Architecture Evaluation

MEXICAN SIGN LANGUAGE ALPHABET IMAGE CLASIFICATION SYSTEM

Erick Ibarra
A00959090

March 27, 2017

PURPOSE

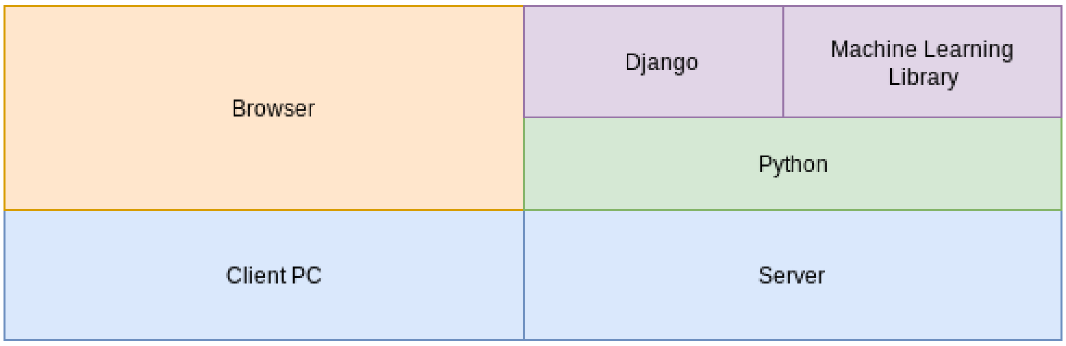
The development of this document has as an objective to evaluate the architecture of the developed Mexican Sign Language Alphabet Image Classification System based on the initial needs and to assess that the chosen architecture actually meets the desired quality attributes.

1 OVERVIEW

The Mexican Sign Language Alphabet Image Classification System is able to take an image as input and predict the letter the hand is representing from the possible alphabet letters. To achieve this the system must know how a hand is represented in an image and the different signs it can make. The complete application is in a version control system in order to be more mantainable, the system used is git + github for the web interface and can be seen visiting the following link: `https://github.com/erikiado/erikiado`. The working application is hosted in the following use and can be used to test the application: `http://www.erikiado.com/`. All the rest of the documents, deliverables, as well as some images to test the system are available under github `https://github.com/erikiado/erikiado`.

2 Architecture Overview and Evaluation

The general architecture decided to build the system is described in the following diagram some of the details on how this components are used are described in the Functional Requirements Document. Most of the decisions were taken with the quality attributes of availability, performance and maintainability on mind. These are the main quality attributes considered for these application, but some others, like security were taken care of.



Although these are the main components interacting live in the system, other components were developed in order to produce an accurate model which later was used to make the classification of the images. These components are some scripts developed in order to create the dataset and expand it, and the to create the actual model and train it multiply times iterating through different configurations. These scripts can be found under the erikiado/mexican_signs module. This architecture ensures the system will be available without me needing to be there to demonstrate it since it is a web application anyone can access it. Most modules were developed in the OO Paradigm in order to use some of its principles to ensure the code is easily mantainable. The performance quality is met in the sense that the classifier can give a prediction in less than 3 seconds. This was possible thanks to the size of the image which is being sent as an input. In the following sections a view to the different components and modules decisions will be explained and some of the standards which were used are going to be introduced.

2.1 Modules

The whole application is wrapped under a Django application which only purpose is to serve the page which displays the instructions of the application and the general roadmap to get to the finishing result, also this page presents the classifier functionality and when it receives the POST request with a valid image the result is displayed in the same page. Since the main focus of the project was to learn more about computer vision and machine learning, most of the time was dedicated to this so a few hours of work could be allocated to the GUI itself and that is why I chose to not complicate myself reinventing the wheel. The following modules are the ones developed for the system and can be found under the erikiado/mexican_signs module.

- dataset_creator.py
  This component implements the functionality used to detect a hand in an image feed from the webcam of your computer. After it detects the hand on the image it save the ROI(Region of the Image) with the hand in multiple folders. Each folder represents a representation of the image spanning from the original image all the way to a mask of 20x20 of just the hand, passing though different representation such as the original image cut on the ROI of the hand and some others.

- dataset_multiplier.py
  Once the dataset is created each of the images is then transformed by this component multiple times(12) in order to ensure that the model is going to be trained very well not just for specific cases, but for general cases too. This is achieved thanks to the different positions and forms of representing the same sign. This will lead the model to learn how a hand looks generally and how the different variations may represent a single label or class.

- nn_model.py
  This is the main script of the application in the sense that the model creation and training occurs in these module. This component was constantly used and iterated to make sure the model improves the accuracy as desired. The model itself consist of 6 layers. Two dense layers ensure the inputs are correctly interpreted and inputed to the different sections of the model, both of theses use the RELU activation layer in order to produce faster results. After each dense layer a dropout layer is connected in order to prevent overfitting of the model by droping a small

percentage of the neurons each epoch in order to train each neuron better and not make the neurons completely depend on the knowledge of all the connected neurons. Lastly the result of these layers is fed to a third dense layer which will be activated by a Softmax function in order to produce the probabilities of each class but with the important effect that the sum of all the probabilties is equal to 1. This quality ensures that a single class will be the one with the most probability of occuring.

- hand_classifier.py
  This component is the only object oriented component in the system. This decision was made because most the other components are just standalone scripts which make a single thing that had to be run a single time. On the other hand, this component is what makes the actual application and interacts with other components of the system such as the server. This also uses the singleton pattern to avoid creating multiple instances and slowing down the system and affecting performance. This component main objective is to load the previously created ML model and build an interface by which the web application can pass an image url and get a prediction.

2.2 Tests

Some tests were developed in order to ensure the functionality of the system works as expected. These tests can be found under the erikiado module and some tests unit tests were developed in order to test the functionality of the classifier and its accuracy and to ensure the page is being served correctly with no errors. Some integration tests were developed in order to assess the quality of the overall system is met. This test mock a user interaction by using an actual browser to test and automating a sequence of clicks and decisions the user can make while using the system. These tests make sure the system receives a valid image as an input and that it can display the results of the classifier correctly. As most of the guides for the system this can be found in the README in the github repository.

2.3 Architecture Patterns

Diffent architecture patterns were used, but the most important for the way the application is working are the following two:

- MVC
  This pattern is used in the web application and helps avoid complicated code by separating the presentation of the information from the model functionality and from the controller of the model. This is used by default with the Django framework and helps organize modules better.

- Client-Server
  This pattern is required in order to ensure the system is available to the desired users when needed. This is different from having a local program which is not available at any time. This way someone with a computer as a client can connect to my server in order to use the application and receive information from the server.

2.4 Design Patterns

The most important design patterns followed to build the application are explained in the next section:

- Singleton
  This pattern is very important to ensure one of the main qualities of the systen which is performance. This as stated previosuly is implemented in the hand_classifier module which is called each time the request is made. To avoid having to load the model each time which takes more than 5 seconds a single model is created and if it is needed again, the class checks if a instance is already loaded and uses it instead of creating other instance.

- Single Responsability Principle
  This principle is the most repetitive in the sens that all the modules have a single task to make and they don't mix with each other. Most code which is going to be reused is summarized in functions. This helps with mantainability in the sense that most code is has a single task to achieve and if that task is not achieved it is a single component which needs to be changed or debugged.

- KISS Principle
  This principle is used by inheritance of the design of the python language which main objective is to keep a simple and understandable

code. This also reproduces in all the components in which their functions are made in the simplest way possible not mixing codes from other functions and by refactoring code once it was completely functional..

2.5 Guides

In order to see more information on how to install and/or use the application the Readme under the github repository explains in more detail these steps. Also a general video and user guide can be found under the documents directory in the github repository: `https://github.com/erikiado/erikiado`.

3 Standards

Standards were used in order to ensure the quality of the code both in the way the code is written and presented and how it is documented in order to ensure it is readable.

3.1 Code Style

The PEP-8 standard for code was used in the development of this project. The tool flake8 which is part of the dependencies is used to lint the code following this standard. `https://www.python.org/dev/peps/pep-0008/`

3.2 Documentation

For the documentation the numpy standard is being used and every piece of code developed is documented to assess the code is understandable by any person which decides to access it. The numpy standard can be found in the following link: `https://github.com/numpy/numpy/blob/master/doc/HOWTO_DOCUMENT.rst.txt`