Helwan University

Faculty of Engineering

Department of Computer and Programming

B.sc Final Year Project

Hieroglyph signs decoding

BY

Remon Samir Zaki

Omar Mohamed Othman Ahmed

Michael Samir Youssef Fam

Amr Ahmed Mohamed Mahmoud

Ahmed Khalil Mohamed Khalil

Mohammed Rabie Khames

Supervised by

Assoc.Prof. Shahira

Graduation Project 2020/2021

Table of content

Abstract

Tourism is a very important topic for Egypt economy and is continuously growing nowadays. Many tourists come to Egypt to see the beautiful Pharaonic antiquities and read the Egyptian hieroglyphs to know more about the old Egyptian habits and how they lived.

Ancient Egyptian writing is known as hieroglyphics ('sacred carvings') and developed at some point prior to the Early Dynastic Period (c. 3150 -2613 BCE). According to some scholars, the concept of the written word was first developed in Mesopotamia and came to Egypt through trade. While there certainly was cross-cultural exchange between the two regions, Egyptian hieroglyphics are completely Egyptian in origin; there is no evidence of early writings which describe non-Egyptian concepts, places, or objects, and early Egyptian pictographs have no correlation to early Mesopotamian signs. The designation 'hieroglyphics' is a Greek word; the Egyptians referred to

their writing as medu-netjer, 'the god's words,' as they believed writing had been given to them by the great god Thoth.

Hieroglyph Signs Decoding app comes to support tourism and researchers who interests in this field using machine learning. It is Flutter app that helps the tourist to understand Egyptian Hieroglyphs instead of just watch them by translation process to English. Now just by taking a photo from your phone to the part of text you want to translate you can read in English the Pharaonic words and understand their great civilization by just a shot.

Our project is a Flutter app so GUI is made by dart programming language. The main functionality of our app is to translate Hieroglyphic sentences to English language by using python programming language and Machine Learning algorithms that manipulate data of the input image and using Deep learning model we transform input image to Gardiner's code then by Natural Language Processing (NLP) we transform Gardiner's code to English sentences. We also use Augmented Reality (AR) to enhance the visuals of the application and give the user the right atmosphere by displaying 3D statues for kings beside their name in the Cartouches. Then we connect our mobile app with python code through Django server that run the python code.

Our project has a unique and new idea that can be used in many fields. The main aim of our project is to help in tourism promotion in Egypt using machine learning and to ease the complication of ancient Egyptian glyphs written in touristic places so the tourist journey becomes more valuable and interesting.

Our app can be used by researchers who are interested in this field to act as a tool that helps them in to ease their studying process and fasten their work.

Chapter One

Introduction

Introduction

Egyptian Hieroglyphs that are written on ancient temple walls and ancient Pharaonic antiquities describe a great part of Egypt history. Hieroglyphs were written on papyrus, carved in stone on tomb and temple walls, and used to decorate many objects of cultic and daily life use. Altogether there are over 700 different hieroglyphs, some of which represent sounds or syllables; others that serve as determinatives to clarify the meaning of a word. Using our program these sign on walls can be easily read in English.

This project aims to promote tourism using machine learning and help tourists to understand the old Egyptian Hieroglyphs in an easy way by taking a photo of the text to be translated and the program will process this photo and return the text in English. The program also help researchers in this field to fasten their work as the program acts as a tool to translate any Hieroglyphic sentences.

Problem Statement

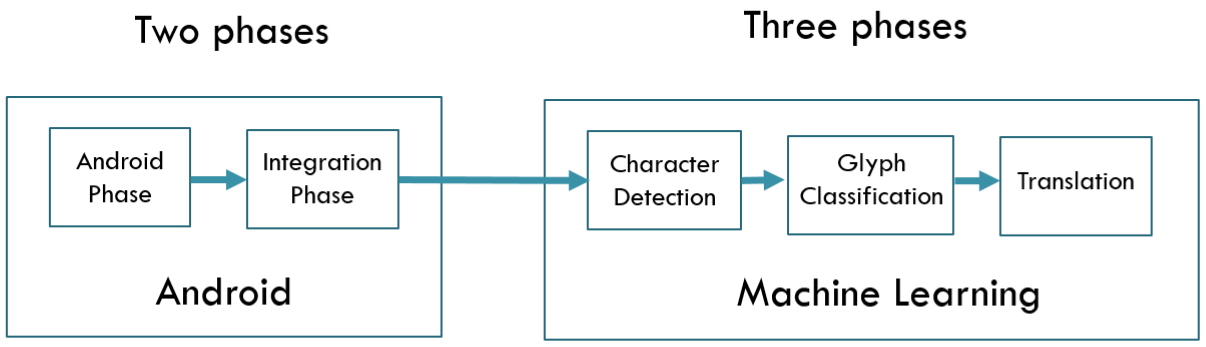
Knowledge of the hieroglyphs had been lost completely in the medieval period. All medieval and early modern attempts were hampered by the fundamental assumption that hieroglyphs recorded ideas and not the sounds of the language. As no bilingual texts were available, any such symbolic 'translation' could be proposed without the possibility of verification. The breakthrough in decipherment came only with the discovery of the Rosetta Stone by Napoleon's troops in 1799 (during Napoleon's Egyptian invasion). As the stone presented a hieroglyphic and a demotic version of the same text in parallel with a Greek translation, plenty of material for falsifiable studies in translation was suddenly available.

So using **Machine Learning** and **Deep Learning** we can make a huge difference in presenting this translation on android application and we also used **Augmented Reality** to detect cartouche and statues of ancient Egyptian Kings and produce sound that talks about their great history and a 3D statue of this king will appear in the real world in the camera to give the tourist a great visualization and deep feeling of the historical place environment.

Our aim is to give any researcher or student in Hieroglyph writing and translation field a unique tool to help him continue studying and search in an easy way.

Project Phases and Objectives

1) Flutter Phase
2) Integration
3) Character Detection
4) Glyph Classification
5) Translation
6) Augmented Reality



Objectives

1) Flutter Phase

This phase contains GUI and has an important feature as it is working on Android and IOS. This phase contains a complicated step that joins Augmented Reality in Unity game engine with Flutter.

2) Integration

Integration phase is concerned to running server on Django and connects the Flutter app with ML and Deep Learning python code as this code runs on this server.

3) Character Detection

This phase takes a Hieroglyph wall image and detect contours that represent the Hieroglyph language characters extract them to small images in the right reading order of the Hieroglyph line.

4) Glyph Classification

Using ML and Deep Learning we classify glyphs

5) Translation

Translation is done by taking the Gardiner's code from the previous step and apply ML algorithms on it to get the English translation. We apply grammar correction to the output sentence to be organized in a correct manner.

Project Motivation

Tourism is one of the leading sources of income, crucial to Egypt's economy. At its peak in 2010, the sector employed about 12% of workforce of Egypt, serving approximately 14.7 million visitors to Egypt, and providing revenues of nearly $12.5 billion as well as contributing more than 11% of GDP and 14.4% of foreign currency revenues.

So we wish to contribute in tourism promotion and helping in increasing the ability to give the tourist a valuable and interesting journey in Egypt by the aid of Augmented reality that give a high quality visualization and a very good experience. It also make the tourist live in the atmosphere of historical places and know more about what he is seeing. We also give the researchers a unique and valuable tool that help them in their field as studying Egyptian Hieroglyphs is not easy but now they can translate any text they want to know its meaning in English.

Chapter Two

Background

Related Work

As the idea is unique there are not many resources to check but we can make use of some research papers or books that talks about things related to our project phases. We divide our project to five phases that we can search in previous work to help us and we checked some books and papers in these fields:

1- Image and text recognition using machine learning techniques like (OCR)

2- Training model to convert each letter in Hieroglyphs from image to letter (as Gardiner code) using machine learning model like SVM model

3- Translate Hieroglyph sentences to English sentences using machine learning concepts like N-gram

We make use of some papers and books like:

1- Rimon Elias, Image Based Hieroglyphic Character Recognition, Cairo, Egypt, 2018

2- Reham Elnabawy, Extending Gardiner's code for Hieroglyphic recognition and English mapping, Cairo, Egypt, 2020

3- Mark-JanNederhof, OCR of hand written transcriptions of Ancient Egyptian hieroglyphic text, University of St Andrews, 2004

We were inspired by Morris Franklen's "Automatic egyptian hieroglyph recognition by retrieving images as texts" which provided a dataset of more than a 7000 labeled images –covers almost 171 different glyphs– from the pyramid of Unas. This paper talks only about the concept when dealing with Hieroglyphic letters recognition and some useful information.

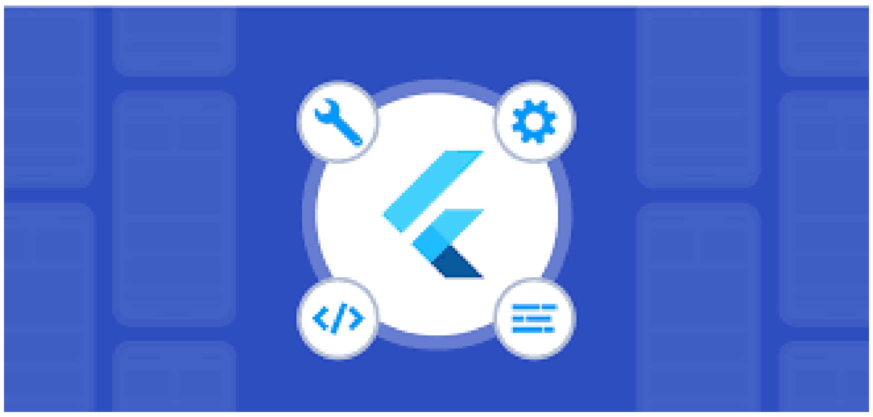
Flutter Overview

Flutter is a cross-platform UI toolkit that is designed to allow code reuse across operating systems such as iOS and Android, while also allowing applications to interface directly with underlying platform services. The goal is to enable developers to deliver high-performance apps that feel natural on different platforms, embracing differences where they exist while sharing as much code as possible.

During development, Flutter apps run in a VM that offers stateful hot reload of changes without needing a full recompile. For release, Flutter apps are compiled directly to machine code, whether Intel x64 or ARM instructions, or to JavaScript if targeting the web. The framework is open source, with a permissive BSD license, and has a thriving ecosystem of third-party packages that supplement the core library functionality.

Let us divide this into a number of sections:

- The layer model: The pieces from which Flutter is constructed.
- Reactive user interfaces: A core concept for Flutter user interface development.
- Widgets: The fundamental building blocks of Flutter user interfaces.
- The rendering process: How Flutter turns UI code into pixels.
- Platform embedders: The code that lets mobile and desktop OSes execute Flutter apps.
- Support for the web: Concluding remarks about the characteristics of Flutter in a browser environment.



Django Overview

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation

Django helps you write software that is:

Complete

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation.

Versatile

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc).

Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

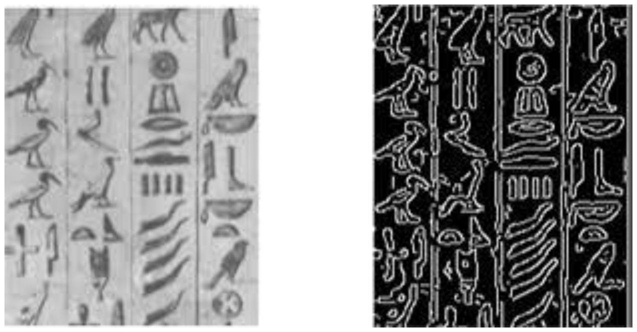
Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

Digital Image Processing

OCR Overview

Optical Character Recognition (OCR) is a process run by an OCR software. The software will open a digital image, e.g. a tiff file containing full text characters, and then attempt to read and translate the characters into recognizable full text and save them as a full text file. This is a quick process that enables automated conversion of millions of images into full-text files that can then be searched by word or character. This is a very useful and cost efficient process for large scale digitisation projects for text based materials including books, journals and newspapers.



The OCR process is dependant upon a number of factors and these factors influence results quite radically. Experience to date has shown that using OCR software over good quality clean images (e.g. a new PDF file) has excellent results and most characters will be recognized correctly therefore leading to successful word searching and retrieval. However over older materials e.g. books and newspapers the OCR is extremely variable and for this reason some projects advocate re-keying the text from scratch, rather than attempting OCR.

The process is labour intensive and sometimes a combination of both re-keying and OCR will be performed for a project. It is usual to undertake sample tests on the actual source material to be digitised before making decisions about OCR and re-keying.

Image Filtering

Image filtering is changing the appearance of an image by altering the colors of the pixels. Increasing the contrast as well as adding a variety of special effects to images are some of the results of applying filters. In order to obtain a high success rate of OCR (optical character recognition) performed on text images, the main target of filters is, however, to reduce the noise around characters in the image. Thereby, I created two new nonlinear efficient image filters (for RGB and ARGB images) and elaborated the optimization technique suitable for each of the filters in order to make them perform faster.

is carried out to produce an altered image that is more suitable for the intended application. Image filtering and image warping are considered to be the most common methods of image processing. Within image filtering, these operations are carried out to aid the altering or enhancing an image to either remove specific features or highlight features of interest within the image. These operations are generally carried out in the pre-processing stage and can have very positive results on the quality of feature extraction and the results of image analysis.

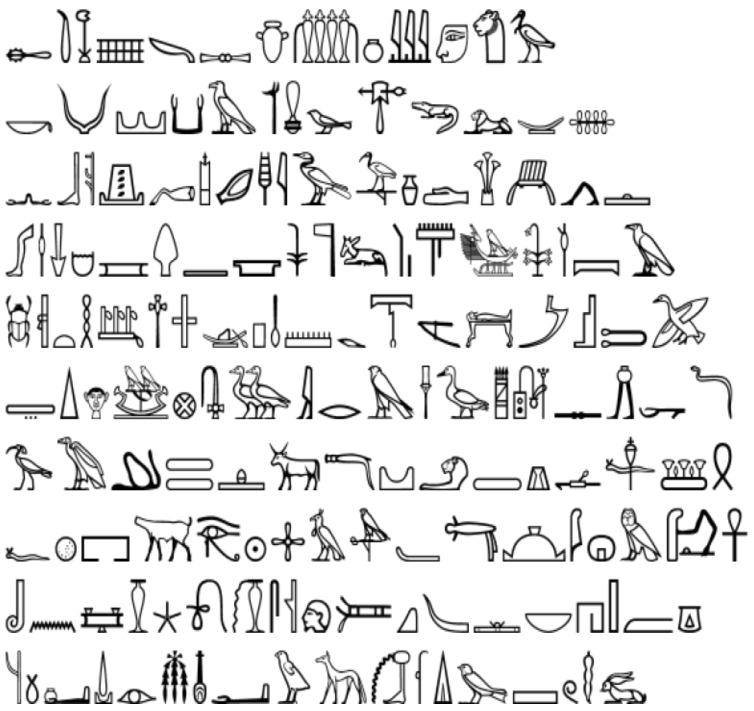
Image segmentation

Image segmentation is one of the hotspots in image processing and computer vision. It is also an important basis for image recognition. It is based on certain criteria to divide an input image into a number of the same nature of the category in order to extract the area which people are interested in. And it is the basis for image analysis and understanding of image feature extraction and recognition.

With image segmentation, each annotated pixel in an image belongs to a single class. It is often used to label images for applications that require high accuracy and is manually intensive because it requires pixel-level accuracy. A single image can take up to 30 minutes or beyond to complete. The output is a mask that outlines the shape of the object in the image. Although segmentation annotations come in a lot of different types (such as semantic segmentation, instance segmentation, panoptic segmentation, etc), the practice of image segmentation generally describes the need to annotate every pixel of the image with a class.

Hieroglyphs Classification

Hieroglyphs were mysterious codes for a long time and still mysterious to many laymen as it consist of ideograms. Our dataset consists of more than a 7000 labeled images –covers almost 171 different glyphs– from the pyramid of Unas.



HOG

The Histogram of Oriented Gradients (HOG) is a feature descriptor used in computer vision and image processing. The HOG descriptor have a key advantages over other descriptors because it operates on localized cells. It’s invariance to geometric transformations but sensitive to object orientation.

Histogram of Oriented Gradients (HOG) is computed by:

1) Global image normalization –optional–.

2) Get gradient of the image in x and y direction.

3) Get orientations histograms.

4) Normalize across blocks.

Gradients operators

$$G_x = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}, G_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

gradient magnitude and orientation equations

$$|G| = \sqrt{G_x^2 + G_y^2}, \Phi = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

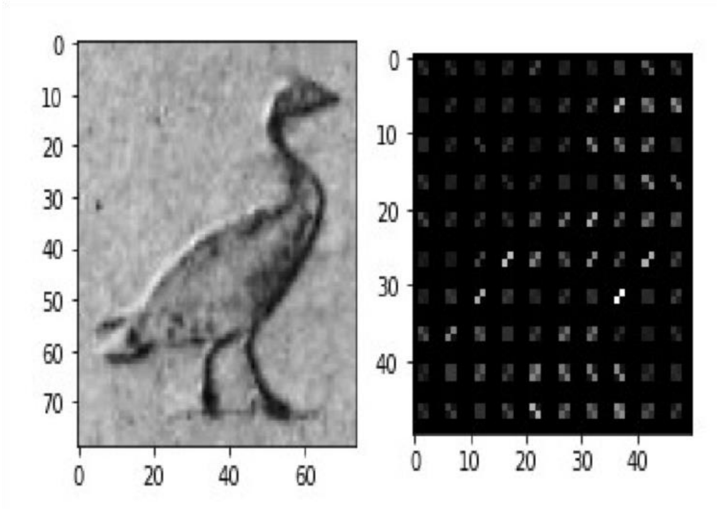


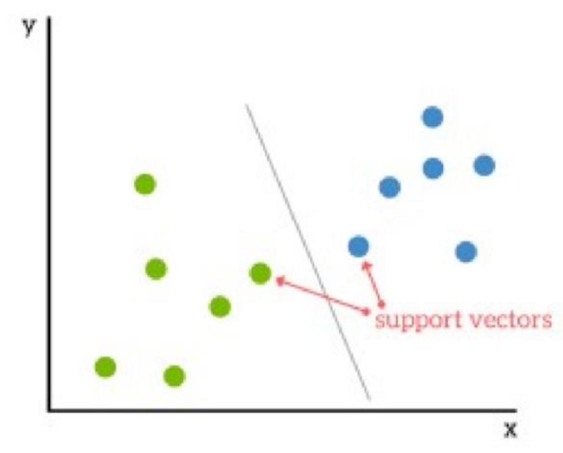
Figure x: General example for HOG

We divide the image into cells, and we calculate the histogram of the orientations Φ. We divide the orientation 0°→360° into N bins, and store the frequency of each orientation range into its corresponding bin.

SVM

1) Concept

Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification and regression purposes. SVMs are more commonly used in classification problems. SVMs are based on the idea of finding a hyperplane that best divides a dataset into two classes. Support vectors are the data points closest to the hyperplane. We used SVM to train an offline supervise model.



One of SVM advantages is accuracy but it’s not suited for a large –tens of thousands– datasets as its training time can be high. It can be efficient as it uses a subset of training points. It is used with HOG to detect car plates, car logos and pedestrians.

2) Sklearn SVC

Sklearn’s SVC class is a C-support vector classifier which implements the “one-versus-one” approach for multi-class classification. In total, (N(N−1))/2 classifiers –where N is the number of classes– are constructed and each one trains data from two classes.

Sklearn provides many out of the box kernels like linear, polynomial, sigmoid, ...etc out of which we choose the RBF kernel also called a Gaussian function.

The RBF kernel equation

$$K_{rbf} = e^{-\gamma \|x-\bar{x}\|^2}, \gamma = \frac{1}{(N_{features} * var(X))}$$

Natural Language Processing

Natural language processing (NLP) is the intersection of computer science, linguistics and machine learning. The field focuses on communication between computers and humans in natural language and NLP is all about making computers understand and generate human language. Applications of NLP techniques include voice assistants like Amazon's Alexa and Apple's Siri, but also things like machine translation and text-filtering.

NLP has heavily benefited from recent advances in machine learning, especially from deep learning techniques. The field is divided into the three parts:

- Speech Recognition: The translation of spoken language into text.
- Natural Language Understanding: The computer's ability to understand what we say.
- Natural Language Generation: The generation of natural language by a computer.

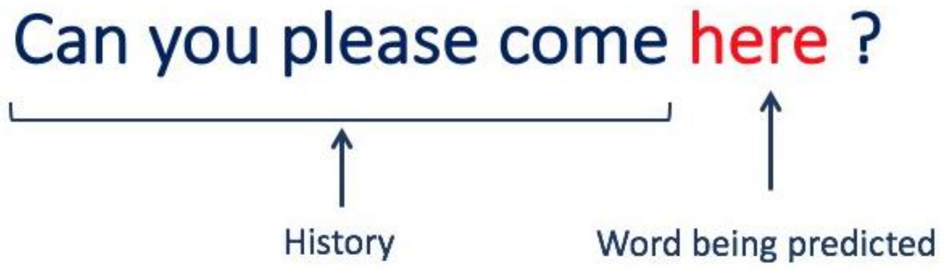
Why NLP is difficult

Understanding human language is considered a difficult task due to its complexity. For example, there is an infinite number of different ways to arrange words in a sentence. Also, words can have several meanings and contextual information is necessary to correctly interpret sentences. Every language is more or less unique and ambiguous. Just take a look at the following newspaper headline "The Pope’s baby steps on gays." This sentence clearly has two very different interpretations, which is a pretty good example of the challenges in NLP.

N-Gram

An N-gram language model predicts the probability of a given N-gram within any sequence of words in the language. If we have a good N-gram model, we can predict p(w | h) – what is the probability of seeing the word w given a history of previous words h – where the history contains n-1 words. We must estimate this probability to construct an N-gram model.

How do N-gram Language Models work?



We compute this probability in two steps:

1) Apply the chain rule of probability
2) We then apply a very strong simplification assumption to allow us to compute p(w1…ws) in an easy manner

The chain rule of probability

It tells us how to compute the joint probability of a sequence by using the conditional probability of a word given previous words.

Chapter Three

System Architecture

Flutter phase

Before talking about Flutter we developed a prototype using android to simulate the project on then we upgrade our code by using Flutter framework with dart programming language.

Android App Prototype

The android application is presenting the user interface that the user will use to do the translation by his phone.

The android app is divided into two main parts which are:

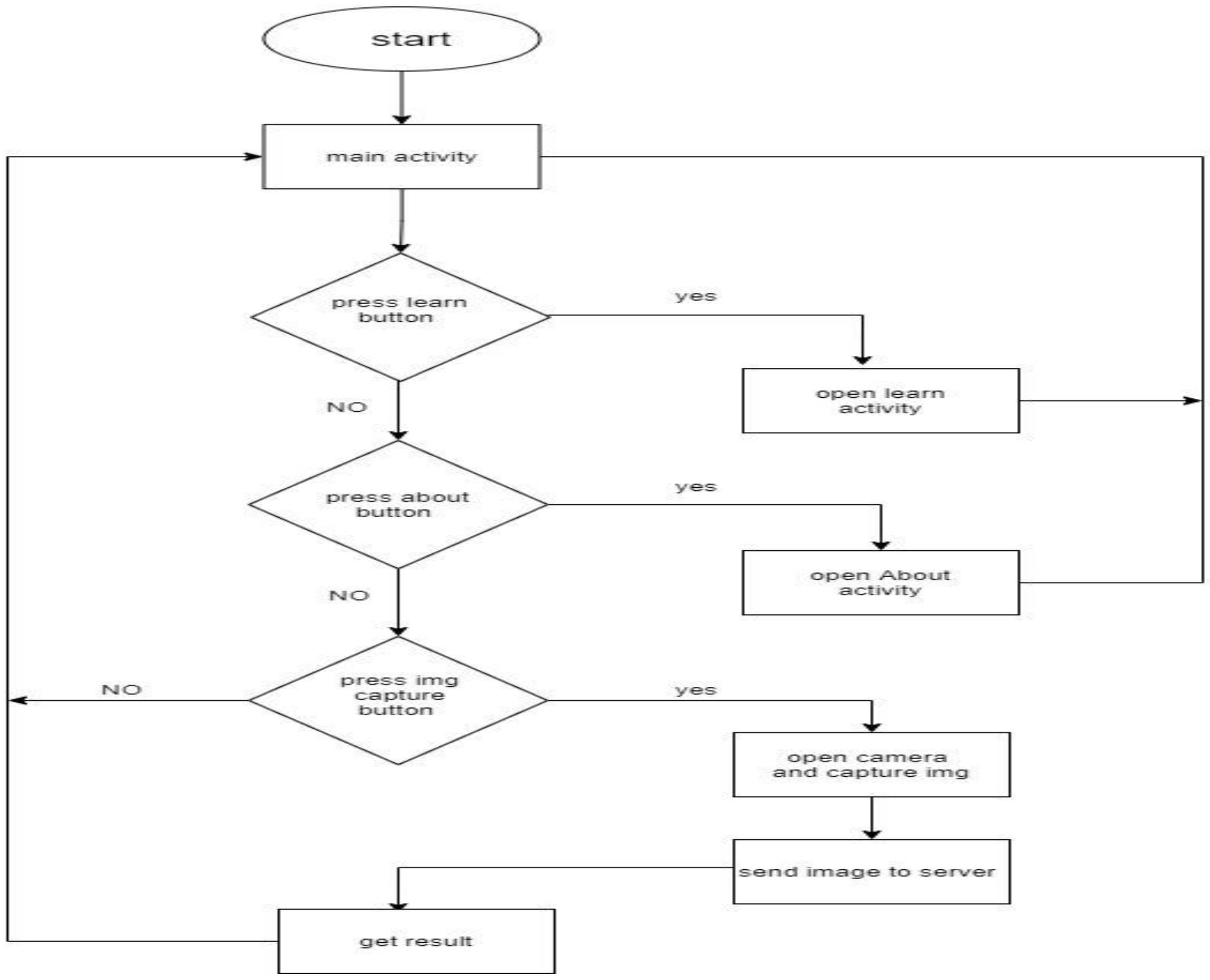
- FRONT END: it represents the GUI of the application
- BACK END: it represents the function and classes that's work in background

Android process

1) Open the camera by press the capture image button
2) Sent the image to server with new name and unique id
3) The machine learning code doing the process of translation of Hieroglyphic letters in input image to English statement and return text
4) The android get the returned text to show it to user.
5) Go back for new capture or main page of design

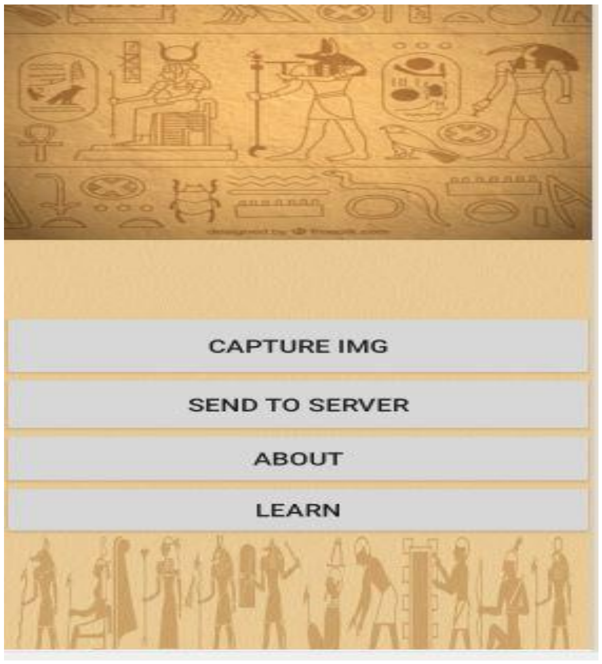
Flow Chart

This flow chart describe this app flow and some details



Project design

The GUI has about 4 buttons



- Capture the image

This button is used to access the camera of phone and capture the image.

- Send to server (translation)

This button is used to send the image to server to make processes on it to get the result.

- About

This button is used to open new page in app to known some information about the application

- Learn

This button is used to learn about the Hieroglyphic characters and symbols.

Stack holder

- User

Who can interact with the test cases of the system.

1) Capture image and Send it
2) Open learn activity
3) Open about activity
4) Get result

- Server

Which can receive the image for storing in data base and sent the result to the user.

Flutter

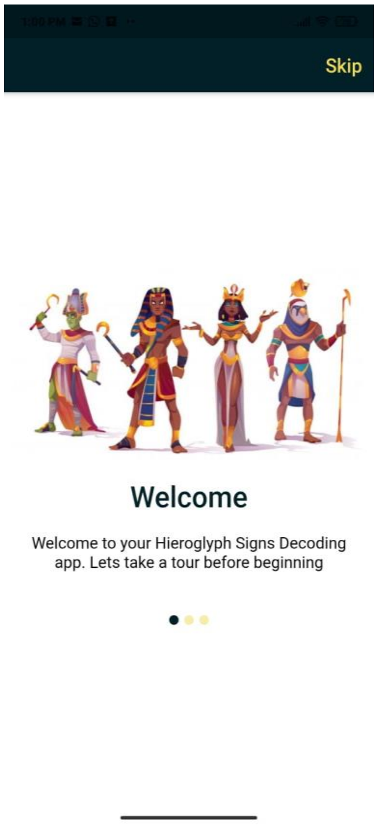
Flutter is a free and open-source mobile UI framework created by Google. In a few words, it allows you to create a native mobile application with only one codebase. This means that you can use one programming language and one codebase to create two different apps (for IOS and Android). Flutter combines ease of development with performance similar to native performance while maintaining visual consistency between platforms.

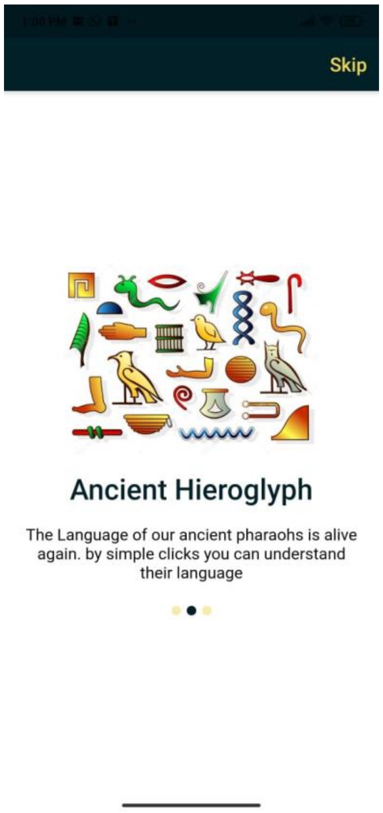
Flutter consists of two important parts:

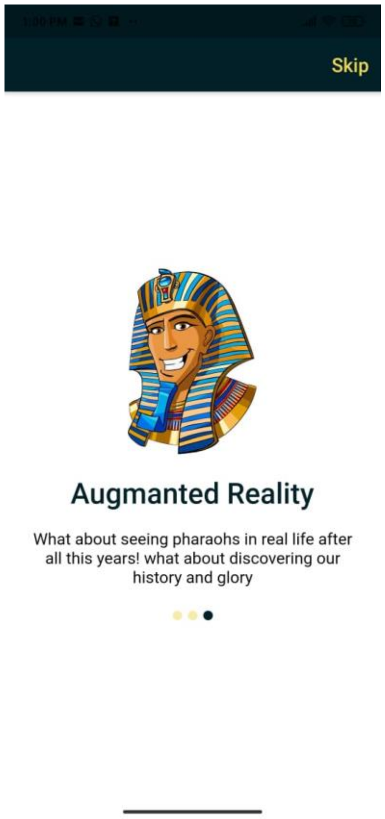
1) An SDK (Software Development Kit): A collection of tools that are going to help you develop your applications. This includes tools to compile your code into native machine code (code for iOS and Android).
2) A Framework (UI Library based on widgets): A collection of reusable UI elements (buttons, text inputs, sliders, and so on) that you can personalize for your own needs.

Here some GUI screenshots from our app:

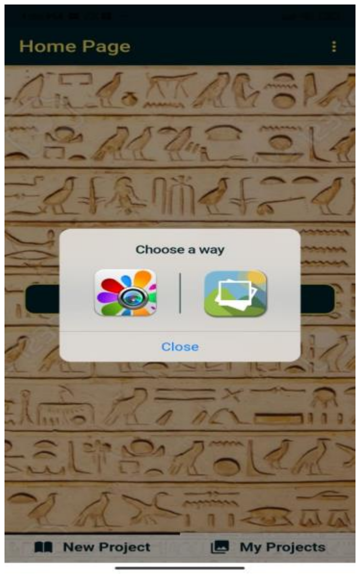
1) First a welcome screen


Skip
Welcome
Welcome to your Hieroglyph Signs Decoding app. Lets take a tour before beginning

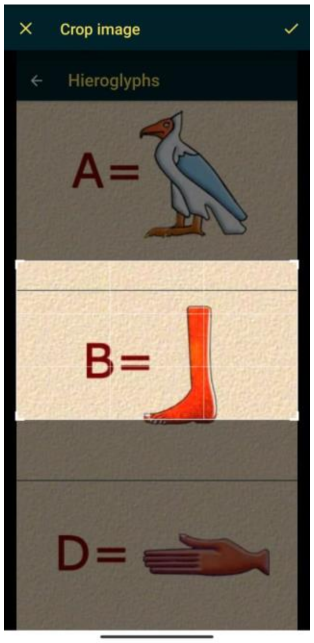

Skip
Ancient Hieroglyph
The Language of our ancient pharaohs is alive again. by simple clicks you can understand their language


Skip
Augmanted Reality
What about seeing pharaohs in real life after all this years! what about discovering our history and glory

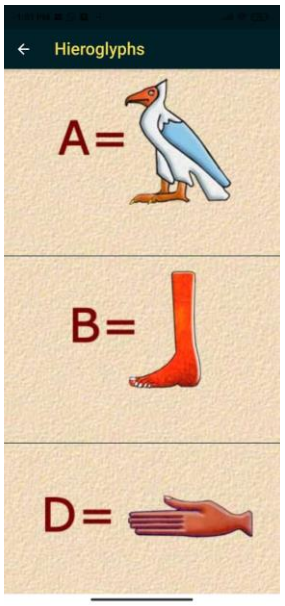
2) Input image can be entered by two methods either take a photo and choose one in gallery



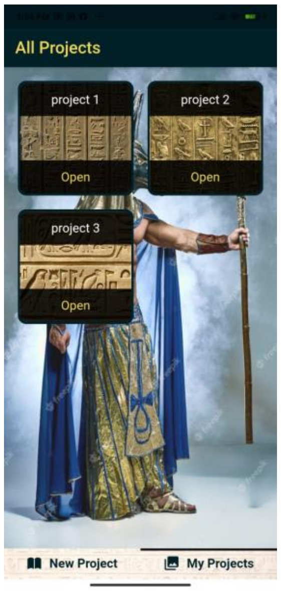
3) You can use crop tool



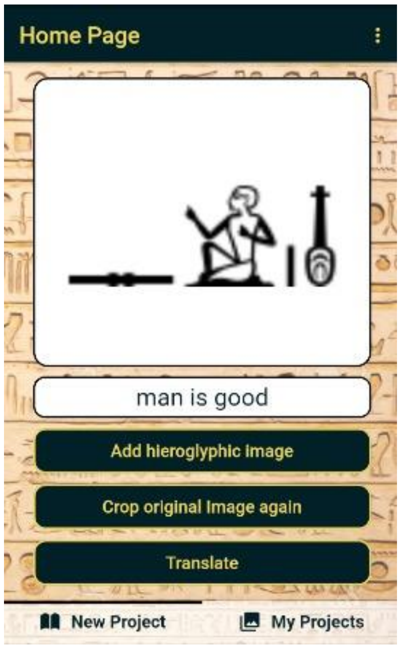
4) You can also learn more about Hieroglyphic letters



5) You can see the previous image translations used as we store them in our database



6) The result screen



Purpose and Goals

We used Flutter in our application to increase our product quality and performance and to be able to run our application on android and IOS. Both platforms android and IOS are quite mature at this stage, having existed for more than a decade. That means both have amassed comprehensive feature sets, and there's very little one can do that the other cannot. Still, however, each has its advantages, and there are reasons you might want to choose one over the other. So our app works on both for more users usage and audience expansion.

We also implement Unity game engine Augmented Reality (AR) in Flutter widget. Now we can present awesome AR features of our app in Unity and get it rendered in a Flutter app both in embeddable mode. Works great on Android, iPad OS and IOS.

Our Flutter app has many features like translating Hieroglyphic sentences to English by taking a shot from mobile camera or from a photo stored in gallery. Another feature is translating Hieroglyphic sentences from Gardiner's code to English statement.

You also can learn more about Hieroglyphic letters and its mapping in English. You can also know the Hieroglyphic letters of your name and see the exact picture of each glyph that are used in ancient Egypt writing system.

The framework of Flutter, written in the Dart programming language, has the Flutter engine, Foundation library, and widgets. Most importantly, Flutter is open-source and completely free. The approach to development in Flutter differs from others by its declarative UI writing. Here, there is a need to start from the end, meaning before starting the development of some element, the user needs to have in mind a complete picture of what kind of UI it will be. Many developers distinguish this UI writing as a more clear one, but it also causes certain difficulties for developers at first.

Dart

Dart is a client-optimized language for developing fast apps on any platform. Its goal is to offer the most productive programming language for multi-platform development, paired with a flexible execution runtime platform for app frameworks. Flutter’s programming language, Dart, was initially intended as a replacement for JavaScript.

Dart’s compiler technology lets you run code in different ways:

1) Native platform: For apps targeting mobile and desktop devices, Dart includes both a Dart VM with just-in-time (JIT) compilation and an ahead-of-time (AOT) compiler for producing machine code.
2) Web platform: For apps targeting the web, Dart includes both a development time compiler (dartdevc) and a production time compiler (dart2js). Both compilers translate Dart into JavaScript.

Flutter Advantages

1) Same UI and Business Logic in All Platforms
2) Reduced Code Development Time
3) Increased Time-to-Market Speed
4) Similar to Native App Performance
5) Custom, Animated UI of Any Complexity Available
6) Own Rendering Engine

Django Server

We would like first to work on a PHP server that act as a prototype to test on then we will build the final project with Django server to enable the communication between the Flutter app and Machine Learning code.

Coding processes

1) Capturing an image
2) Sending image parameters (name, date, time) to web service (PHP)
3) PHP file will have (image parameters , server content)
4) Then sending PHP file to the server & applying it to our database

Capturing image

First we define our permissions :- Like (camera , internet , wifi , read external storage , write external storage)

After opening the camera and captures the image , We get the image file and put it in variable ; Then we construct the image itself from that variable and put it in another variable to make processing on it

To improve performance we will convert that image to string form By using (imagetostring) function ; which allow us to specify and control the image quality

Then collecting some parameters like:-

1) image as string
2) name = android-id + current-date + current-time ((using android-id to make sure that we have a unique parameter))
3) Operation-name
4) current-date

To send those parameters to the web service:

We using VolleySingelton Class as an object to send request (with parameters) To our web service.

Web Services

We using PHP as Web Service

After sending the parameters of image, PHP holds all information about image & about server information like:-

A) PHP receives picture (parameters) as keys

- Image as string with key (img)

- Name of picture with key (in)

- Current-date with key (da)

- Operation-name with key (opn)

B) PHP holds server information:-

Like (server name , username , password , name of database) & the connection method

PHP processes

A) Create a folder with operation date if it does not exist

B) And gives the SQL query which refer to our ((stored procedures on server))

C) Holds reaction responses after connecting to the server

We have two responses:-

A) On Response:

Means PHP successfully connected with server so we have two cases

1) Image was sent to server Response message ((image loaded successfully))

2) Image not send to server Response message ((please try later))

B) ErrorOnResponse:

Means PHP can't reach to the server so we have one response message ((can't reach server))

Server contents

On the server we have our database with (Username, Password)

The database which has three tables for storing the captured image:-

1) Operation name
2) Pic name
3) Result

We are using ((stored procedure feature)) to apply queries on our database

After sending image to server some processes are done on it by Machine Learning phases which return result as a text so by using JSON we get that result and show it to user.

Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django is ridiculously fast. Django was designed to help developers take applications from concept to completion as quickly as possible. Django takes security seriously and helps developers avoid many common security mistakes. Some of the busiest sites on the Web leverage Django's ability to quickly and flexibly scale.

Aim

Our usage to Django is to act as a server to run our ML and Deep Learning code on it. We make a communication between Django python server and Flutter to send the captured image from Flutter to server to be processed then return back to the Flutter app to be displayed to user.

We choose to work with Django server as most of our ML code is written in python so we want the server to directly communicate with our code and by connecting this server to our app in Flutter by sharing the same network we can open our app in android or IOS. Also using Django is useful as it is easy to extend and scale. We used this technique to minimize the overall processing time as any other server will complicate and increase communication time in our app.

Django REST framework

Django REST framework is a powerful and flexible toolkit for building Web APIs. REST is a loosely defined protocol for listing, creating, changing, and deleting data on your server over HTTP. The Django REST framework (DRF) is a toolkit built on top of the Django web framework that reduces the amount of code you need to write to create REST interfaces.

Advantages

1) Written in Python
2) Designed as a batteries-included web framework
3) Supports MVC programming paradigm
4) Compatible with major operating systems and databases
5) Provides robust security features
6) Easy to extend and scale

OCR (Optical Character Recognition)

The goal of this part is the detection of the characters present in the image, then trying to find their optimal reading order.

X.1.1. Hieroglyphics

Hieroglyph is an ideographic language like Cuneiform, Chinese and Japanese characters, each of their characters stand for a specific meaning that is understood by each culture. Most of the characters stand for the sounds within the spoken language, but most of them also have an ideographic meaning in the written language.

X.1.1.2. Hieroglyphics reading order:

Hieroglyphs are written in rows or columns and can be read from left to right or from right to left. You can distinguish the direction in which the text is to be read because the human or animal figures always face towards the beginning of the line. Also, the upper symbols are read before the lower.

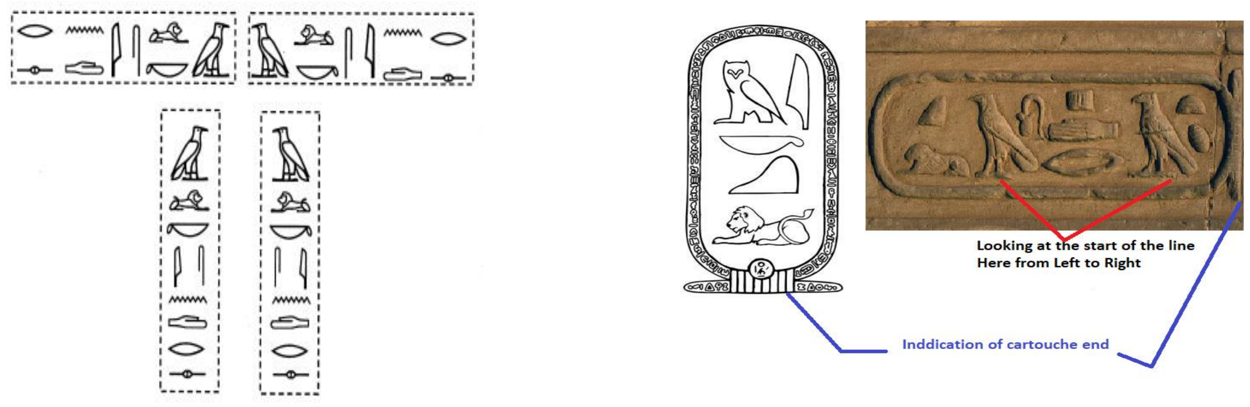

Looking at the start of the line
Here from Left to Right
Inddication of cartouche end

Figure x: Hieroglyphics reading order.

X.2. Used Approach

Since Hieroglyph is considered as a dead language there is no available library to detect Hieroglyphics, while for other popular languages there are a lot of libraries like Tesseract and EasyOCR.

Also, Hieroglyph have more than 1000 distinct characters and a few available texts, and as mentioned before detecting reading order is a tricky task, which make the approach we use the best approach.

X.2.1. Approach (Module) Brief:

First, the desired image to be translated and Its reading direction are passed to the module.

If the reading direction is not from right to left, we process it to be from right to left.

Then we apply grey effect and blur filter.

Then we apply canny edge detection algorithm to our image, after that we apply dilation to increase the thickness of the detected edges.

Then we check if the image has any lines, if it has lines, we use the “Get Contours” function,

If it does not have lines, we use “Get Contours no Lines” function, both of those function return the sorted detected contours and their reading order.

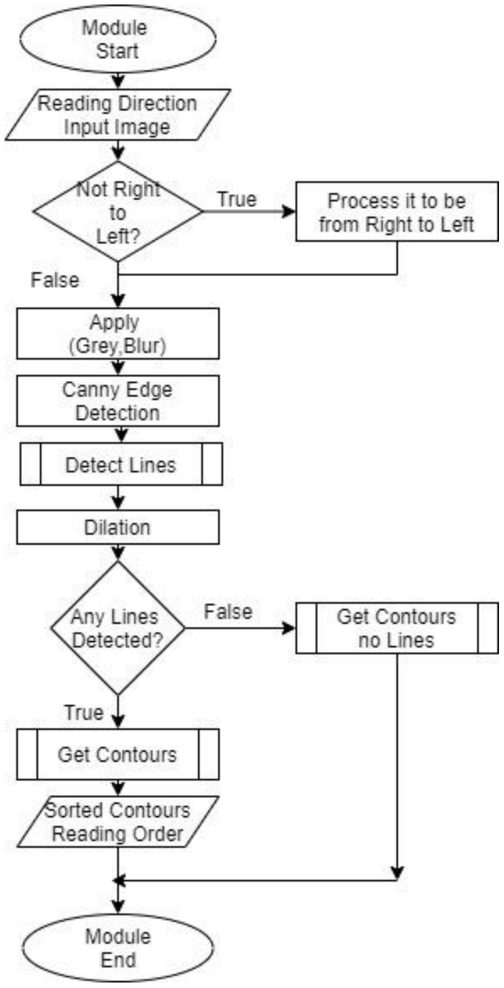


Figure x: OCR Module Flowchart

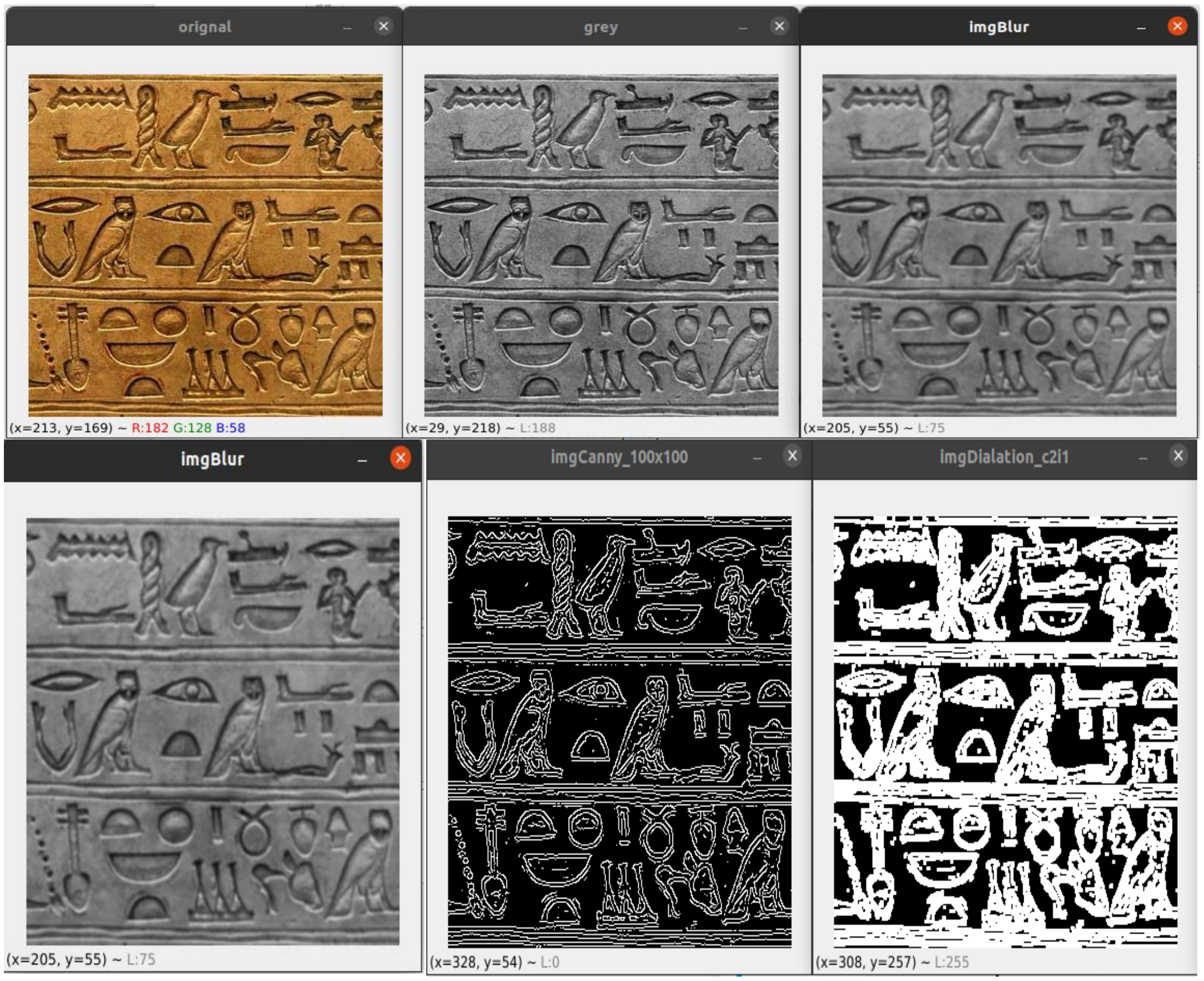


Figure x: (original, grey, blurred, post canny, dilated) Images

X.2.1.1. Reading Direction:

The desired image to be translated is passed to the module, according to Wikipedia 96% of hieroglyphics are horizontal and from right so by default the reading direction is set to right to left, and for some future work the image may be not from right to left, or it may be rotated or so on, so the module may rerun again but it will be given the correct reading direction so the module will process the image.

X.2.1.2. Grey and Blur

First, we convert the image to grayscale since grayscale images are entirely sufficient for our tasks and so there is no need to use more complicated and harder-to-process color image.

Then we apply a blur filter to smooth edges and removes noise from an image (Gaussian Noise Reduction)

X.2.1.3. Canny Edge Detection

Canny edge detection is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed, and it’s a must to use edge detection technique before using any detecting contour function.

There are other edge detection techniques like Sobel, Prewitt and Laplacian, but canny give the best results here so we go with it.

X.2.1.4. Detect Lines

Detecting lines will be helpful for determining the reading order of the characters, since it can indicate whether the writing direction is vertical or horizontal, and also lines help to divide the image into separate portions (reading line).

X.2.1.5. Dilation

Edge detection algorithms will produce a thin line that may make a problem when passed to the cv2.findContours function, that not all the contours got detected; so, to overcome this problem we can increase the thickness of the edges by using dilation.

But in some cases, dilation may cause the detection of multiple close neighbored characters as one character; so, we better choose a suitable kernel.

X.2.1.6. Get Contours:

We use the function Find Contours from OpenCV Library, it will output an array of contours, we will iterate on this loop and we will exclude contours with area smaller than a threshold, then we will pass the contour to Crouches Detection function to detect whether the contour is a cartouche or not, if it's a cartouche it will return an array of contained contours and their reading order, if the contour is not a cartouche, we will get its reading order with respect to lines.

We will append all the detected contours to an array with their reading order, then we will sort them according to their reading order and output the array so the next module can use

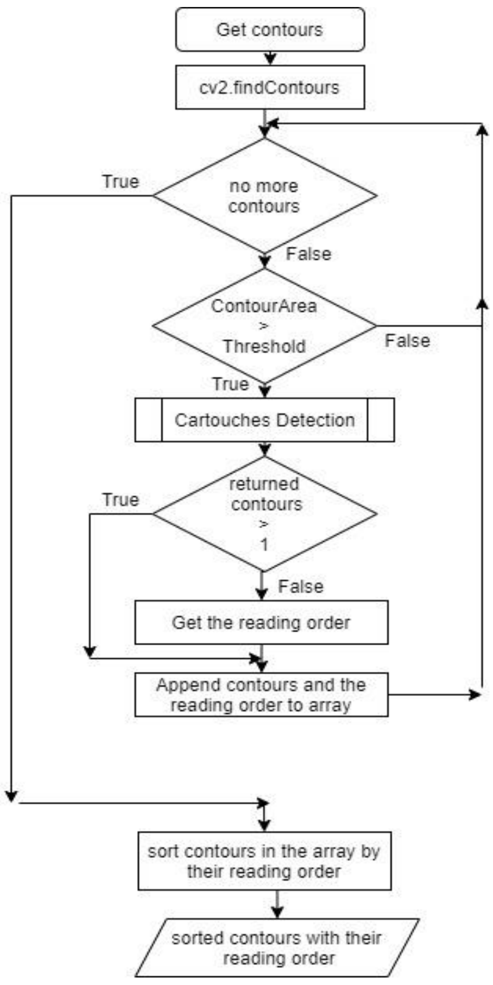


Figure x: Get Contours Function Flowchart

X.2.1.6.1. Crouches Detection:

It's a function that work with the same manner as Get Contours function but it will crop the frame of the contour then apply the Find Contours function again if we detect more than a contour, we will get their reading order according to the cartouche and the whole image.

X.2.1.7. Get Contours No Lines:

It works with the same manner as get contours function but in this function the image passed has no lines so it's either an image that contains only a single line of text, or maybe it has several lines of text but without a separating line; which makes the detection of the optimal reading order harder than if there is separating lines.

So, we will use Detect Spaces function to detect and stretch spaces do get the optimal reading order.

X.2.1.7.1. Detect Spaces:

First, we create a horizontal or vertical shaped kernel and dilate to connect the words of each line into a single contour.

Then, we extract each line contour and we sort them, and extract each line ROI

After that, we can append white space between each line contour, to increase the spaces between each line to increase the accuracy and ease the detection afterwards.

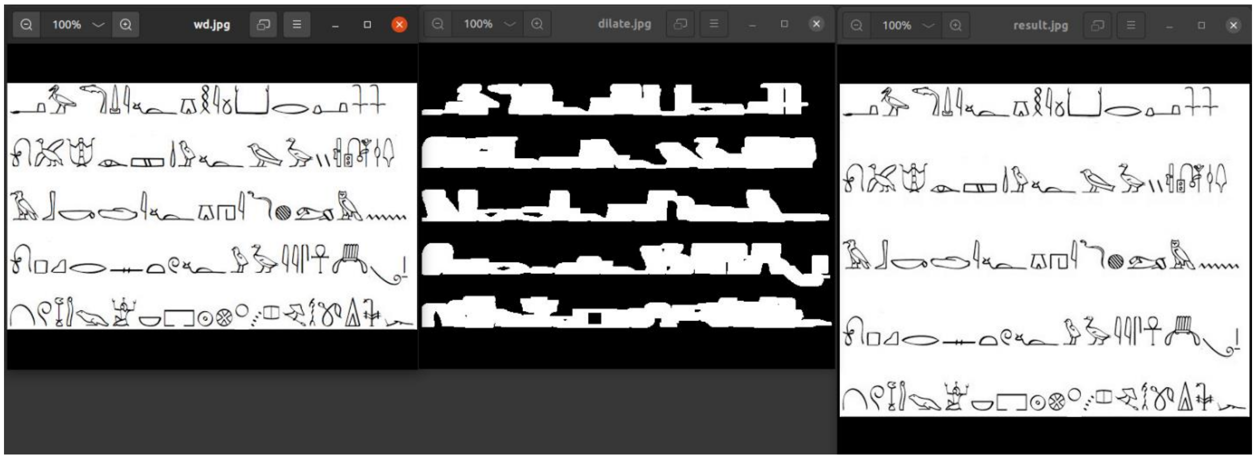


Figure x: Image after (Dilation with horizontal kernel and appending spaces between lines)

Classification phase

1) Overview

In Order to do the process of classification we need to extract features from the dataset and feed the features to a model and then deploy the model to production. Both feature extraction and model training are time/CPU intensive. A solution to this problem is to separate each process and provide the output in a reusable format.

This is the solution we went with to solve the problem by building different software components to abstract each part of the process.

2) Lazy Object

The idea of a lazy object is inspired by Haskell’s lazy evaluation where an expression isn’t evaluated until needed. And so we tried to emulate the concept to provide a better overall performance by doing a process when needed and when a process is done is serialized into a binary format to be reused when needed again.

All the three components implements different aspects of the lazy object. For example the image component loads an image from storage only when the image is need to be processed once the image is no longer needed the memory space is freed.

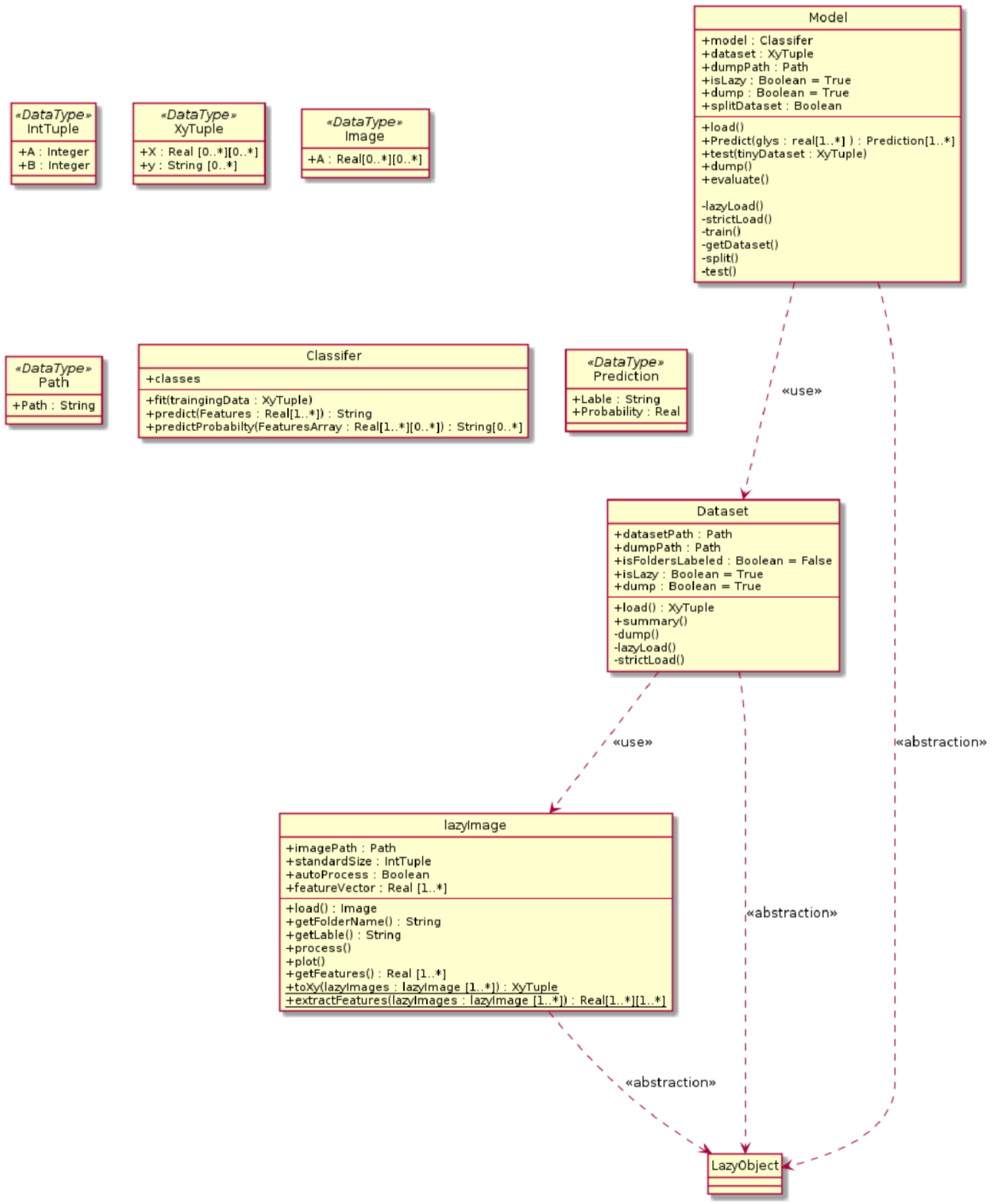


Figure 6: shows a class diagram for the classification phase components

3) Image component

The function that the image component serves is to abstracts the glyphs images in the dataset. It does so by loading the image itself when need to be processed and applying the needed operations on it, freeing the space the image takes.

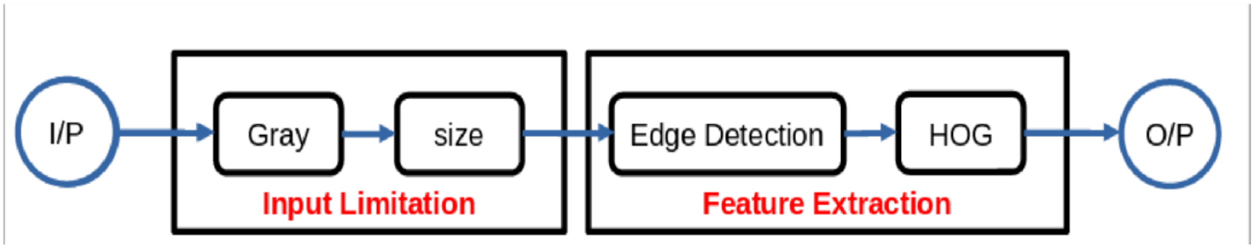


Figure 7: the main operations done on an image in the image object

The Image object operation:

• Loads an image from storage –it can also accept other type of images like numpy arrays, ...etc–.

• Changes image to gray scale.

• Resizes –resample– the image to a standard size.

• Applies Sobel edge detection.

• Applies HOG feature descriptor.

• Keeps the final features vector and free the memory of the original image.

• Gets the image predefined –label if existed–.

1) Gray scale

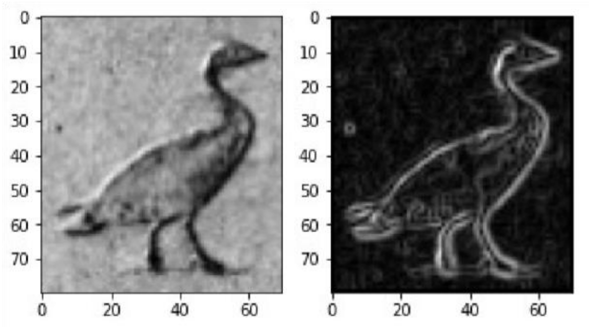
RGB to gray scale conversion equation:

$$L = \frac{1}{1000}\begin{bmatrix} R & G & B \end{bmatrix}\begin{bmatrix} 299 \\ 587 \\ 114 \end{bmatrix}$$

To turn the image from a colored image to a gray scale image we apply the formula to each pixel in the image; turning the three color channels into a one gray channel 8-bit image.

2) Edge detection

1) Sobel edge detection

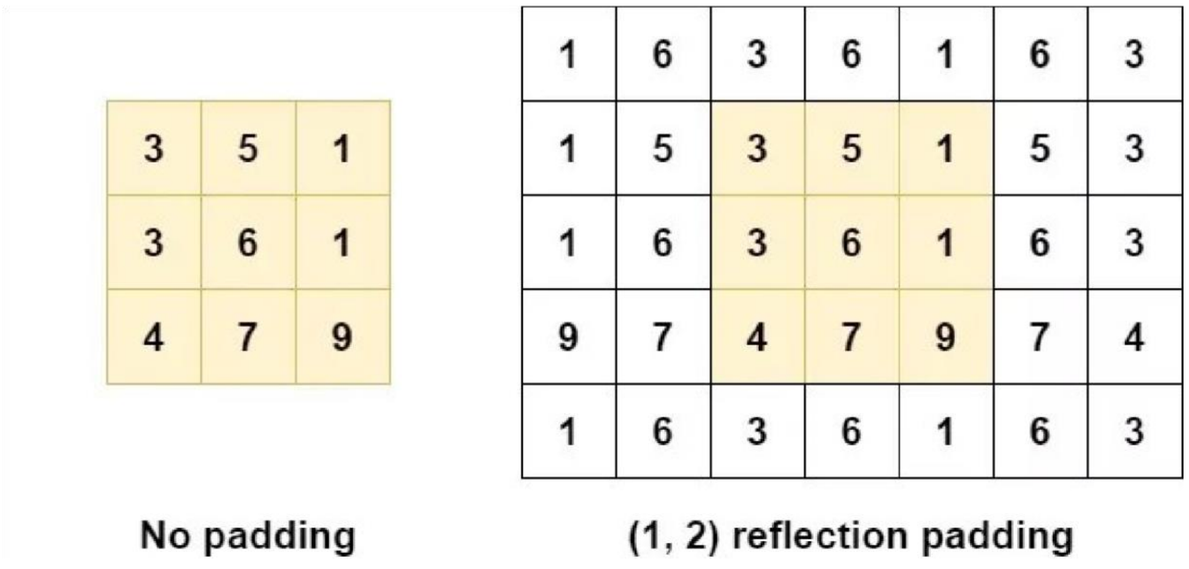


We apply Sobel operators on the image by means of convolution using reflection padding.

Sobel operators

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}, G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

In reflection padding we reflect the row into the padding.



3	5	1
3	6	1
4	7	9

No padding

1	6	3	6	1	6	3
1	5	3	5	1	5	3
1	6	3	6	1	6	3
9	7	4	7	9	7	4
1	6	3	6	1	6	3

(1, 2) reflection padding

Figure x: example of reflection padding.

We used skimage’s implementation of Sobel edge detection.

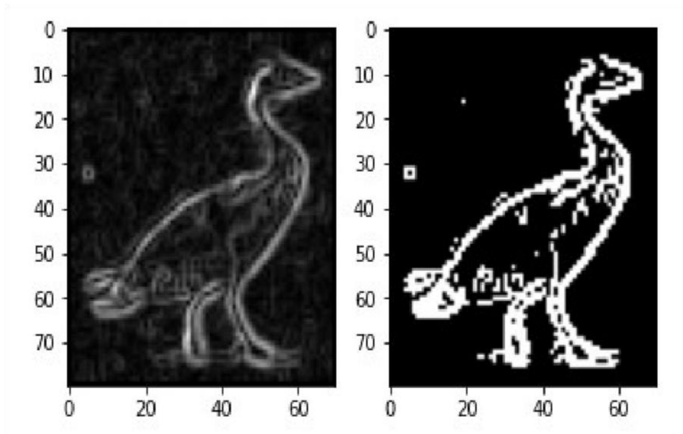
2) Otsu Thresholding

Then we apply Otsu thresholding –an adaptive thresholding for image binarization– to remove some background noise. In Otsu method we aim to minimize the within class variance (σW). In order to do so we maximize the between class variance (σB) instead because the overall variance (σ) is a constant and it is easier to calculate iteratively for t= 0 → 255 and choose the threshold that results in the maximum value of between class variance (σB).

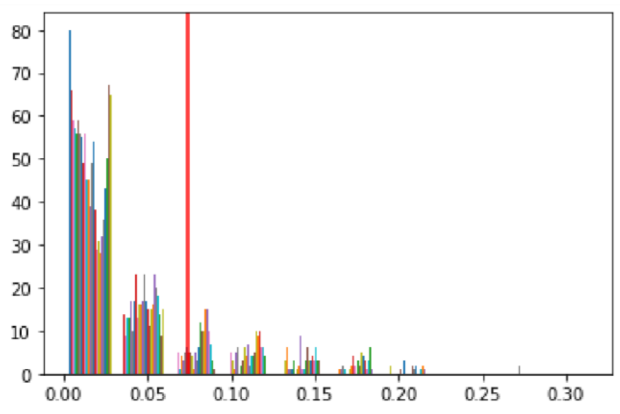
$$\sigma^2 = \sigma_w^2 + \sigma_B^2$$
$$\sigma_B^2 = \sigma^2 - \sigma_W^2$$
$$\sigma_B^2 = W_f W_b (\mu_b - \mu_f)^2$$
$$W_b(t) = \sum_{i=0}^{t} P(i), W_f(t) = \sum_{i=t+1}^{I} P(i) = 1 - W_b$$
$$\mu_b(t) = \frac{1}{W_b(t)} \sum_{i=0}^{t} i.P(i), \mu_f(t) = \frac{1}{W_f(t)} \sum_{i=t+1}^{I} i.P(i)$$

Figure 12: Otsu thresholding equations

To calculate σB for a given threshold t, we calculate the background weight Wb which equal the sum of probabilities of intensities in the background, then Calculate foreground weight Wf which equal 1-Wb then we calculate the mean of background μb and foreground μf. We used Skimage’s Otsu thresholding.



example for an image before and after Otsu thresholding.



Example for applying Otsu threshold on a histogram.

3) HOG

In the HOG phase we used skimage’s implementation of HOG descriptor we divided the image into 10×10 cells with 10×10 block with 8 bins orientation to reduce the feature vector size to reduce the time of prediction of an image.

Feature vector size can be calculated using the given equation

$$\Gamma=\lambda . B_x . B_y . (\Delta_x+1) . (\Delta_y+1), where\, \Delta_n=P_n-B_n$$

where:

- Γ is the length of the flattened feature vector.
- ƛ is the number of bins.
- Bx is the number of cells in a block in the x-direction.
- Px is the number of cells in an image in the x-direction.

In our case the numbers in x and y direction are the same:

$$\lambda=8, B=P=10, \Delta=0, then\, \Gamma=800$$

4) Finding image label (if existed)

There are different methods to organize dataset's data labeling most common way to organize a dataset is adding all images that have the same label in a directory that has the same name as its label. The dataset we mainly worked with followed a different approach by naming the image a name that consisted of 5-digit number, underscore and the image label – Gardiner code– for example : "090429_M17.png".

To extract the label from such a dataset we used regular expression and error handling to extract the label, we also supported the more common method mentioned before in case of failure.

Example for the regular expression used : ".*_(.*)\..*"

Finally it should be mentioned that each part of the image module operation can be customized –in case of experimenting or developing a different approach to classification– and changed by means of inheritance which will be explained in next parts.

3) Dataset component

The dataset software component accepts:

• A path or list of paths for a dataset's directory.

• The type of dataset labeling method –discussed previous part–.

• A path to keep a binary version of the computed features.

• A configured image object –an image object or descendant of the image class–.

When it's needed to read the data in the underlying dataset the 'load()' method can be called and the dataset object starts scanning a hierarchy of directories –provided in the path(s)– applies the image object on each one and ignores non-image files.

The dataset component also provide some methods to manipulate a features- labels tuple. For example:

• ‘add’ : overloads the ‘+’ operator to allow concatenation of two dataset objects.

• ‘drop’ : drops specified labels from a features-labels tuple.

• ‘head’ : to select first n elements of features-labels tuple.

It should be mentioned that all paths is handled by python’s pathlib’s Path object to provide a path portability over different operating systems –for eg. Linux, Windows, …etc –.

4) Model Component

The model component abstracts the underlying model –in our case Sklearn's SVC– , it accepts:

• A model object.

• A dataset object or a features-labels tuple.

• A path to keep a binary copy of the model once it’s trained.

When the model is needed, a ‘load’ method can be called to start model training.

5) Top view

Main use-cases for the three mentioned components are :

• Feature selection

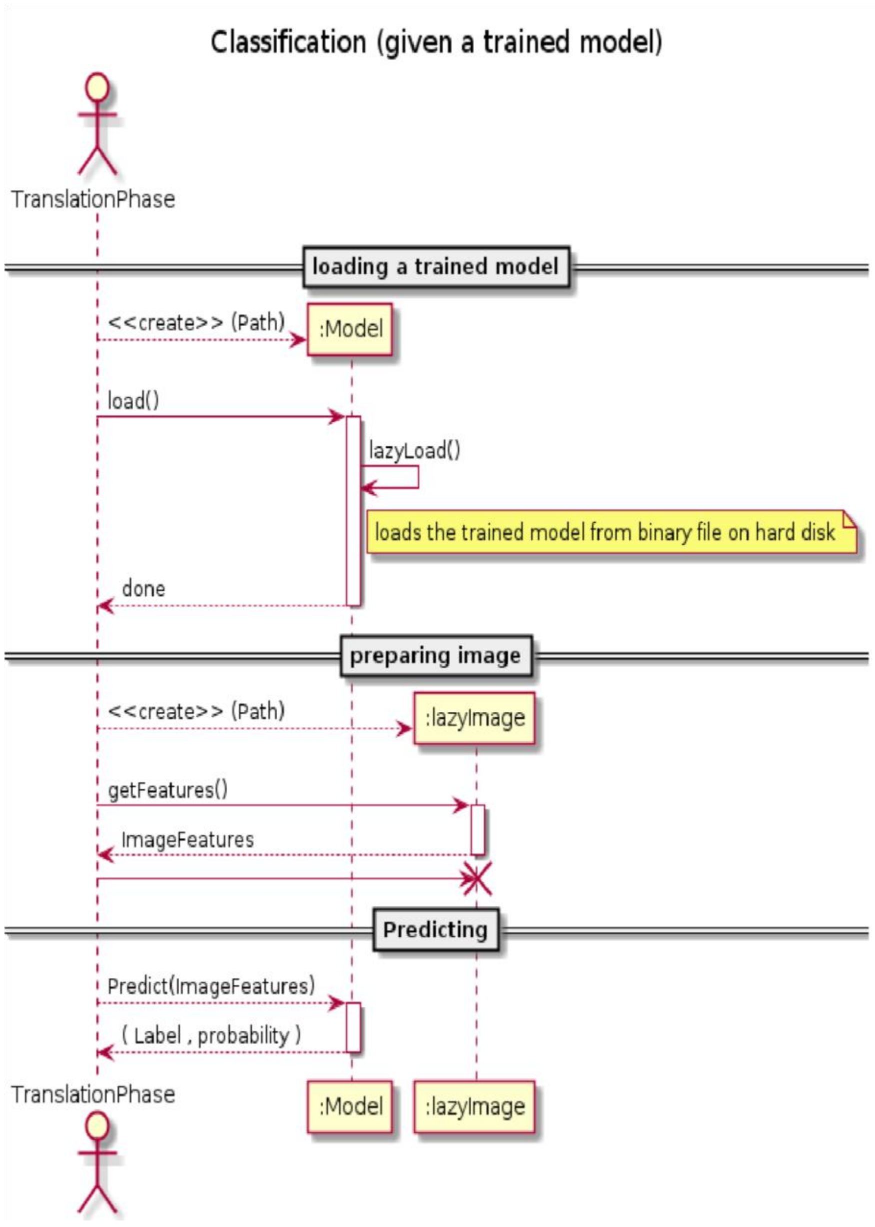
• Training/Developing a model.

• Use of model in production.

In feature selection use- case we can experiment until features are chosen then they can be implemented in a descendant of image object that overrides the needed methods to achieve the needed effect. Then this configured image object can be passed to a dataset object to compile the dataset to binary file that can be used in training a model.

In training use-case a model can be trained with a given compiled dataset and a compiled version of the model will be stored so it can be reused.

In production use-case the binary file version of the model and a configured image object –which used to extract the features– can be provided to the production server to run the model.



Deep Learning

Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

Features

- Deep learning is an AI function that mimics the workings of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions.
- Deep learning AI is able to learn without human supervision, drawing from data that is both unstructured and unlabeled.
- Deep learning, a form of machine learning, can be used to help detect fraud or money laundering, among other functions.

We used Deep Learning to increase the accuracy of our predictions as ML model SVM is not giving us the high accuracy numbers we wish to work with. We use One Shot Learning with Siamese Networks using Keras.

Deep Convolutional Neural Networks have become the state of the art methods for image classification tasks. However, one of the biggest limitations is they require a lots of labelled data. In many applications, collecting this much data is sometimes not feasible. One Shot Learning aims to solve this problem.

Classification vs One Shot Learning

In case of standard classification, the input image is fed into a series of layers, and finally at the output we generate a probability distribution over all the classes (typically using a Softmax). For example, if we are trying to classify an image as cat or dog or horse or elephant, then for every input image, we generate 4 probabilities, indicating the probability of the image belonging to each of the 4 classes.

Two important points must be noticed here. First, during the training process, we require a large number of images for each of the class (cats, dogs, horses and elephants). Second, if the network is trained only on the above 4 classes of images, then we cannot expect to test it on any other class, example “zebra”. If we want our model to classify the images of zebra as well, then we need to first get a lot of zebra images and then we must re-train the model again. There are applications wherein we neither have enough data for each class and the total number classes is huge as well as dynamically changing. Thus, the cost of data collection and periodical re-training is too high.

On the other hand, in a one shot classification, we require only one training example for each class. Yes you got that right, just one. Hence the name One Shot. Assume that we want to build face recognition system for a small organization with only 10 employees. Instead of directly classifying an input (test) image to one of the 10 people in the organization, this network instead takes an extra reference image of the person as input and will produce a similarity score denoting the chances that the two input images belong to the same person. Typically the similarity score is squished between 0 and 1 using a sigmoid function; wherein 0 denotes no similarity and 1 denotes full similarity. Any number between 0 and 1 is interpreted accordingly.

Notice that this network is not learning to classify an image directly to any of the output classes. Rather, it is learning a similarity function, which takes two images as input and expresses how similar they are

In a short while we will see that to train this network, you do not require too many instances of a class and only few are enough to build a good model. This advantage is perfect for our application as we don’t have very high number of data to train a normal model and similarity difference is exactly what we need to know which Hieroglyphs are in the input image.

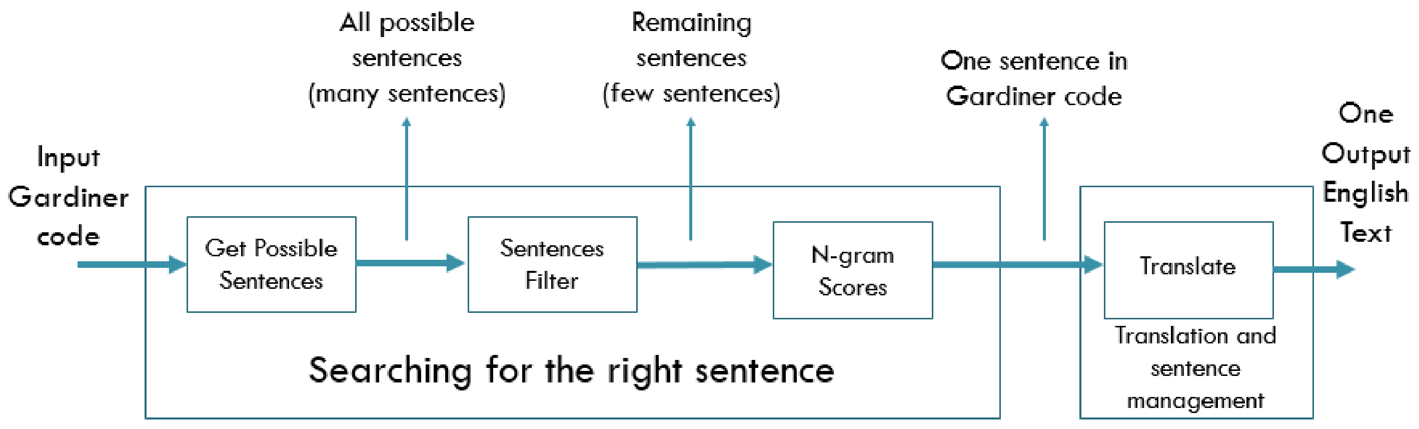
Triplet Loss

The idea of comparative loss can be further extended from two examples to three, called triplet loss. Triplet loss was introduced by Florian Schroff. Rather than calculating loss based on two examples, triplet loss involves an anchor example and one positive or matching example (same class) and one negative or non-matching example (differing class).

The loss function penalizes the model such that the distance between the matching examples is reduced and the distance between the non-matching examples is increased.

Translation

It is really powerful and unique tool that can be used separately as an API to translate any Hieroglyph text to English language. By using our translation tool you can get translation of any number of words in Hieroglyph not only one word as in dictionary so you can translate a paragraph so our tool is really wonderful and helpful.


All possible sentences (many sentences)
Remaining sentences (few sentences)
One sentence in Gardiner code
One Output English Text
Input Gardiner code
Get Possible Sentences
Sentences Filter
N-gram Scores
Translate
Translation and sentence management
Searching for the right sentence

Gardiner's Code

Egyptian hieroglyphs were the formal writing system used in Ancient Egypt. Hieroglyphs combined logographic, syllabic and alphabetic elements. Hieroglyphs may have emerged from the preliterate artistic traditions of Egypt. Egyptian hieroglyphs have a total of 1,000 distinct characters.

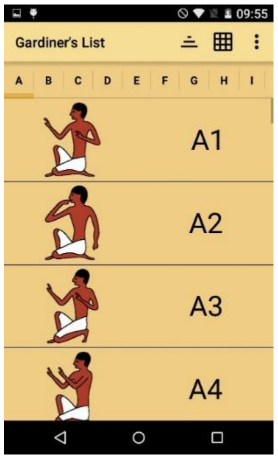
Subsets

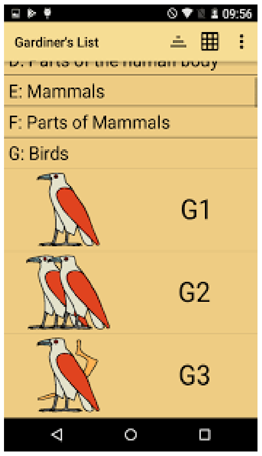
Notable subsets of hieroglyphs:

1) Deterministic

2) Uniliteral signs
3) Biliteral signs
4) Triliteral signs
5) Egyptian numerals

Gardiner's Sign List is a list of common Egyptian hieroglyphs compiled by Sir Alan Gardiner. It is considered a standard reference in the study of ancient Egyptian hieroglyphs. It describes 763 signs in 26 categories (A–Z, roughly). Gardiner lists only the common forms of Egyptian hieroglyphs, but he includes extensive subcategories, and also both vertical and horizontal forms for many hieroglyphs. He includes size-variation forms to aid with the reading of hieroglyphs in running blocks of text. So by using Hieroglyph sentence written in Gardiner's code as an input we can process this code and translate the sentence to English letters.





Transformation from Gardiner's code to English

There are many steps to be done before we can translate the Hieroglyphic language to English language and these steps manipulate the sentence to be ready to translation and there are many differences in the structure of the sentence in the two languages. As in many ancient writing systems, words are not separated by blanks or punctuation marks, there is no full-stop (.) between sentences or even a space,

Probably in all languages, a word does not necessarily have one meaning but in Hieroglyph the word may have 15 meaning or more as it is a language with symbolic letters that describe a natural elements in life so multiple meaning of the same word is widespread. Tour guides translation is also not the same and every guide translate on his own explanation style, however there are base words in the sentence that determine most of the translation. Also we must take care of grammar differences between the two languages.

This processes are classified into:

1) Suggested Sentences
2) Sentences Filter - Optimization
3) N-gram Scores
4) Translate
5) Grammar Correction

Suggested Sentences

There are no spaces in Hieroglyphic language so every word is directly attached to the next word. The idea behind this phase came from the following: Imagine English had no spaces so we test each combination of words to see which combination is valid.

For example, the sentence “Thegladiatorisstrong”. Trying to take “t” or “th” would fail, but taking “the” would succeed. Also taking “the g” would fail. Moving to the next word, “glad” and “gladiator” would succeed. But if “glad” was picked, it would ruin the rest of the sentence, so “gladiator” is picked then “is” and “strong” are picked. Hieroglyphics can be looked at in that same manner. Pretending “iator” is a word, if a scribe wanted to say “glad”, he/she would never write “iator” right after “glad” in order not to confuse readers. This is the main reason Egyptians thought a space is not necessary.

So this phase output is a number of suggested sentences and then in the next phases these sentences will be filtered to get a few number of sentences and each sentence will be given a score.

Sentence Filter - Optimization

Heuristic Search Concept

A Heuristic is a technique to solve a problem faster than classic methods, or to find an approximate solution when classic methods cannot. This is a kind of a shortcut as we often trade one of optimality, completeness, accuracy, or precision for speed. A Heuristic (or a heuristic function) takes a look at search algorithms. At each branching step, it evaluates the available information and makes a decision on which branch to follow.

The objective of a heuristic is to produce a solution in a reasonable time frame that is good enough for solving the problem at hand. It may simply approximate the exact solution. But it is still valuable because finding it does not require a prohibitively long time.

We can go and search for all suggested sentences scores but we will take few minutes to do that task so we make use of Heuristic search concept to discard sentences that are symmetric. This symmetry is in the words as we have many sentences starts and contains the same word and if this word has no meaning in the lexicon we can discard all of these wrong sentences. So number of sentences will be reduced to only few sentences which greatly fasten and ease the next step, n-gram scores, computation and that means that this step is for optimization.

N-gram Scores

An n-gram model is a type of probabilistic language model for predicting the next item in such a sequence in the form of a $(n - 1)$–order Markov model. In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs but in our application items are letters or glyphs. The n-grams typically are collected from a text or speech corpus.

Using Latin numerical prefixes, an n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram" (or, less commonly, a "digram"); size 3 is a "trigram". English cardinal numbers are sometimes used, e.g., "four-gram", "five-gram", and so on. Two benefits of n-gram models (and algorithms that use them) are simplicity and scalability – with larger n, a model can store more context with a well-understood space–time tradeoff, enabling small experiments to scale up efficiently.

Example : human

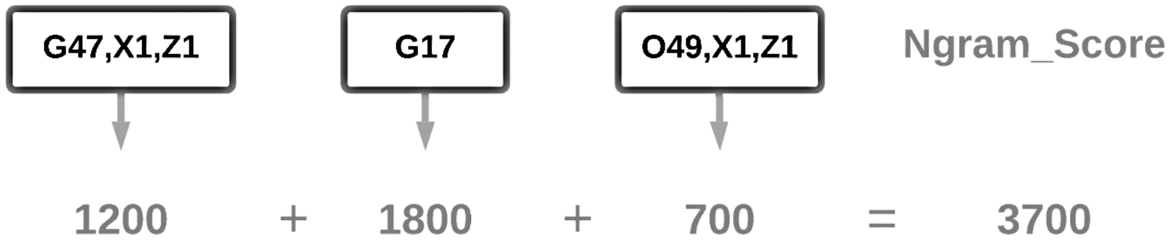
In uni-gram: h-u-m-a-n

In bi-grams: hu-um-ma-an

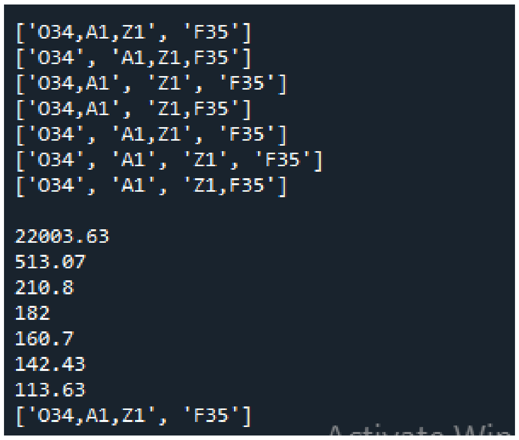
In tri-grams: hum-uma-man

The main use of N-gram language model in our project is to add spaces between glyphs according to the number of words in the sentence. But to make use of this model we give each sentence of the few suggested sentences a probability score and sentences with highest scores will be the right candidates.

The scoring system is really efficient and give very high and accurate results. Its idea is that each sentence of the suggested sentences consists of some words so each word in each sentence will get a probability score then we will add all the word scores in the same sentence so each sentence will get a score.



As we can see there are a huge differences in scores as the first score is very high related to the second or the other score and that ensures that the scoring system is successful and has a very high accuracy

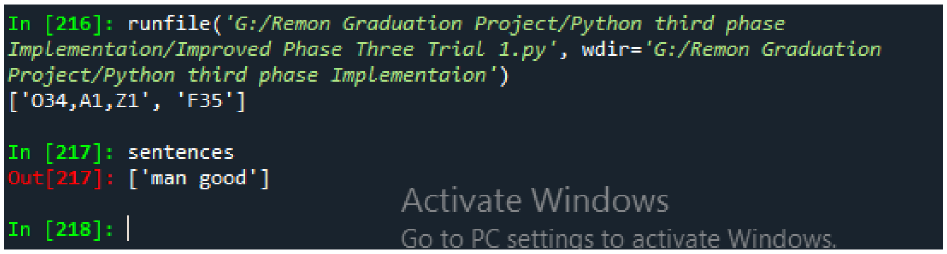


```
['O34,A1,Z1', 'F35']
['O34', 'A1,Z1,F35']
['O34,A1', 'Z1', 'F35']
['O34,A1', 'Z1,F35']
['O34', 'A1,Z1', 'F35']
['O34', 'A1', 'Z1', 'F35']
['O34', 'A1', 'Z1,F35']

22003.63
513.07
210.8
182
160.7
142.43
113.63
['O34,A1,Z1', 'F35']
```

Translate

The sentences will be ordered in descending order and the first sentence in the list or the highest score sentence is the right candidate.

After that we will take each word in the result sentence and check for its meaning in the lexicon to get the right meaning.



```
In [216]: runfile('G:/Remon Graduation Project/Python third phase
Implementaion/Improved Phase Three Trial 1.py', wdir='G:/Remon Graduation
Project/Python third phase Implementaion')
['O34,A1,Z1', 'F35']

In [217]: sentences
Out[217]: ['man good']

In [218]: |
```

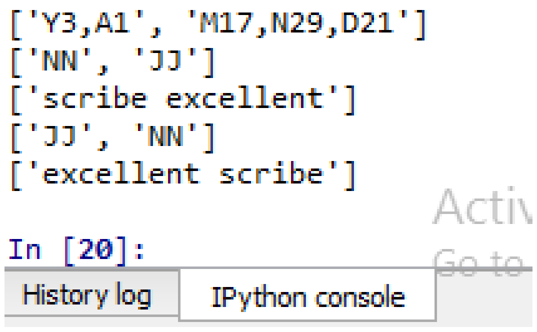
Since each word may have multiple mappings in translation associated with multiple types we will deal with this problem by taking the most important and famous word meanings that is widespread in most translations.

Grammar Correction

After translation of the Hieroglyphic sentence to English sentence we must take care of some grammar rules that is different between the two languages and some restructure processes required to organize the English sentence in a good format.

Sentence go through three phases representing different grammar rules, in order. In phase 1, which is “Adjectives as modifiers”. It looks for adjectives modifying a noun. In English, when adjectives come as modifiers to nouns, they precede the noun, for example “the excellent plan”. On the other hand, in Hieroglyphic, the adjective follows the noun, literally “the plan excellent”. There could be also more than one adjective following each other to describe a noun. The aim of this phase is to fix this problem.

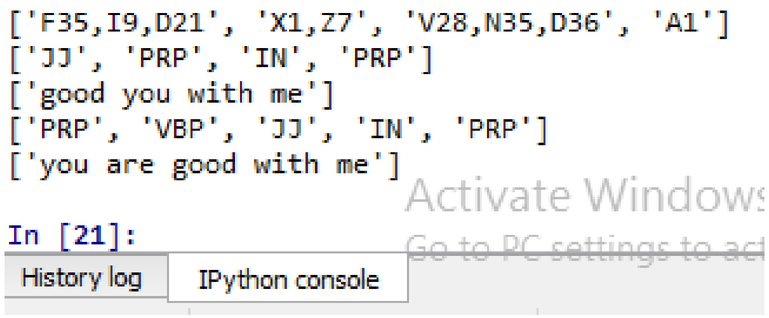
Then their order is switched so that the noun follows the adjective(s). If no modifier adjectives were found, the sentence will proceed to the next phase without any changes. On contrary, if modifier adjectives were present the exchange process will be done and the sentence will proceed to the next phase.



```
['Y3,A1', 'M17,N29,D21']
['NN', 'JJ']
['scribe excellent']
['JJ', 'NN']
['excellent scribe']

In [20]:
```

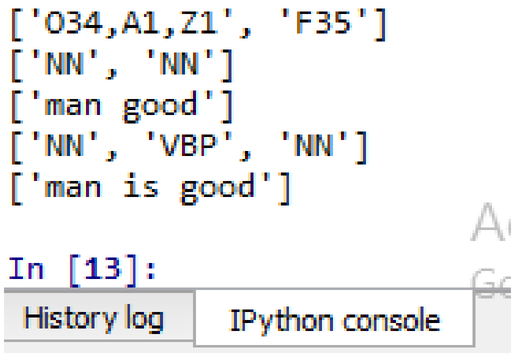
In the next phase, “Pronouns” phase, a very common use of pronouns is with nouns. In English, when used with nouns, pronouns always precede nouns, for example, “my book”, “this man” and “their house”. In Hieroglyphic, there is no single rule for all pronouns. Some pronouns precede nouns, others follow nouns. In a similar manner as the previous phase, this phase tries to fix this. Furthermore in case of a pronoun that follows a noun, the order is switched.

The effect of the previous phase can be seen when a two words of a noun and its adjective(s) is encountered. For example, after exchanging “plan” and “excellent” in the previous phase and passing “excellent plan” to this phase, the next word could be “this”, which will be switched with the noun resulting in “this excellent plan”.

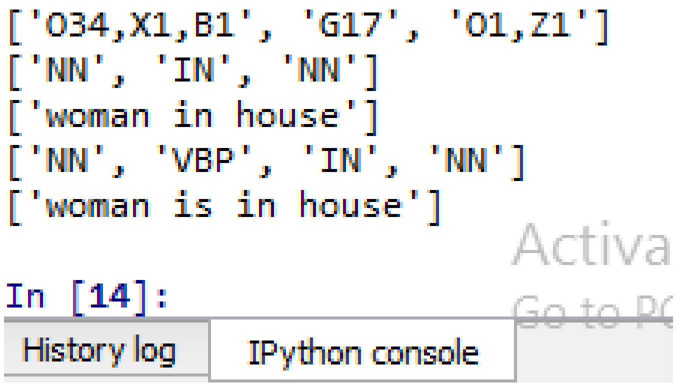


```
['F35,I9,D21', 'X1,Z7', 'V28,N35,D36', 'A1']
['JJ', 'PRP', 'IN', 'PRP']
['good you with me']
['PRP', 'VBP', 'JJ', 'IN', 'PRP']
['you are good with me']

In [21]:
```

In the following phase, “Consecutive nouns” phase, the main aim is to look for any consecutive nouns and try to guess what kind of relationship exists between the nouns. If more than two nouns followed each other, then the program assumes this relationship as connection, since lists were common in Hieroglyphics. When exactly two nouns come together, it is hard to tell their relationship so to solve this problem, if there is no verb in the sentence this relationship could result in verb to be as in Hieroglyph verb to be is sometimes removed or it could result in “and/or/of” that is added between the nouns. Accordingly, when the user reads the output, they can understand the context.

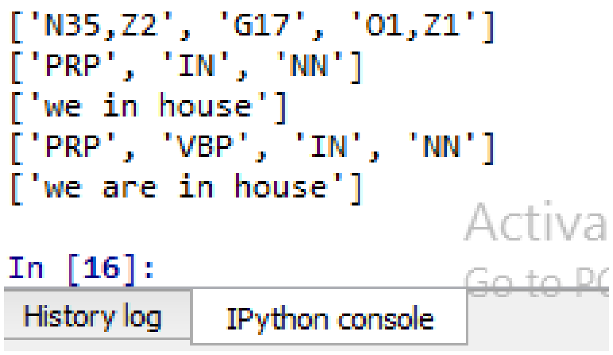


```
['O34,A1,Z1', 'F35']
['NN', 'NN']
['man good']
['NN', 'VBP', 'NN']
['man is good']

In [13]:
```

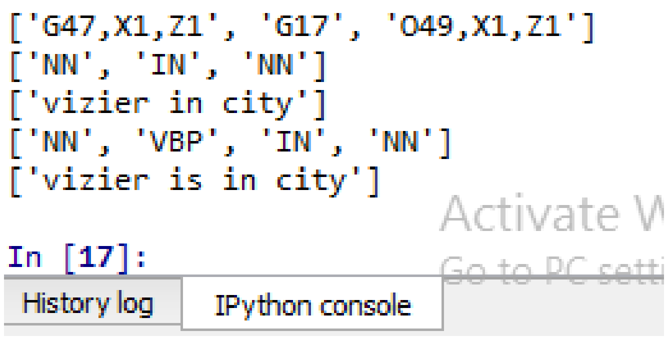


```
['O34,X1,B1', 'G17', 'O1,Z1']
['NN', 'IN', 'NN']
['woman in house']
['NN', 'VBP', 'IN', 'NN']
['woman is in house']

In [14]:
```

The translation system also detect the singular and plural nouns to add right verb to be when needed



```
['N35,Z2', 'G17', 'O1,Z1']
['PRP', 'IN', 'NN']
['we in house']
['PRP', 'VBP', 'IN', 'NN']
['we are in house']

In [16]:
```



```
['G47,X1,Z1', 'G17', 'O49,X1,Z1']
['NN', 'IN', 'NN']
['vizier in city']
['NN', 'VBP', 'IN', 'NN']
['vizier is in city']

In [17]:
```

Augmented Reality (AR)

Augmented Reality with Unity game engine and Flutter

What if the line between your imagination and the real world didn't exist? With augmented reality, not only is that possible, it's here. Augmented reality (AR) is an interactive experience of a real-world environment where the objects that reside in the real world are enhanced by digital visual elements, sound or other sensory stimuli delivered via technology. AR can be defined as a system that incorporates three basic features: a combination of real and virtual worlds, real-time interaction, and accurate 3D registration of virtual and real objects.

AR can be used in our project to increase the visual quality with the technical one. With AR we introduce high level of interaction with the historic atmosphere and a very good experience that you will never forget. AR transforms how you work, learn and connect with the world around you. It's the perfect way to visualize things that would be impossible or impractical to see otherwise.

What about seeing pharaohs in real life after all these years beside their statues and cartouches is not it a good experience to live in any historical place you visit? Many tourists visit Egypt but a few of them that really understand what he sees from either glyphs carved on walls and stones or in any age these things were made but with our application any tourist can open his camera and point it to any king cartouche and then he will listen to a voice that talks about the king that this cartouche related to with some music and he will see a 3D statue of the king appears beside it in real life. This also happen when he points the camera to any statue to know its character history.

So we used Unity game engine and AR Foundation and ARCore package to build AR system and we used Augmented Reality to enhance natural environments or situations and offer perceptually enriched experiences. With the help of advanced AR technologies object recognition the information about the surrounding real world of the user becomes interactive and digitally manipulated. Information about the environment and its objects is overlaid on the real world.

Image Tracking

The Augmented Images APIs in ARCore lets you build AR apps that can detect and augment 2D images in the user's environment, such as posters or product packaging and instantiate 3D models, 2D photos or a video. You provide a set of reference images. ARCore uses a computer vision algorithm to extract features from the grayscale information in each image, and stores a representation of these features in one or more Augmented Image databases.

At runtime, ARCore searches for these features on flat surfaces in the user's environment. This lets ARCore detect these images in the world and estimate their position, orientation, and size if one is not provided.

Capabilities

ARCore can track up to 20 images simultaneously. ARCore will not simultaneously detect or track multiple instances of the same image. Each Augmented Image database can store information up to 1,000 reference images. There's no limit to the number of databases, but only one database can be active at any given time. Images can be added to an Augmented Image database at runtime, up to the 1,000-image per-database limit.

When adding an image, it's possible to provide the physical size of the image to detect. Doing so will improve image detection performance:

1) If no physical size is provided, ARCore estimates the size and refines this estimate over time.
2) If a physical size is provided, ARCore uses the provided size and estimates the image's position and orientation, ignoring any discrepancy between apparent or actual size and the provided physical size.

ARCore can respond to and track images that are:

1) Images that are fixed in place, such as a print hanging on a wall or a magazine on a table
2) Moving images, such as an advertisement on a passing bus or an image on a flat object held by the user as they move their hands around.

Requirement

Images must:

1) Fill at least 25% of the camera frame to be initially detected.
2) Be flat (for example, not wrinkled or wrapped around a bottle).
3) Be in clear view of the camera. They should not be partially obscured, viewed at a highly oblique angle, or viewed when the camera is moving too fast due to motion blur.

Optimizing tracking and performance considerations

Depending on which ARCore features are already enabled, enabling Augmented Images might increase ARCore's CPU utilization so we keep everything simple in this phase so we got high FPS with the base functionality working. This will make additional CPU cycles available to our app, and improve thermal performance and battery life.

Chapter Four

Software

Software Requirements

Translation Software

The main language used in this project is python and we used many algorithms from N-gram language model to grammar correction system and scoring system and we used string manipulation system in large scale to deal with thousands of sentences to organize, order and manipulate data.

We used nltk as a natural language processing (NLP) package to help us in organizing translating sentences and nltk also introduce helpful models like n-gram many corpora to test on. The main use of nltk is in POS Tagging and word tokenization as we need to correct the grammar of the translated sentence so we need to use these two steps in our project.

Word Tokenization

Tokenization is the process by which a large quantity of text is divided into smaller parts called tokens. These tokens are very useful for finding patterns and are considered as a base step for stemming and lemmatization. Tokenization also helps to substitute sensitive data elements with non-sensitive data elements.

POS Tagging

Part-of-speech (POS) tagging is a popular Natural Language Processing process which refers to categorizing words in a text (corpus) in correspondence with a particular part of speech as identification of words as nouns, verbs, adjectives, adverbs, etc. depending on the definition of the word and its context. By defining POS Tagging we will know the sentence structure and many grammar rules. We use Stanford POS Tagging from nltk as it more accurate than the Tagging built in nltk.

Unity game engine Augmented Reality and Flutter

Aim

We used AR that is presented by Unity game engine in the package names AR Foundation and ARCore. The built project is then presented on Flutter app so we managed to open Unity AR as real-time application running on a widget of Flutter application that is connected to the other Flutter features like Hieroglyph detection, translation and others. So we have Connected Flutter with Unity real-time rendering.

Implementation of AR

In Unity we managed to use Image Tracking to detect and track Cartouches of kings in ancient Egypt and after detection a 3d statue will appear and a voice talking about the king mentioned in the Cartouche with an addition of some sound effects. Image.

Connecting Flutter with Unity real-time rendering

A great achievement is to connect two complicated programs as a game engine that uses C# as its programming language and new framework like Flutter that uses dart programing language. This connection requires complicated communication between the two programs so that they can work smoothly with each other and this communication is managed by the two programs.

Chapter Five

Conclusion

Glyph Classification ML SVM Model evaluation

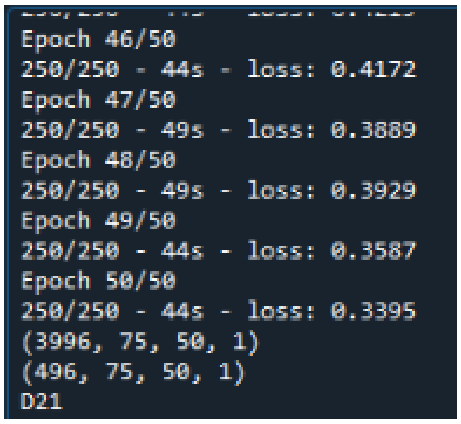
Model accuracy using hold-out method (train/test) (70/30) : 96%

Classifier Performance	
Input size (images)	Time (ms)
20	54.75
25	87.40
50	128.11
100	366.57

Deep Learning Glyph Classification using One Shot Learning with Siamese Networks using Keras Model evaluation

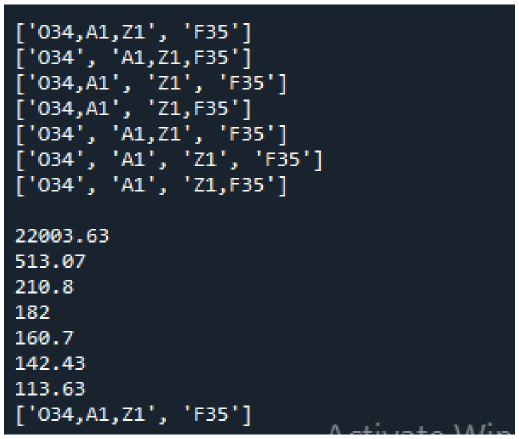
The model predicting accuracy is very high compared to ML model as ML model accuracy is 96% when testing on the trained dataset but when using any input image slightly different or from other dataset it results with low accuracy about 20% but after using Deep Learning model which is One Shot Learning with Siamese Networks the accuracy keeps about 92% in all cases so if we enter any image taken from real temple wall, any other dataset or colored graphic Hieroglyphic letters the output keeps true.

The loss function in the following image ensures that:


```
Epoch 46/50
250/250 - 44s - loss: 0.4172
Epoch 47/50
250/250 - 49s - loss: 0.3889
Epoch 48/50
250/250 - 49s - loss: 0.3929
Epoch 49/50
250/250 - 44s - loss: 0.3587
Epoch 50/50
250/250 - 44s - loss: 0.3395
(3996, 75, 50, 1)
(496, 75, 50, 1)
D21
```

Translation Performance Evaluation

We can say that the translation phase has really succeeded in its job and gives high accuracy as the sentence is reformatted many times inside this phase to reach a very organized and accurate English format. Translation accuracy is about 97% for basic sentences and 93% for complicated sentences and the reason for this high accuracy is two main stages of processing. First is Sentence Filter that removes most of wrong sentences assumptions and keep only the few valid ones using Heuristic Concept and the second is N-gram model stage that gives a high quality probability scores that eases the translation phase a lot. The image below show the big difference in scores of each sentence with the first sentence only has a very high score related to other sentences scores and that ensures the very high accuracy of scoring system.


```
['O34,A1,Z1', 'F35']
['O34', 'A1,Z1,F35']
['O34,A1', 'Z1', 'F35']
['O34,A1', 'Z1,F35']
['O34', 'A1,Z1', 'F35']
['O34', 'A1', 'Z1', 'F35']
['O34', 'A1', 'Z1,F35']

22003.63
513.07
210.8
182
160.7
142.43
113.63
['O34,A1,Z1', 'F35']
```

Chapter Six

Future work

This module is an important module for the application and if it functions well with a high accuracy, it will help the other modules to produce a accurate results that will lead to a good translation so we can add some

X.3.1. Detect if the text is non hieroglyphic:

Sometimes the image may contain more than a language and the user may not crop the hieroglyphic part only which may cause confusion since it will be detected as a contour by this module and passed to the other modules as if it is a hieroglyphic character.

So, we can overcome this problem by using the libraries for other languages like Tesseract and EasyOCR to check if the image contains other languages to exclude this part from the image before starting the detection.

X.3.2. Rerun and Change the Parameters:

We can rerun the module several time and every time we run it we can change the used parameters (e.g. canny edge upper and lower threshold, minimum accepted area, ...) to get different results and use the high voted results.

X.3.3. Use The User Help:

We can add some features in the app that let the user help to increase the accuracy like asking him if the detected characters are really correct by showing them to him, or to choose the reading direction, or maybe adding lines to the lineless images.

Chapter Seven

References

"Automatic egyptian hieroglyph recognition by retrieving images as texts" by Morris Franken and Jan C van Gemert. 2013. In ACM Multimedia Conference. ACM, 765–768

• https://scikit-learn.org/stable/modules/svm.html#svm-classification

• https://wiki.haskell.org/Lazy_evaluation

• https://pillow.readthedocs.io/en/stable/reference/ Image.html#PIL.Image.Image.convert

• https://pillow.readthedocs.io/en/stable/handbook/ concepts.html#concept-filters

• N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In CVPR, 2005.

• http://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/svms/ RBFKernel.pdf

• Wu, Lin and Weng, "Probability estimates for multi-class classification by pairwise coupling", JMLR 5:975-1005, 2004.

• https://pillow.readthedocs.io/en/stable/handbook/ concepts.html#concept-modes
• https://www.machinecurve.com/index.php/2020/02/10/using-constant- padding-reflection-padding-and-replication-padding-with-keras/ #reflection-padding

• https://scikit-image.org/docs/dev/api/skimage.feature.html?highlight=hog#skimage.feature.hog

• https://scikit-image.org/docs/dev/auto_examples/features_detection/ plot_hog.html#sphx-glr-auto-examples-features-detection-plot-hog-py

• https://homepages.cae.wisc.edu/~ece539/fall13/project/Chen_rpt.pdf

• https://courses.analyticsvidhya.com/courses/applied-machine-learning- beginner-to-professional

• https://www.youtube.com/watch?v=GuqBJ4W7_58&list=PL1MGGQM589HJjtSAiJJIrLxWGxX008bfA &index=6
• http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html