Cheesy Identifier

For

Project Report

10-May-2020

Prepared by: Team Cheese

Team Members: Andre De Macedo Wlodkovski

Darren Greene

Gustavo Hammerschmidt

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Gustavo | Apr. 15 | Project creation | 375e0b8 |
| Andre | Apr. 15 | Created training python file | A29618a |
| Andre | Apr. 21 | Update training | 582d4d0 |
| Gustavo | Apr. 21 | Update database | 28d47e6 |
| Gustavo | Apr. 29 | Update database | D7636a8 |
| Andre | Apr. 30 | Update training: model is saved | 2e12499 |
| Andre | May 1 | Started implementing networking | 0bcf917 |
| Andre | May 1 | Networking completed | Cf0981e |
| Andre | May 9 | User-friendliness implementation (UI & class names) | 94fe7d0 |

Table of Contents

[1. Executive Summary 4](#_Toc39414090)

[2. Project Overview 5](#_Toc39414091)

[2.1 Problem Statement 5](#_Toc39414092)

[2.2 Scope 5](#_Toc39414093)

[2.3 Project Team 5](#_Toc39414094)

[3. Machine Learning Aspects 6](#_Toc39414095)

[3.1 Dataset 6](#_Toc39414096)

[3.2 Model Creation and Training 6](#_Toc39414097)

[3.3 Inference 6](#_Toc39414098)

[3.4 Evaluation of the Model 6](#_Toc39414099)

[4. Software 7](#_Toc39414100)

[4.1 System as a Software 7](#_Toc39414101)

[4.2 System Architecture 7](#_Toc39414102)

[4.3 User Interface Overview 7](#_Toc39414103)

[5. Use Cases 8](#_Toc39414104)

[6. References 9](#_Toc39414105)

[7. Glossary 10](#_Toc39414106)

[8. Appendices 11](#_Toc39414107)

# Executive Summary

In this project, we implemented a program that identifies a few types of cheese, such as Camembert, Cheddar, Cottage, Edam, Gorgonzola, Parmesan and Swiss. The program has a client-side, which is responsible for sending an image to the server side, who will do the inference and return the result to the client. The communication is based on a web-server and the back-end communication was implemented in JS Query.

The most interesting experience was researching ways to improve every single bit of the project, such as Database, Machine Learning and Design. Examples of that are the PHP Query for database, which was responsible for getting about 75% of all of the training and testing images, then on the Machine Learning part it can be mentioned that we had to do a lot of optimization to minimize overfitting, especially with a small database.

Since the database is fairly small, we have had to remove a few classes, because the accuracy results were not bad at all. After removing those classes, we have had a significant accuracy and performance boost.

# Project Overview

## Problem Statement

The purpose of the Machine Learning project is to help beginners to identify a few types of cheese. Therefore, anyone could access the website and send a cheese image to be identified.

## Scope

The software does not identify all possible types of cheese, because it would require a huge database, with a much bigger team and much more resources. Therefore, it only identifies the following types of cheese: Camembert, Cheddar, Cottage, Edam, Gorgonzola, Parmesan and Swiss.

The main goals of the project are to deliver a Machine Learning model that is about 75% accurate, with fast inferences, and a pleasant website design.

## Project Team

Name of the project:

|  |  |  |  |
| --- | --- | --- | --- |
| Name of the Team member | Responsibility | Contribution % | Notes |
| Andre De Macedo Wlodkovski | Train model and establish server communication | 33.3% | N/A |
| Darren Greene | Design website | 33.3% | N/A |
| Gustavo Hammerschmidt | Gather database resources | 33.3% | N/A |

**Please note:** The total contribution of all team members shall add up to 100%. For example, if you have 3 people in the group and each of them contributed equally, then each member’s contribution is 33.3%.

# Machine Learning Aspects

## Dataset

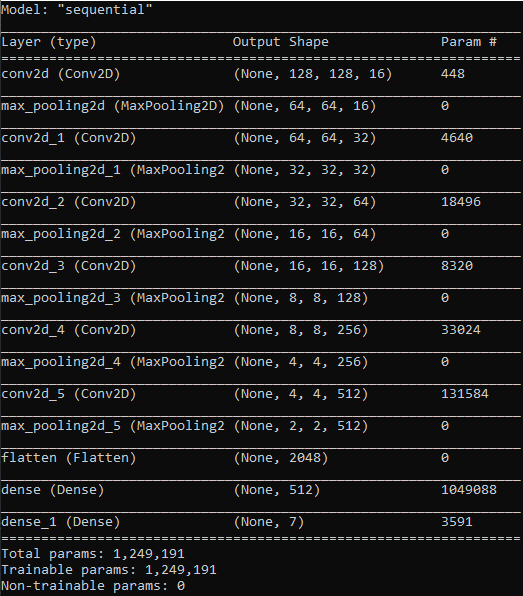
The dataset consists of cheese images, and it has been acquired from gathering images from Google Images, with a total of 828 images.

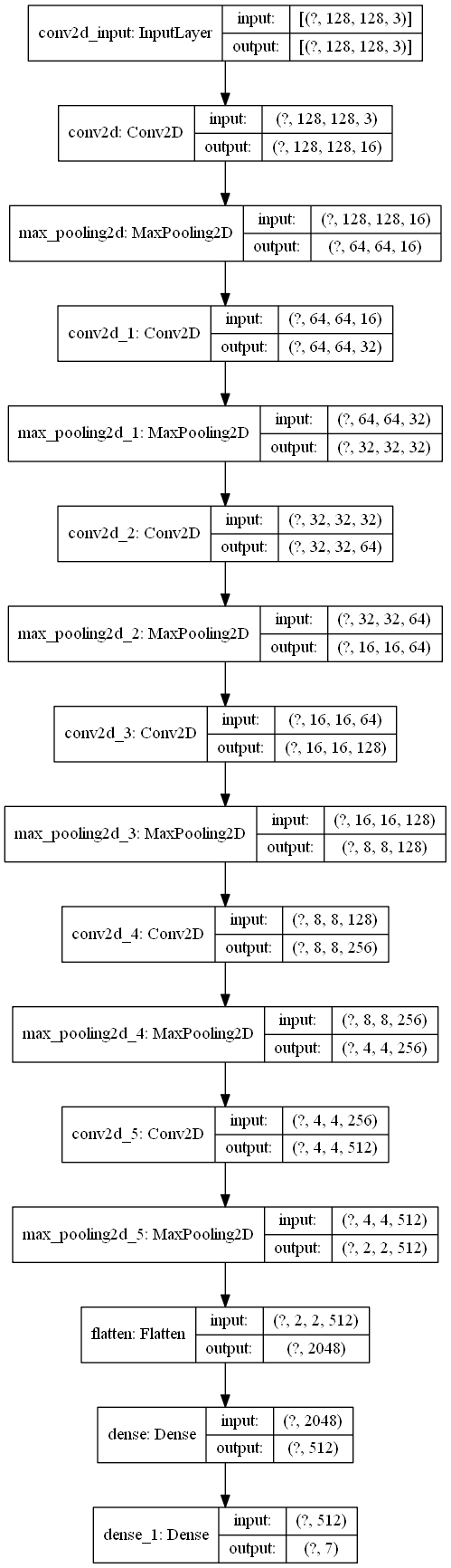
Since the database is small, there was a lot of overfitting in the beginning, so we have had to do some data augmentation, which includes shuffling images, vertical and horizontal flipping, width and height shift, and rotating the images.

The total split of training and validation images is about 75% training and 25% validation, with 7 batches of 256 images generated for training and 1 batch of 512 images generated for validation.

To gather the database, python control files were made to define calls to PHP files made, these python files defined the inputs of the PHP files and how should the program run them, once the python main file was threaded to perform better. To execute the PHP files, a application that creates a local server connection was used, named xampp. In order to define the means by which the database was extracted, all files involved on the process were uploaded with the application, except for the xampp application files, that have no impact on the project development.

## Model Creation and Training





As mentioned before, the training data is made of 7 batches of 256 images generated from the ImageDataGenerator class to provide augmentation, and the total amount of epochs is 40.

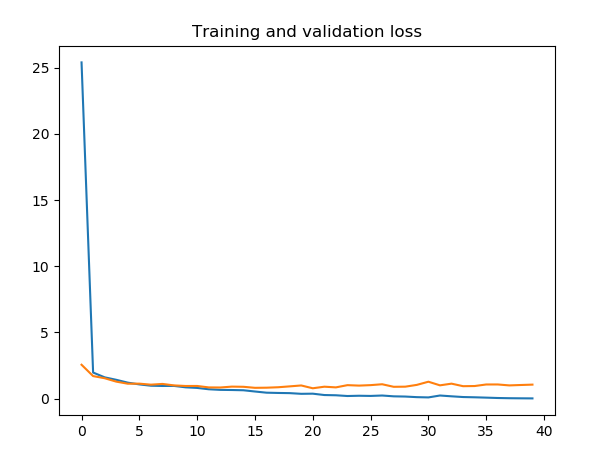
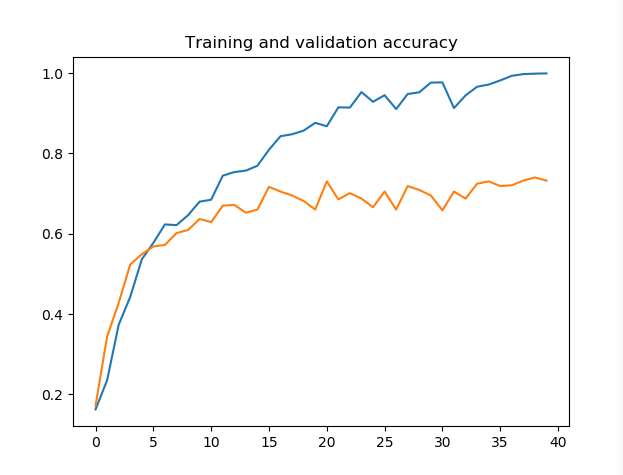
The layer setup was designed in a way to keep the model efficient and fast. In the first tests, there were a few layers with bad results and huge model sizes, such as 96MB. After adding convolutional layers with parameters 128, 256 and 512, respectively, the model showed significant better results and much smaller size: 14MB.

## Inference

The inference process is the simplest one regarding the machine-learning part. In order to display smooth names instead of IDs, each cheese ID was mapped to a smoother display name in a list (e.g camembert\_cheese becomes Camembert). When a client requests a prediction, it converts the image received to a matrix, resizes it, loads the model from disk and predicts it, returning the smooth name to the client.

## Evaluation of the Model

The final results were satisfying, given the database is not ideal because it is small, there was still an accuracy of 75%, meaning most of the data has been generalized. The loss values for training and validation are also quite similar.



# Software

## System as a Software

System functionalities:

* Upload an image: this functionality allows the user to upload an image to the server running the model. When an image is uploaded, the software state changes from idle to busy once it will predict the label of the image. After the model finishes the prediction, it redirects the user to the prediction answer html page which will display the cheese type of the image.
* Cancel upload: when the user clicks the upload button, it can either select and upload an image or cancel the request to upload.
* Return to upload page: At the prediction answer html page, the user can press a button to return to the index page, the functionality of it is to simply redirect the user back to able to make another prediction if it wishes to.

## System Architecture

The project structure was delivered as a client-server architecture application. Our application has a database and a server running on a same computer and a client that interacts with the server either remotely or locally(in our example, the application was run only locally). The server interacts with the database at the beginning to train the model. For the application, the database has no other use rather than training the model, still we consider it relevant to the architecture and have added it here. Many users may access the server and get a answer to their uploaded image at the same time.

Uma imagem contendo relógio, branco, preto, atletismo

Descrição gerada automaticamente

The functional structure of the project is:

## Tela de celular com texto preto sobre fundo branco Descrição gerada automaticamenteMapa com linhas pretas em fundo branco Descrição gerada automaticamenteUser Interface Overview

The application’s User Interface was designed to be simple to use; it consists of a button that opens the user’s local-machine browser file explorer for the user to select a image on its computer. Once the user selects a image and confirms the selection, the image is sent to the server for it to make a prediction, a label is found from the prediction and it’s embedded into the answer html-page text div that is returned to the user. So, the user interface consists of two pages, the index and the answer pages. If no image is selected or the user cancel the selection, then he will remain on the index page. If he has sent an image and now finds himself on the answer page, then he can either remain on the page or click another button to return to the previous page. The user can always close the tab or the browser.

To install the ML system and software, the user must download the project folder. Then, he must execute the Server/server.py file on a anaconda prompt. Some libraries may be needed to be installed previously, such as: Pillow or TensorFlow. When the server is running, the user can open the index file on its browser with the path to it.

How to install your ML system and software <- Maybe add something to the paragraph above

How to repeat your ML training and inference process

Running your software system <- Write about these topics too.

* Include the screen capture images of main functional user interfaces

# Use Cases

* Describe your use cases of using (at least two use cases ) your own ML system software
  + Describe each use case with input data
  + Describe results your system produced
  + Include screen capture images captured doing these test use cases

# References

* Provide a list of all documents and other sources of information referenced in your project. Include resource/document title, date, and author for each.

# Glossary

* Define all terms and acronyms required to understand your project and this report.

# Appendices

* Include any relevant appendices (if any).