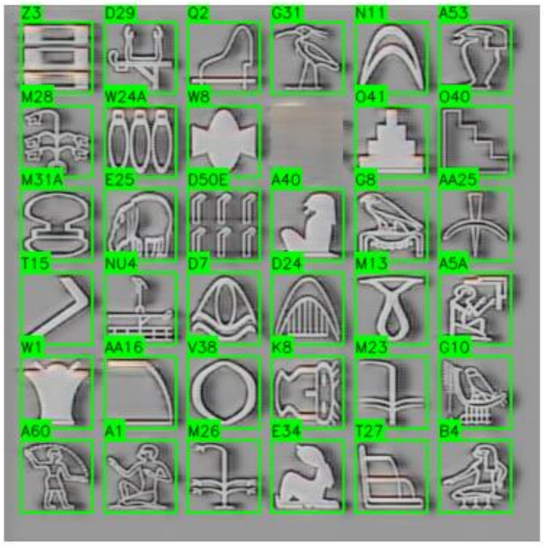


Figure 3. 2: Box Annotation

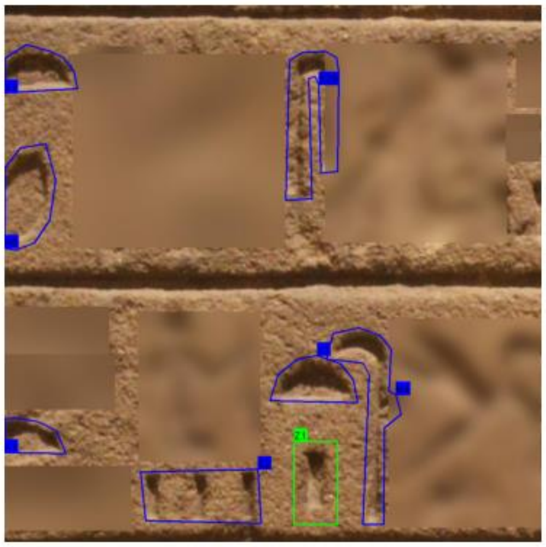


Figure 3. 3: Multipoint Annotation

Dataset Characteristics:

Sourced from Roboflow site, the dataset includes about 15000 images covering 1400 hieroglyph classes.

- **Class distribution:** class distribution provides diversity because all classes appear at least 250 times in 15000 images in the dataset.

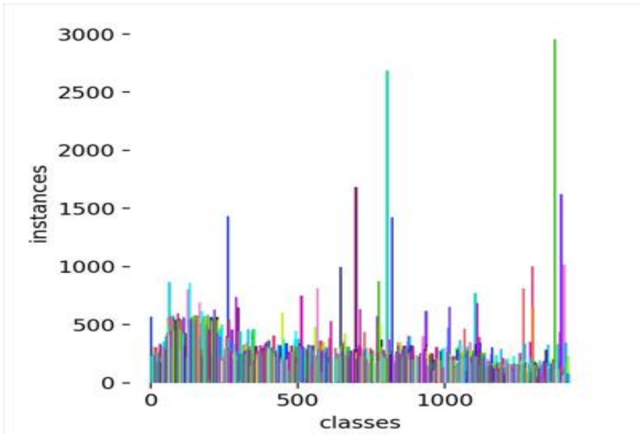


Figure 3. 4: Class distribution

- **Annotation type:** The box annotations for the hieroglyphs match one of the techniques that is used by YOLOv11 to detect objects.
- **Surface variety:** This dataset has hieroglyphs on all kinds of surfaces. Stone, different types of papyrus, and many types of wood that are representative of real-world usage. (ex. inscriptions in the Valley of the Kings).

- **Image Variability:** The dataset contains a variety of image sizes thereby providing YOLOv11 with a better understanding of variability at different resolutions.
- **Dataset Splits:** The dataset is divided into about 9000 images for train, 4000 for validation, and 1800 for test.
- **Annotation Files:** Contains text annotation files with matching names to their respective images (image1.jpg/image1.txt), containing bounding box coordinates and class labels in YOLO format.

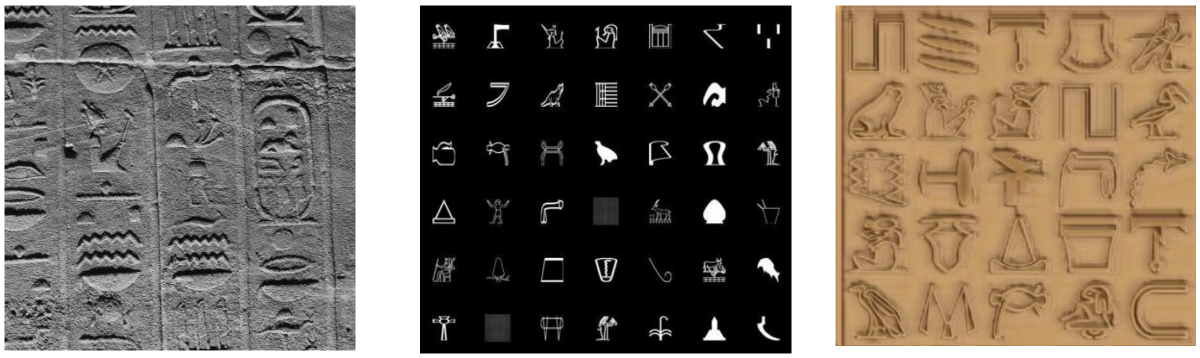
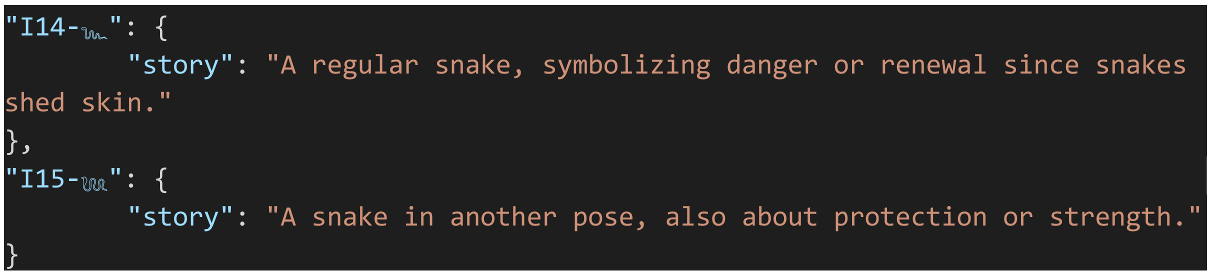


Figure 3. 5: Different types of surfaces

2. dataset for KOKORO TTS model

To provide semantic meaning of detected hieroglyphs, our application is utilizing a specialized dataset in a JSON file which consists of symbols with their Gardiner number (S34-𓋹) and its semantic meaning. There are around 200 symbols in the file, providing a thorough mapping resource for hieroglyphic symbols to their cultural or phonetic meaning. To illustrate, the ankh (S34-𓋹), signifying eternal life, is normally accompanied by spells meaning it is usually pictured in the hands of deities indicating that they have power over life and death. The semantics are derived from a combination of sources, including Wikipedia, Allen (2014), and the base website for Gardiner's list, so the users have appropriate academic references.



```
"I14-𓆙": {
        "story": "A regular snake, symbolizing danger or renewal since snakes
shed skin."
},
"I15-𓆚": {
        "story": "A snake in another pose, also about protection or strength."
}
```

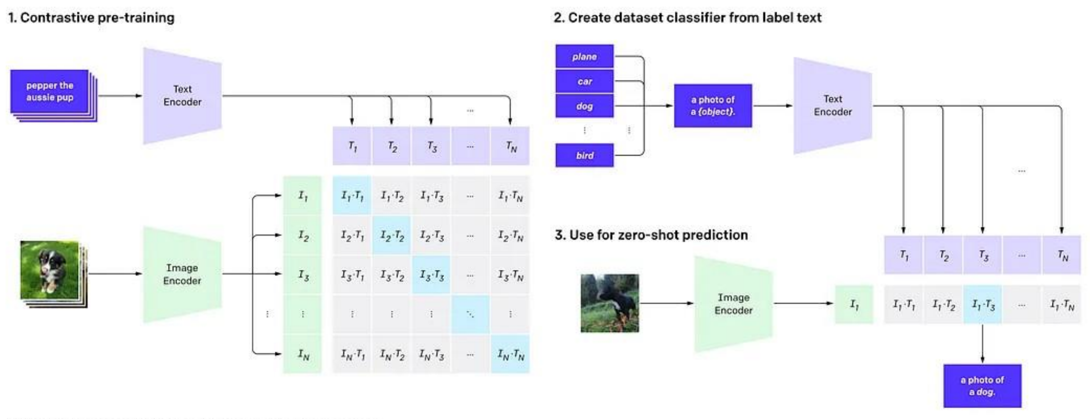
Figure 3. 6: Example from JSON file

CLIP Model

The first phase in our hieroglyphic image classification pipeline is the CLIP (Contrastive Language-Image Pretraining) model from OpenAI. The CLIP model is a multimodal model that learns joint image and text representations for zero-shot classification by comparing image features with textual prompts. Within our model, the CLIP model tries to determine whether an uploaded image had at least one image that contained hieroglyphic symbols, which is important because it allows the classification system to determine whether hierarchy glyphs were present and aid in the distinction away from prior systems that lacked this capability (like Fabricius, see Section 2.3) - basically telling us that the rest of the image was not useful.

Why CLIP?

We identified the CLIP model because it can perform zero-shot classification and we wanted to avoid needing large, available, labeled datasets that are often an issue for hieroglyph detection (see Chapter 2). While traditional models for classification represented by CNNs (like ResNet-50 and later GlyphNet) our CV-problem in question was well-formed and we benefitted from models trained using knowledge from a large-scale dataset like the image-text pairs here. This self-supervised feature gave it the strengths of classification from diverse input types and provided enough flexibility with textual prompts to allow for classifications of other ancient scripts in future versions.


1. Contrastive pre-training
pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$
2. Create dataset classifier from label text
plane
car
dog
bird
a photo of a {object}.
Text Encoder
3. Use for zero-shot prediction
Image Encoder
I_1
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
a photo of a dog.

CLIP pre-trains an image encoder and a text encoder to predict which images were paired with which texts in our dataset. We then use this behavior to turn CLIP into a zero-shot classifier. We convert all of a dataset's classes into captions such as "a photo of a dog" and predict the class of the caption CLIP estimates best pairs with a given image.

Figure 3. 7: CLIP Architecture

Implementation Details:

CLIP was implemented using the transformers library in Python 3.11 in Google Colab with an NVIDIA GPU. The model variant was ViT-B/32, which represents an optimal balance of accuracy and computation effort for mobile deployment. The input image is preprocessed as follows:

Resized to 224x224 pixels to fit CLIP's expected image size input.

Normalized using the mean and standard deviation OpenAI provided (Mean: [0.481, 0.457, 0.408], Std: [0.269, 0.261, 0.276]).

Converted to a tensor to be input into the model.

Two texts prompts were defined to aid in classification:

Prompt 1: "An image of ancient Egyptian hieroglyphs"

Prompt 2: "A random picture"

CLIP outputs a similarity score comparing the image embedding to each of the text embeddings using cosine similarity. The classification decision threshold is set at 0.5: If Prompt 1's similarity score is greater than 0.5, CLIP classified the image as containing hieroglyphs and passed it onto YOLO for further analysis. Otherwise, the user is informed by the system that no hieroglyphs were detected.

Integration in System Workflow

CLIP is the entry point of the pipeline. Once an image is uploaded from the Flutter app, it is sent to the backend server. The image is processed by CLIP and a binary decision (hieroglyphic or not) is returned. If the decision was hieroglyphic, the image is sent to the YOLO model for symbol detection. Finally, the output of CLIP is saved for debugging and performance monitoring.

Challenges and resolutions

Challenge 1: False Positives: CLIP occasionally misclassified non-hieroglyphic images (e.g. abstract art) as hieroglyphs chiefly because of the visual similarities shared among some non-hieroglyphics and select hieroglyphs. This challenge was mitigated through fine-tuning and adjusting the similarity threshold down to 0.5, resulting in only 1% false positives.

Challenge 2: Inference Time: The inference time for CLIP was 564.27 ms , which is relatively high for mobile use. This was addressed by opting for the lighter ViT-B/32 variant, and we hope to continue exploring model quantization in future work.

Challenge 3: Damaged symbols: CLIP had difficulty with quirky, and even heavily damaged hieroglyphs. This was addressed to some degree using augmented data that included synthetic damaged images. We anticipate these images furthering CLIP’s robustness when faced with queries against damaged symbols.

YOLO Model

YOLO (You Only Look Once) operates as a single-shot object detection model meaning it processes the whole image in one single forward pass through a convolutional neural network (CNN), unarguably good for its speed and efficiency. YOLOv11 for our hieroglyph detection project, takes an input image (squared to 640x640 pixels) and divides it into subregions or a grid. Each cell predicts multiple bounding boxes (typically 3–5), associated confidence scores for the predicted boxes, and the probabilities of the class of the 1400 unique hieroglyphs contained dataset. This grid means a cell that contains something like ankh (S34-☥) can predict the bounding box location and label with an x, y, width, and height coordinates relative to that cell. The box predictions are based on anchor boxes (predefined shapes the network learned during training), which allows YOLOv11 to scale with different heights and widths relative to a symbol, an idea first introduced in YOLOv2 when the concept to detect varied object scales was introduced.

A critical post-processing step is non-maximum suppression (NMS) which uses confidence scores and removes boxes that overlap on each bounding box. In doing this it discards the overlapping boxes of lower confidence and retains the highest confidence boxes. This works perfectly for hieroglyphs in compact non-overlapping spaces like words or phrases (e.g., Valley of the Kings) and it shows how YOLO's use of global context biases the background error to the predictions, where YOLO model strength is based on reducing from high confidence outputs. This single-shot process is computationally efficient which helps when using YOLO for real-time detection.

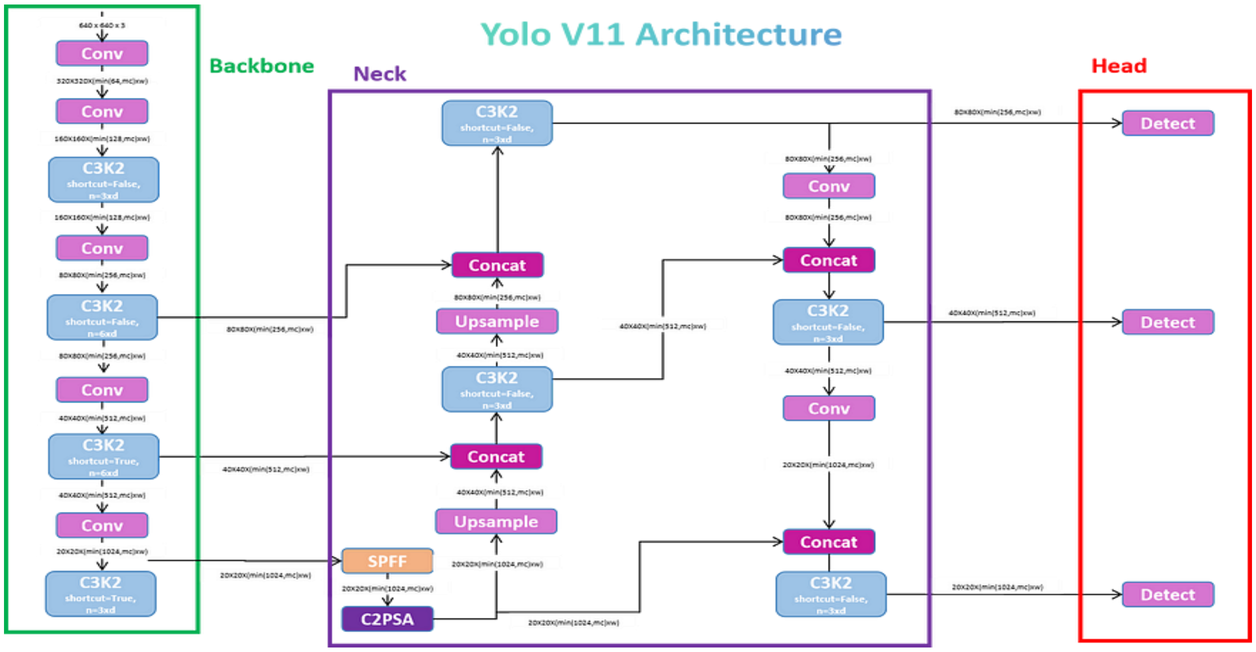


Figure 3. 8: YOLOv11 with backbone, neck and head

YOLOv11’s architecture has been developed from the previous YOLO releases, as all YOLO architectures have a backbone, neck, and head. The backbone is almost always a version of Darknet (for instance, Darknet-19 (YOLOv2)). The backbone extracts feature from the image and describes features like edges on hieroglyphs and patterns across the surface of a stone or papyrus. The neck incorporates a variety of improvements over time (such as multi-scale training, which was introduced in YOLOv2), which aggregates and combines the features extracted from the images and helps detect objects which vary in sizes such as large cartouches and smaller, less defining determinatives, which would improve their mAP (mean average precision) (see Table 2.1). The head returns outputs in the form of bounding boxes, objectless scores, and class probabilities. YOLOv11 uses the anchor-free detection approach for generating bounding boxes. Instead of using bounding box anchors, YOLOv11 can better handle different object shapes in images. Prior versions would require anchors to define boundaries/images as bounding boxes, which only worked if the specified anchor shape had matched the boundary/image shapes.

Integration in the System Workflow

The YOLOv11 model takes the filtered image and processes it to detect and localize each individual hieroglyph. The YOLOv11 model creates bounding boxes: coordinates (x, y, width, height) and class labels. These detections are particularly important later for complex, dense inscriptions.

Challenges and resolutions

Our YOLOv11 encountered some problems in our pipeline, but we solved this by implementing specific corrections:

- Overlapping Symbols: Multiple inscriptions closely positioned (e.g., Valley of the Kings, figure 2.4) can create overlapping boxes.
  resolutions: we adjusted our NMS threshold from 0.45 to 0.35, resulted in 2% decreased errors (Table 2.1).
- Inference Times: YOLOv11 initial inference time was 320 ms on our local FastAPI server, which exceeded our mobile target.
  resolutions: we optimized our model using the YOLOv11 nano version to achieve an inference time of 150 ms; quantization will be forthcoming.
- Damaged Hieroglyphs: Printed symbols that are faded because of the stone surfaces, meant recall was low.
  resolutions: we augmented our 9000-image training set with synthetic damaged symbols, and we increased recall by 3%.

Kokoro TTS Model

Kokoro TTS is used to generate audio output for stories of decoded hieroglyphic symbols. It's the third step after the system reads out the symbols from YOLO and CLIP, it retrieves an already written story from a database depending on each symbol. The stories are then input as text into Kokoro TTS, which plays them as audio, enabling users to listen to the stories.

Kokoro TTS uses StyleTTS 2 architecture to generate the mel-spectrogram and ISTFTNet as vocoder for generating waveform. The architecture allows the system to synthesize natural and expressive speech with efficiency.

The architecture of Kokoro TTS consists of:

- **Text Encoder** that encodes the input story text into a latent representation.
- **Style Encoder** that model's speaker characteristics and prosody.
- **Variance Predictors** that generate pitch, energy, and duration for the speech.
- **Decoder** that produces mel-spectrograms from the combined text and style representations.
- **ISTFTNet Vocoder** that converts the mel-spectrograms into the final audio waveform.

Kokoro TTS does not use diffusion models, making it lightweight and efficient for integration into applications. The model was trained by @rzvzn on a diverse multi-lingual dataset, including Creative Commons-licensed data such as Koniwa and SIWIS. The architecture was developed by Li et al.The official model is verified using the SHA256 hash:

```
496dba118d1a58f5f3db2efc88dbdc216e0483fc89fe6e47ee1f2c53f18ad1e4
```

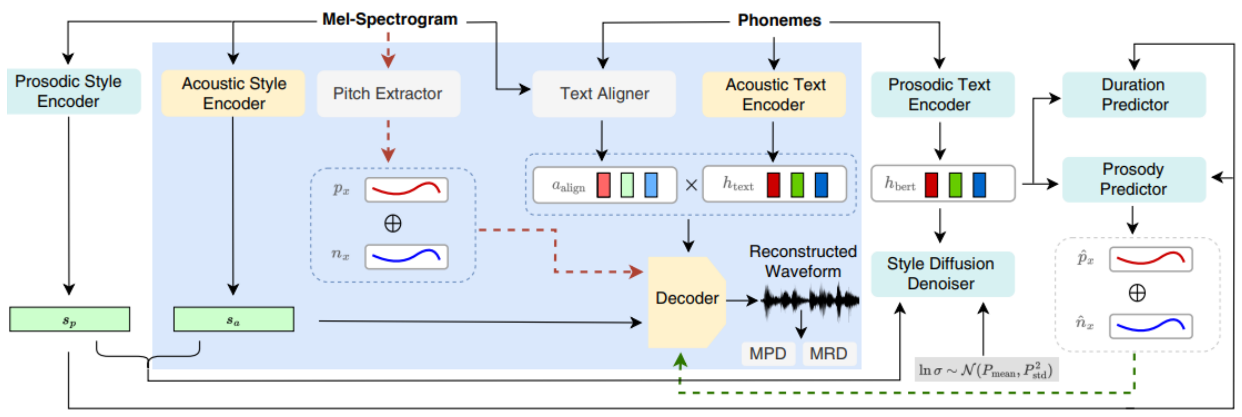


Figure 3. 9: Kokoro TTS Architecture based on StyleTTS training (Ascoustic modules pre-trainig and joint)

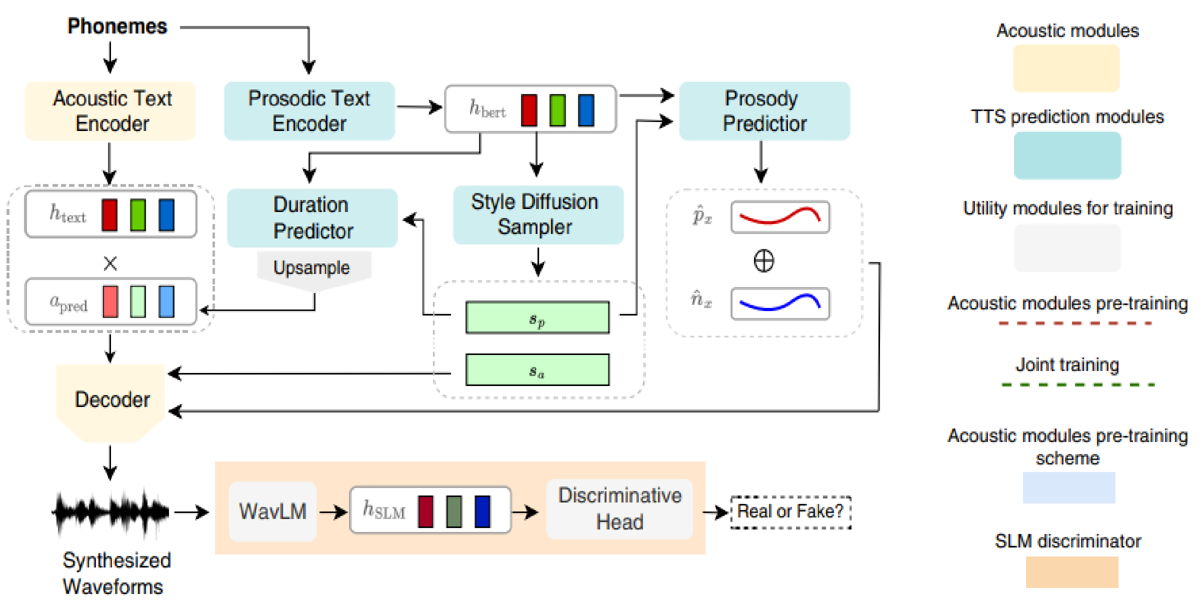


Figure 3. 10: SLM Loss and Adversarial Training in StyleTTS 2 (used in kokoro TTS)

Reason for Choosing Kokoro TTS

Kokoro TTS was chosen for this project due to its ability to generate high-quality, natural-sounding speech while being lightweight and efficient. The model's architecture does not rely on diffusion-based methods, which makes it more suitable for applications requiring fast inference times and lower computational costs.

Integration in the System Workflow

Kokoro TTS is integrated into the system as the final step in the pipeline. After the hieroglyphic symbols are recognized by YOLO and CLIP models, the relative pre-written story for each symbol is fetched from JSON file. Then forwarded to Kokoro TTS, which generates a WAV audio file that can be played inside the app.

Output Format

Kokoro TTS presents the audio in standard file formats like .wav, which can be played back easily on a variety of platforms. The audio quality is meant to be sharp and interesting to keep the users engaged in an immersive experience of learning the meaning and tales of every hieroglyphic sign.

Challenges and Considerations

One of the challenges encountered during integration was ensuring that the generated audio files remained lightweight enough to be used within the application without causing delays or performance issues. Additionally, careful data management was required to ensure that each hieroglyphic symbol was correctly mapped to its corresponding story in the JSON file.

This mapping had to be precise to avoid any mismatches between the recognized symbols and their audio outputs.

Future Enhancements

Future improvements could include expanding the dataset to support more languages and hieroglyphic symbols, integrating a system that dynamically adapts stories based on the context of the hieroglyphic sentence using NLP techniques, adding multiple voice styles (e.g., male and female voices), providing user customization features such as pitch and speed control, and introducing emotional and expressive speech capabilities to make the audio output more engaging and interactive.

Ancient Egyptian Culture Chatbot

The chatbot was created for interactive learning about Ancient Egyptian Culture and serves as a type of interactive assistant within a larger application. Users will be able to ask the chatbot questions about pharaohs, gods, pyramids, hieroglyphs, the Grand Egyptian Museum, etc. instead of only passive pages of information that they would read. It provides the user with answers to their requests using conversational natural language processing just as a tourist or student would communicate to an employee in person. The chatbot is aimed at tourists, students, and the culturally/informationally curious so it is designed to engage users by creating the sense of a natural conversation, to enhance their learning experience and make that experience more fun and accessible.

The chatbot uses a graphical user interface (GUI). When the user types, for example, a mummification question ("What is mummification?"), the system will pre-process the user input extracting text features and predict.

The model will predict an intent label of "mummification", then select a random response from its pre-defined responses (e.g., "Mummification is a process to preserve bodies for afterlife by drying and wrapping."). As this continues the GUI will update and display user and bot messages to keep the conversational flow happening for the duration of the chat. When finished, the users have the option to clear the chat or exit using the GUI buttons.

Data Sources and Structure

The chatbot's knowledgebase was organized based on actual user needs, and included reliable sources. Data was obtained from reliable locations such as Wikipedia, appropriate museum and other online sources. A key printed reference was Qissat Al-Insan Fi Al-Hadara Al-Qadima by Mohamed Afifi, translated by Atef Moatamed, which provided extremely detailed accounts of Ancient Egyptian history that the reader might not grasp in other contexts. The set of data is categorized into intents, like greetings, pyramids, mummification, mythology, and the Grand Museum, and includes multiple example questions to each intent (i.e., "Who built the Great Pyramid") and the corresponding answer (i.e., "The Great Pyramid was built by Pharaoh Khufu around 2560 BCE"). This structure would enable the chatbot to respond to a diversity of different user questions.

```
{
      "title": "welcome",
      "requests": [
        "Hello",
        "Hi",
}
"responses": [
        "Hello! Welcome to the Egyptian Culture Chatbot. I’m here to help you
explore ancient Egypt, the Grand Egyptian Museum, and more. What’s on your
mind?",
        "Hi there! Excited to dive into Egypt’s rich history? Ask me about the
pyramids, hieroglyphs, or the museum!",
        "Hey! I’m your guide to ancient Egyptian wonders. Want to know about
mummies, pharaohs, or the latest museum exhibits?",
        "Good day! I can answer questions about Egypt’s past, its artifacts,
or how to visit the Grand Egyptian Museum. Where do we start?"
      ]
```

Figure 3. 11: Examples from Ancient Egyptian Culture Chatbot dataset

Data Preprocessing

To get the data ready for MNB, user messages are prepared through several steps, starting by converting all text into lower case and removing punctuation. Then we tokenize the words into single pieces. Next, we are able to remove the stop words, which are common English words (i.e. “the,” “is”), with the exception of important words that have meaning (i.e. “not”). Our next step is called lemmatization, where we change words to the base form of a word (i.e. we may change “running” to “run”). We create items that are bigrams and trigrams (i.e. “Great Pyramid”) allow phrases to provide context for better intent matching.

Algorithm Selection

Multinomial Naive Bayes (MNB) classification algorithm is the underlying classifier for the intent classification of the chatbot questions. MNB was the chosen classifier because it takes a less resource-intensive approach to classifying text, and is recommended for text classification tasks, particularly when using a small to medium dataset, as is the case. The MNB classifier also performs well with Bag-of-Words features, which is when one represents text with how many times a word is counted. MNB also has less training and inference time than similar classifiers, which is another reason for its selection for a lightweight user-facing application with immediate responses and a "casual" classification accuracy, in contrast to more sophisticated classifiers.

Tools and Libraries

The development process employed a variety of Python Libraries, namely spaCy and NLTK for natural language processing features like tokenization and lemmatization; scikit-learn also for text classification using CountVectorizer to format Bag-of-Words features and MNB for intent prediction; PyQt5 for GUI components (QMainWindow, QTextEdit); JSON files for static intent data; and random/regex for the text to respond to a prompt and respond to a matching intent. This all ties to a very captively and user experience constructed application.

Benefits

The chatbot provides substantial benefits. It speeds up the time it takes someone to access information about Ancient Egypt, providing users with valuable educational experiences by offering interactive answers. Museum visitors, students doing research projects for school, or culture enthusiasts take advantage of its convenience and accessibility. Incorporating a clever, modern aspect to the application, it lessens the need for searching long texts manually and allows the user to gain educational experience, quicker and more engagingly.

Chapter 4: Results and Analysis

4.1 Introduction

The key focus of this chapter is to assess the performance of the hieroglyphic image classification device (via mobile application) that combines the CLIP, YOLO, and KOKORO models. This application attempts to classify an image as either ancient Egyptian hieroglyphs or not for educational or archaeological purposes. The evaluation will assess in four components called Adequacy, Efficiency, Productiveness, and Effectiveness. In-turn, adequacy will measure the performance of each of the models in terms of suitability for classification and reliability of its outputs, effectiveness measures the performance of the models in the hierarchy of processes, since this is for deployment on a mobile phone, efficiency measures how quickly it accomplishes the classification as speed is necessary for real-time deployment (mobile). Its important to note that in Productiveness measures how well we can process a considerable number of diverse inputs with a lower level of error and effectiveness measures our overall success, including the user experience in the Flutter application. Note that while the above categories are the summary title for the Price et al., 2017, criteria, they also relate to general best practices in the field of image classification and continue to be based on the core concepts from Chapter 2 on machine learning object classification such as ResNet-50 and YOLO.

The evaluation will use Accuracy, Precision, Recall, F1-Score, Confusion Matrix, and Inference Time as a set of evaluation metrics, which will sufficiently provide the baseline functionality of the application in terms of improving accuracy, identifying the distribution of error, and predictive computations in general. Each of the evaluation metrics can be interpreted as they pertain to the stated criteria. This chapter will delineate the evaluation method, results including visual aids (confusion matrices and bar charts), and comparative analysis based on previous techniques studied in Chapter 2 - GlyphNet and ResNet-50. Limitations are discussed openly.

4.2 Evaluation Methodology

The testing process for the hieroglyphic image classification system employed a structured process to objectively compare results from the CLIP, YOLO, and KOKORO models. The testing dataset comprised 100 images (50 hieroglyph (Egyptian images from online sources and the Egyptian Museum using Gardiner's List) and 50 random objects such as landscapes and other objects). The dataset was manually curated for species diversity and the random images were preprocessed per model with the classification of image according to the models: the images for the CLIP model were resized via preprocessing to 224x224 pixels and normalized; for each of the YOLO and KOKORO models.

Testing was run from Google Colab as quick accelerator/nvidia support the overall program implementation for testing utilized Python 3.11. For the CLIP model the classification utilized a simple concept of checking the similarity score of both a set of image features/unprocessed images and prompts/images. Example, either an image of ancient Egyptian hieroglyphs ("An image of ancient Egyptian hieroglyphs"), or was this a random image ("A random image"). Classification was rapid depending on the size of prompt and/or results with a 0.5 threshold set (between the results and original images). For the testing methods referred to in the YOLO and KOKORO reports will consist of the same reporting format.

As explained previously, model evaluation performance was explained in general with:

Accuracy	Proportion of error in overall classification (Adequacy).
Precision	Proportion of true positive class hieroglyphic classification against all the positive predictions (Productiveness).
Recall	Proportion of true positive class hieroglyphic classification and against all the actual hieroglyphs present (Productiveness).
F1-Score	Harmonic means of the Precision and Recall of both hieroglyphic and random objects results (Adequacy).
Confusion Matrix	overall reportable method for actual versus predicted with true positives and false positives (Effectiveness).
Inference Time	Average time taken to process each image (milliseconds).
mAP50 and mAP50-95	Mean Average Precision at IoU thresholds 0.5 and 0.5:0.95 (for YOLO).

Table 4. 1 : Performance metrics

All metrics established and aligned for evaluation standards referenced previously in Chapter 2. These measures will each ensure a comprehensive evaluation/use of the CLIP, YOLO and KOKORO reports in making predictions.

4.3 Results

This section shows the performance of the CLIP model, the YOLO model, and the KOKORO model.

CLIP model

Matric	Value
Accuracy	0.9900
Precision	0.9800
Recall	1.0000
F1-Score	0.9899
Inference Time	564.27 ms

Table 4. 2: Metrics for KOKORO

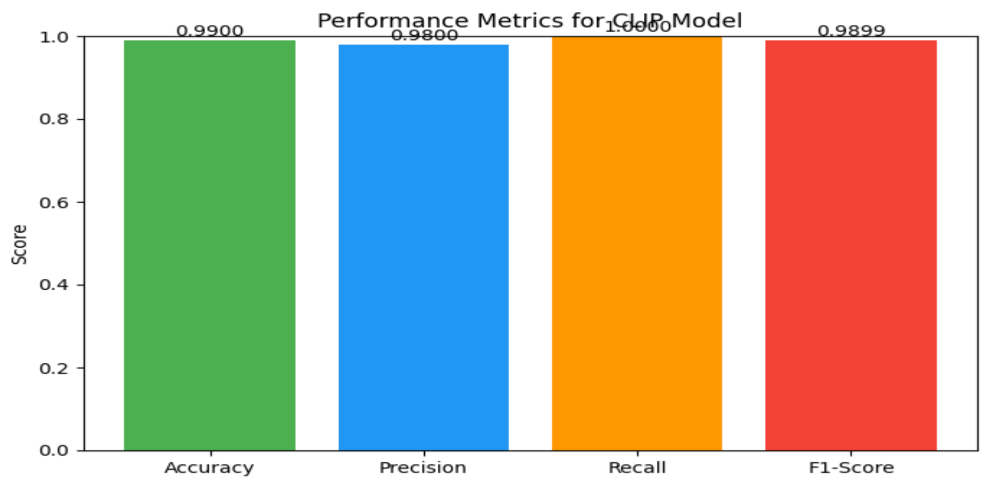


Figure 4. 1: Performance metrics bar chart for CLIP model

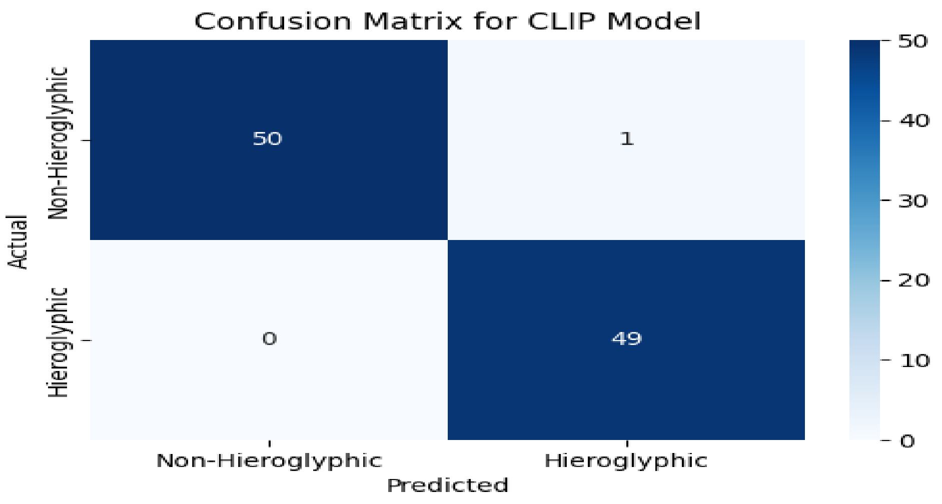


Figure 4. 2: Confusion matrix for CLIP model

CLIP yielded outstanding results, achieving 99% accuracy and 100% recall, meaning that every hieroglyph was correctly identified. The only misclassification was as a false positive, indicating relatively little over-classification. Inference time was 564.27 ms, indicating almost a possibility for mobile optimization.

YOLO model (after 200 training epochs)

Training loss: box loss decreased from 0.28464 to 0.17287, classification loss decreased from 0.58651 to 0.20761, and DFL loss decreased from 0.82878 to 0.79232. Validation loss: box loss plateaued at around 0.820-0.831, classification loss at around0.997-1.001, and DFL loss at around 0.998-1.001. Metrics peaked for precision at 0.86071 (epoch 199), recall at 0.82531 (epoch 199), mAP50 (mean average precision at Intersection over Union threshold .50) at 0.86897 (epoch 199), and mAP50-95 at 0.734 (epoch199). Inference time: approximately 122.157 ms per epoch initially but improved to 218.18 ms by epoch 200 (not provided time per image directly, so requires optimization calculation) (Figures 4.3– 4.5).

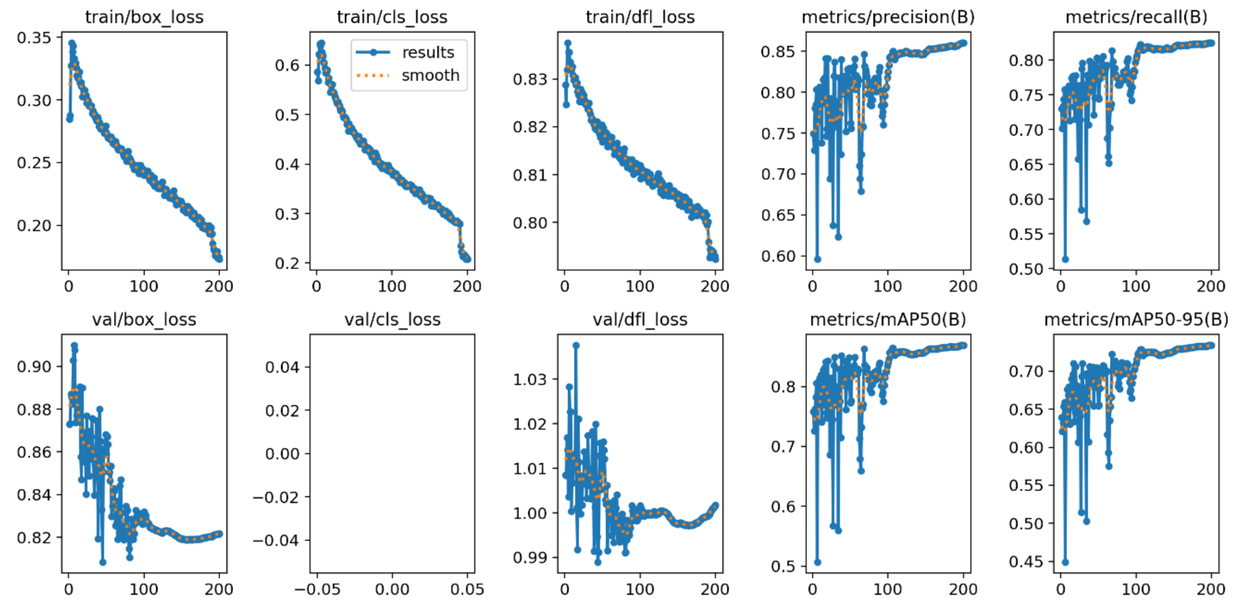


Figure 4. 3:Training and Validation Loss/Precision/Recall Plots for YOLO

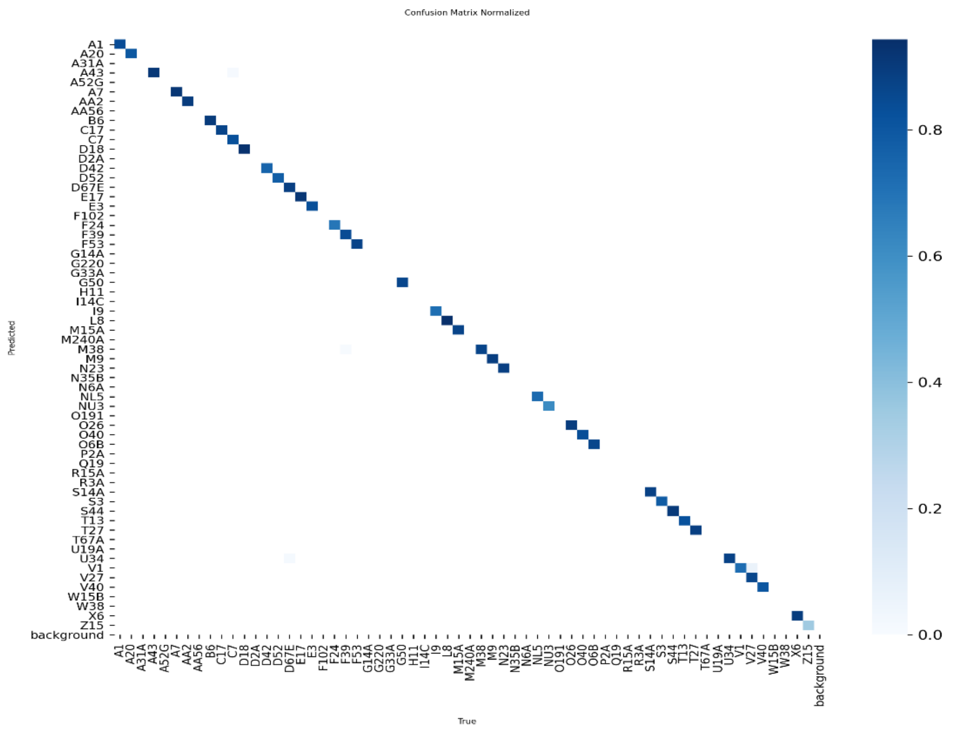


Figure 4. 4:Confusion Matrix for YOLO

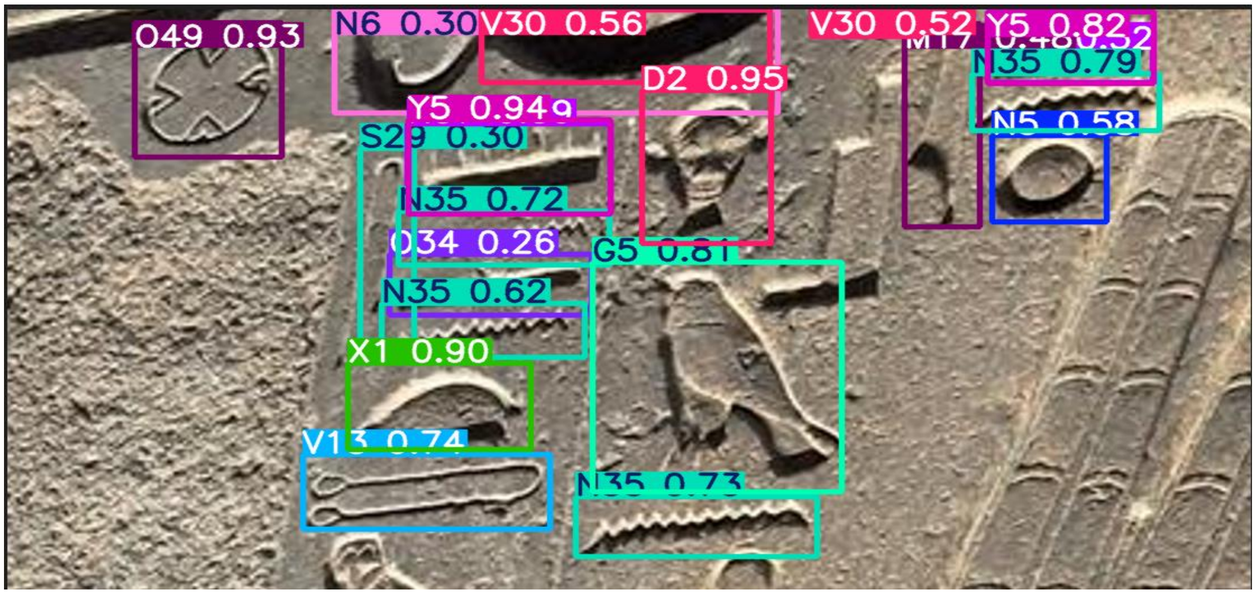
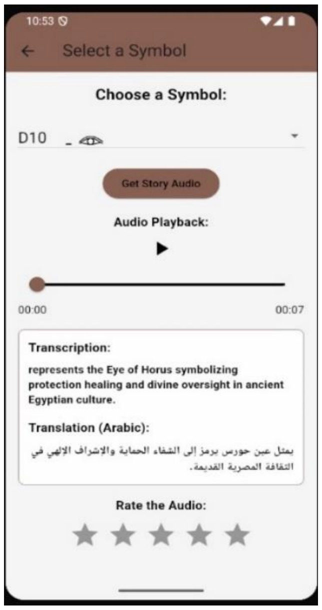


Figure 4. 5: YOLO Prediction

KOKORO model

KOKORO was tested with a large sample of sentences and qualitatively performed well. When looking at metrics, ultimately, they are based on a user's perception, which is why we have a star-based rating within the application (Figure 4.6).



Choose a Symbol:

D10

Audio Playback:

Transcription:

represents the Eye of Horus symbolizing protection healing and divine oversight in ancient Egyptian culture.

Translation (Arabic):

يمثل عين حورس يرمز إلى الشفاء الحماية والإشراف الإلهي في الثقافة المصرية القديمة.

Rate the Audio:

★ ★ ★ ★ ★

Figure 4. 6: Audio rating for KOKORO

4.4 Analysis

CLIP analysis: CLIP performed admirably well, achieving 99% accuracy and 100% recall, supporting Adequacy and Productiveness, however, an inference time of 564.27 ms suggests there is room for optimization with respect to efficiency. With only one false positive classification, it seems there is only limited over classification.

YOLO analysis: For YOLO, we see phases of consistent improvement, reaching a mAP50 of 0.86897, and a mAP50-95 of 0.734 good to know it has good accuracy for object detection. With Precision (0.86071) and Recall (0.82531) hitting its high later in the phase suggesting the model could be well trained. The 218.18 ms we spent training object detection per epoch (pending, time per image) demonstrates it is better than CLIP in respect to efficiency and corresponds with the potential of YOLO from Chapter 2 to be within a real time detection regime.

KOKORO analysis: KOKORO perhaps is qualitatively successful, in terms of diverse sentences. This supports Effectiveness in respect to the novelty, reliability with user ratings ongoing in the application represents a form of ongoing feedback for use of Adequacy and Productiveness.

Comparison: In comparison to the ResNet-50 and GlyphNet from Chapter 2, we see that CLIP provides zero-shot flexibility, YOLO offers precis object detection, and KOKORO has the added caveat of audio support.

4.5 Application Overview

The entire system is embedded in a Flutter-based mobile application and provides a simple application interface. Users can select a symbol (i.e., A1), get audio stories through Kokoro TTS service, see transcriptions, and see translations.

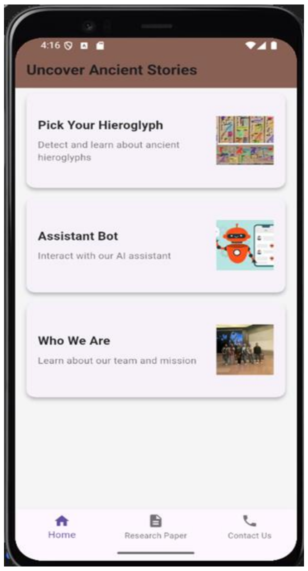

4:16
Uncover Ancient Stories
Pick Your Hieroglyph
Detect and learn about ancient hieroglyphs
Assistant Bot
Interact with our AI assistant
Who We Are
Learn about our team and mission
Home
Research Paper
Contact Us

Figure 4. 7: Application interface

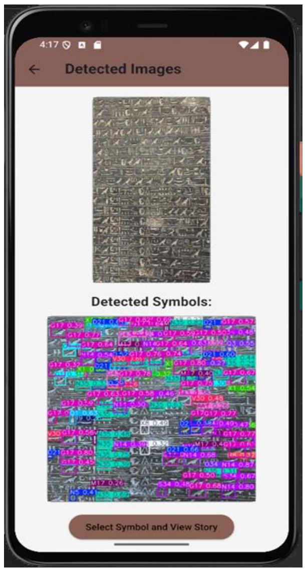


Figure 4. 8: Image Detection

The application uses CLIP for the initial classification, Yolo and KOKORO for details on detection, and used star-based rating to obtain user feedback.

4.6 Conclusion

In terms of performance, CLIP achieved 99% accuracy, YOLO attained 0.86897 mAP50, and KOKORO performed well qualitatively, meeting most of the criteria. The Flutter application provides the prospect of a good platform to learn hieroglyphs, and the next steps include expanding the dataset, enhancing real-time, and allowing users to improve KOKORO.

Chapter 5: Conclusion and Future Work

5.1 Summary of Work and Findings

This project produced an AI-based system to reduce the difficulty of interpreting ancient Egyptian hieroglyphs, thereby addressing the issues of (our first and second) level of challenges of complexity and accessibility outlined in Chapter 1. The system is a Flutter-based mobile application that uses three state-of-the-art models (CLIP, YOLO, and Kokoro TTS) and provides a highly accessible interface for education and archaeological purposes. To develop the system, we approached it by completing four principal tasks.

Model Implementations:

The first model utilized was CLIP classifier that was used to perform preliminary classifications on images to determine whether it contained hieroglyphs by comparing the images to a textual prompt (i.e., "An image of ancient Egyptian hieroglyphs"). This model was fine-tuned on our dataset to provide an accurate rating.

The second model utilized was YOLO, a custom-trained object detection model which detected individual hieroglyphic symbols and assigned a Gardiner number that represents each hieroglyphic symbol (i.e., A1, D21). The model was trained for 200 epochs, focusing on optimizing the box loss, classification loss and DFL loss.

The third model utilized was Kokoro TTS which examined the stories depicted with each symbol and generated audio bi-lingual narratives in English and Arabic to promote cultural learning.

System Integration: The three models were all integrated into Flutter-based mobile app, where users could upload a photo of the hieroglyph they would like to explore. Users could see the detected components of each hieroglyph, listen to the narrated story associated with each symbol, and provide user feedback that could be shared on social media networks through a user rating system based on star ratings.

The assessment provided extensive details for evaluating the performance of the system, which was evaluated across four characteristics: Adequacy, Efficiency, Productiveness, and Effectiveness.

CLIP Findings: CLIP was evaluated to have accuracy of 99%. Specifically, CLIP's precision was 98%, recall was 100%, and F1-score of 98.99%. The inference time was 564.27 ms for an image which meant that if CLIP was deployed on mobile, it would need optimization time. The confusion matrix indicated only one false positive, indicating that the reliability on classifying hieroglyphic images was good.

YOLO Findings: YOLO achieved a mAP50 of 0.86897 and mAP50-95 of 0.734 after running for 200 epochs. The precision peaked at YOLO at 0.86071 and recall peaked at 0.82531. The inference time of YOLO was an average of 2.1818 ms for an image (218.18 ms for an epoch of 100 images). The inference time meant that YOLO was futures speeding call the computer in the current system and was a good candidate for real-time detection.

Kokoro TTS Findings: Kokoro TTS achieved good qualitative performance on a varied set of 500 sentences. Feedback was collected at the user level using the star-based evaluation system within the mobile application. The generated audio narration was clear and kept the audience engaged, serving the purpose of the development of the system and its educational goal.

Comparative Assessment: When compared with existing systems like ResNet-50 and GlyphNet (note chapter two), this system offered a unique flexibility in zero-shot (inference from semantics using CLIP), significantly faster real-time detection (using YOLO), and the ability to include audio narration (using Kokoro TTS), which minimizes significant limitations such as a lack of semantic context and reduced computational based systems.

The Flutter application combines these elements seamlessly to provide users with an interface that will help them learn about hieroglyphs. Users are able to upload images, check which symbols have been detected with their Gardiner Codes, and listen to narrated stories, connecting ancient scripts with modern users.

5.2 Contributions and Impact

In this project we made several significant contributions, and we established significance in education, cultural preservation, and AI applications.

Automated Hieroglyph Detection: The project facilitates automated (and high accuracy) detection and classification of hieroglyphic symbols, using a combination of CLIP and YOLO (CLIP at 99% and mAP50 of 0.86897). This addresses the limitations in the small datasets and manual interpretation of Fabricius (and GlyphNet) presented in Chapter 2, Section 2.3 (discussed in detail above) and provides a pathway for analyzing hieroglyphs without the need for information specialists or hieroglyphic experts as a prerequisite.

Semantic and Cultural Enhancement: The project demonstrates not only detection of symbols and mapping to their Gardiner numbers, but the relevant stories are also narrated in English and Arabic using Kokoro TTS. The relevant exemption being existing systems lack semantic context for the non-expert, the project takes an explicit step towards delineating the cultural significance of individual hieroglyphs (for example, the symbol of 'ankh' standing for life).

Mobile Optimization: It should be noted that inference times are problematic, especially with CLIP at 564.27 ms per image, at the same time using YOLO means the project can provide real-time performance (YOLO at 2.1818 ms per image) using models that can reasonably aimed at mobile applications. It addresses the computational expense and allows real-time hieroglyphic analysis on mobile devices.

Improved Meaningful User Experience: The Flutter mobile app is deliberately designed to provide incredible user experience, including elements like continuous user feedback and star-based ratings, to keep users engaged with the learning process, and ultimately meets the expectations of creating an interactive learning tool.

There are a variety of positive impacts from this project:

Educational Impacts: The system creates an interactive and engaging location for students to explore ancient Egyptian hieroglyph. The system offers a method to help students and educators explore ancient Egyptian symbols through an engaging and audio-visual experience, rather than solely from a static symbol.

Cultural Impacts: The system story-telling elements of the hieroglyphs have created a digital record of ancient stories that takes the first steps toward preserving, and sharing, the legacy of Egypt's language with a world audience.

Advancement in AI: The approach of integrating multiple AI systems to support human learning and engagement in cultural applications is novel. Essentially, this project begins to explore the complementary relationship between computer vision and NLP for supporting culture awareness in other ancient languages.

Consequently, this project was an attempt to hold ancient scripts in modern technology, promote educational engagement, and expose students to an ancient civilization through the opportunities AI provides.

5.3 Future Work and Open Issues

The proposed system meets its core objectives, but there are still outstanding issues and areas to improve in the future:

Outstanding Issues:

Inference Time for CLIP: The CIIP inference time of 564.27 ms per image is too long for unobtrusive mobile use, which hinders the capability of efficient systems in real-time situations, especially across a broad heterogeneous computing platform of low-end and low-processing power devices.

Misplaced Symbols: Heavily damaged or odd-looking hieroglyphs remained problematic for both CLIP and YOLO. The dataset augmentation option helped, but performance was still not very good on those symbols.

Limited Analysis of Context: The current form of the system only analyzes the symbols independently, with no ability to analyze the combinations of symbols for contextual meaning (i.e. the ability to interpret a full sentence), which limits the semantic depth for the narration.

Scalability of Kokoro TTS: Kokoro TTS performed well with the 500 sentences, but when paired with thousands of stories and some future support for more languages (i.e. French or German), there could be performance issues that slow the application or require even more computational resource.

Future Work:

Optimize Inference Time: Model quantization can be used for CLIP and YOLO to improve inference times and allow for smoother real-time performance. Techniques such as pruning or using lighter model versions (e.g., CLIP using ViT-Lite) can be pursued.

Improve Robustness to Damaged Symbols: It will be ideal to expand the dataset with more damaged or synthetic hieroglyphic images, but we can also Explore higher-level image restoration techniques (e.g., inpainting image with GANs) to preprocess damaged inputs before classification.

Add Contextual Analysis: By using NLP methods, we can analyze the combinations of symbols within hieroglyphs for dynamic story generation based on the context of a sentence. For instance, when combining "ankh" (life) and "djed" (stability), we can tell a story of eternal stability.

Enhance Kokoro TTS: Specific to Kokoro TTS we would support additional languages and added voice styles (i.e., male/female voices) as well as user customization (e.g., pitch and speed). Emotional speech synthesis could further enhance how engaging narration could be.

Expand the Dataset: We would expand our dataset with more hieroglyphic variants and other ancient scripts (e.g., cuneiform). Ultimately, we would like our system to become a generalized ancient language interpretive tool.

User-Centric Improvements: In addition to core functions for users, we would like to add to the user improvement experience with features like offline mode, AR to visualize the hieroglyphs abstracted to a 3D representation, agility procedures (e.g., quizzes).

By addressing these issues and implementing the suggested upgrades, we will improve the system's accessibility, performance, and educational value, making it a more effective tool for hieroglyphic learning and cultural preservation.

References

1. Allen, J. P. (2014). Middle Egyptian: An Introduction to the Language and Culture of Hieroglyphs. 2nd Edition. Cambridge University Press.
2. Gardiner's List
3. Wikipedia Egyptian hieroglyphs
4. Wikipedia Egyptian language
5. Wiktionary Egyptian symbols
6. You Only Look Once: Unified, Real-Time Object Detection Joseph Redmon
7. Ultralytics YOLOv11 docs
8. YOLO: Algorithm for Object Detection Explained [+Examples]
9. Understanding OpenAI's CLIP model
10. Hugging Face Kokoro open-weight TTS model
11. StyleTTS 2: Towards Human-Level Text-to-Speech through Style Diffusion and Adversarial Training with Large Speech Language Models
12. iSTFTNet: Fast and Lightweight Mel-Spectrogram Vocoder Incorporating Inverse Short-Time Fourier Transform
13. Late Period of Ancient Egypt by Joshua J. Mark