UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

DEVELOPMENT OF A FOOD IMAGE CLASSIFICATION SYSTEM BASED ON TRANSFER LEARNING WITH CONVOLUTIONAL NEURAL NETWORKS

ALBERTO SÁNCHEZ LÓPEZ

2019

TRABAJO DE FIN DE MÁSTER

Título:	DESARROLLO DE UN SISTEMA DE CLASIFICACIÓN DE IMÁGENES DE COMIDA BASADO EN TRANSFERENCIA DE APRENDIZAJE CON REDES NEURONALES CONVOLUCIONALES
Título (inglés):	DEVELOPMENT OF A FOOD IMAGE CLASSIFICATION SYSTEM BASED ON TRANSFER LEARNING WITH CONVOLUTIONAL NEURAL NETWORKS
Autor:	ALBERTO SÁNCHEZ LÓPEZ
Tutor:	CARLOS A. IGLESIAS FERNÁNDEZ
Ponente:	—
Departamento:	DEPARTAMENTO DE INGENIERÍA DE SISTEMAS TELEMÁTICOS

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos

Grupo de Sistemas Inteligentes

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

DEVELOPMENT OF A FOOD IMAGE CLASSIFICATION SYSTEM BASED ON TRANSFER LEARNING WITH CONVOLUTIONAL NEURAL NETWORKS

ALBERTO SÁNCHEZ LÓPEZ

2019

Resumen

Hasta hace unos años, la práctica común de la inteligencia artificial no era sostenible fuera de los laboratorios de investigación y la ciencia ficción. En la actualidad, existe una demanda generalizada de sistemas con inteligencia avanzada, capaces de simular el comportamiento humano. El aprendizaje automático ha sido el factor que ha fomentado el cambio, cubriendo la necesidad que permite aprender y generalizar a las maquinas bajo experiencia.

La evolución de la inteligencia artificial posiblemente pase por la visión artificial, disciplina donde el aprendizaje profundo proporciona mejoras notables asemejándose al sistema nervioso humano con neuronas artificiales. Entre las aplicaciones más destacadas de visión artificial se encuentran: detección de objetos, reconocimiento de objetos, reconstrucción de escenas, restauración de imágenes y estimación de movimiento.

El objetivo principal de este proyecto es el desarrollo de un sistema de clasificación de imágenes basado en transferencia de aprendizaje con redes neuronales convolucionales, centrándose en reconocimiento de objetos. Se emplean varias arquitecturas pre-entrenadas accesibles desde Keras, una API de alto nivel desarrollada sobre TensorFlow destacada para investigación en aprendizaje profundo. La evaluación del sistema se lleva a cabo a través de un popular conjunto de datos de comida etiquetada (ETHZ Food-101) con el que se mide la calidad de cada una de las arquitecturas anteriores.

El caso de uso práctico que se lleva a cabo con este sistema es la predicción de la clase y etiquetado de miles de imágenes de comida de restaurantes minadas desde TripAdvisor con Selenium. A partir de este nuevo conjunto de datos, se implementa un buscador de restaurantes basado en imágenes de comida, es decir, la búsqueda de restaurantes se realiza por el aspecto de las fotografías de comida. Todo lo mencionado anteriormente se desarrolla en una aplicación web para mostrar todos los objetivos alcanzados.

Palabras clave: redes neuronales convolucionales, keras, aprendizaje profundo, transferencia de aprendizaje, vision artificial, clasificador comida, web scraping, foodiefy, web application

Abstract

A few years ago, the common practice of artificial intelligence was not sustainable beyond research laboratories and science fiction. Currently, there is widespread demand for advanced systems capable of simulating human behavior intelligence. Machine learning has been the factor that has fostered change, covering the need that allows learning and generalizing to machines by experience.

The evolution of artificial intelligence may go through computer vision, a discipline where deep learning provides remarkable improvements resembling the human nervous system with artificial neurons. Among the most important machine vision applications include object detection, object recognition, scene reconstruction, restoration and image motion estimation.

The main goal of this project is developing an image classification system based on transfer learning with convolutional neural networks, focused on object recognition. Are employed several pre-trained architectures accessible from Keras, a high-level API developed on TensorFlow highlighted for deep learning research. System evaluation is carried out by a popular dataset of labeled food (ETHZ Food-101) which measures the quality of each of the previous architectures.

The practical case of use carried out with this system is the prediction of the class and labeling of thousands of food images of restaurants mined from TripAdvisor with Selenium. From this new set of data, a restaurant search engine based on food images is implemented, that means, the search of restaurants is done by food photographs appearance. Everything mentioned above is developed in a web application to show all the objectives achieved.

Keywords: convolutional neural networks, keras, deep learning, transfer learning, computer vision, food classifier, web scraping, foodiefy, web application

Agradecimientos

Quiero expresar mi agradecimiento a Marina Mollá Ballesteros, una de esas personas que marcan una vida por su simple forma de ser, equilibrio y apoyo incondicional.

Mi gratitud a todas las autoridades y personal que forman el Grupo de Sistemas Inteligentes (GSI), por abrirme las puertas, aportarme herramientas e incluirme como uno más en el proceso investigación.

De igual manera mis agradecimientos a la Universidad Politécnica de Madrid (UPM), a toda la Escuela Técnica Superior de Ingenieros de Telecomunicación (ETSIT), y a la Sapienza Università di Roma, que me han aportado las herramientas necesarias para llegar a donde estoy profesionalmente y me han permitido conocer gente extraordinaria.

Finalmente, quiero expresar mi más sincero agradecimiento al Dr. Carlos A. Iglesias Fernández, mi profesor y tutor durante todo este proceso, quién, con su dirección, conocimiento, y colaboración me permitió desarrollar este trabajo de fin de máster.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

This chapter introduces the context of the project, including a brief overview of all the different parts that will be discussed in the document. It will also break down a set of objectives to be carried out during the realization of the project. Furthermore, it will introduce the structure of the document with an overview of each chapter.

1.1 Context

Artificial Intelligence (AI) is known as the simulation of human intelligence processes by machines. These processes include those capable of learning, reasoning and improving themselves. AI has changed our lives substantially with the arrival of many products and services, but its beginnings were not easy. Currently, there is a widespread desire for systems with advanced intelligence, capable of simulating human behavior.

The key factors that have encouraged the use of this variety of systems were the increase in the computational capacity of computers and the digitalization produced huge amounts of data that could be processed to obtain value from them. An unprecedented investment was made by technological companies, seeing that by applying analytics and data algorithms, products, services, and insights could be obtained that would add value to companies and society. It seems that the advance of artificial intelligence goes through computer vision. Among the most important applications of computer vision are object detection, object recognition, scene reconstruction, image restoration, and movement estimation.

One of the fields that include artificial intelligence is deep learning, which focuses on the development of computer programs and systems that can manage data to transform it into useful information that generates knowledge without the need for human supervision. Deep learning techniques includes neural networks, specifically the Convolutional Neural Networks (ConvNets or CNNs) that is one of the main classes to perform image recognition and image detection.

Entering other topics of interest, food-related investigations propose multiple applications and services to improve health, guide human behavior and understand the culinary culture. The development of social networks has allowed the sharing of lifestyles through images, recipes, and diets, which have led to the collection of large datasets of information. This information grants acquiring the necessary knowledge of how food can help people to solve or improve current social problems. Whereas, search engines emerged from the need to classify and manage information in a reasonable time. Numerous websites and internet began to appear every time had a more generalized use reason why the first searchers, began to fulfill the function of classification of pages, documents, places, and servants of the network.

The way in which humans are related to technology today is largely due to the apps change, which has forced an evolution of the web to adapt to the already practically consolidated mobile environment. With the appearance of new devices, each with different screen resolutions that break into the market regularly, it is essential that the web project is adapted to them, so that the content is liquid, occupying all the available space, and thus allowing a correct experience for the user, no matter where you are accessing from.

1.2 Project goals

All objectives proposed for this project revolve around the context of the previous section. The aims are divided into general and specific depending on the purpose.

As **general** project goals, regarding previous job and research to carry out the the practical project part:

- Understand the current state of the art in terms of image recognition and deep learning techniques for image analysis.
- Investigate the possible methodologies and frameworks of formatting and extraction of information.
- Study the different ways of presenting information and reach the widest potential audience.

As **specific** project goals, regarding the businesslike activity that makes up the case study:

- Analyze the different image recognition architectures for transferring learning to choose the most suitable one.
- Build a deep learning model of image classification using transfer learning techniques to recognize food images.
- Extract the necessary information to elaborate a dataset and generate an extra value by applying the food image classifier model.
- Develop a multiplatform web application for all devices to show the project achievements and results.
- Design a general system architecture to fit each of the modules that make up the project.
- Implement all previous specific objectives as a case study approach.

1.3 Structure of this document

The remaining of this document is structured as follows:

- State of the Art ***(chapter 2)***. This chapter introduces the reader all necessary knowledge to understand the project and its different sections. It deals with the current state of food image recognition systems and the different methodologies to carry it out. Sections included in this chapter are Food image recognition, and Deep Learning for image analysis.

- Enabling Technologies ***(chapter 3)***. This chapter offers a brief review of the main technologies that have made possible this project, as well as some of the related published works. Sections included in this chapter are Machine Learning technologies, Data technologies, and Web application technologies.

- General System Structure ***(chapter 4)***. This episode summarizes the next chapters showing the general structure of the system. Modules, components, and connections are described and analyzed. Sections included in this chapter are Architecture, Main modules, and Auxiliary modules.

- Transfer Learning Model for Image Recognition ***(chapter 5)***. This chapter shows how to build a image classifier model transferring learning and analyzes multiple deep learning architectures. It describes the overall process of fine-tuning a model, with specific analysis of each architecture and comparison of results. Sections included in this chapter are Introduction, Methodology and configuration of the model building process, Analysis of the models, Comparison of the models, and Analysis of the selected model: Xception.

- Web Scraping System for Data Extraction ***(chapter 6)***. This chapter presents the data extraction process using web scraping techniques and classifying results with a deep learning model. Additionally, it contains a previous analysis and database formatting for the case study. Sections included in this chapter are Introduction, Previous analysis, and Methodology and configuration of the data modeling process.

- Food/Restaurant Searcher Web Application ***(chapter 7)***. This chapter presents the architecture, features and target devices of the web application for data visualization as the principal element of the case study and secondary part of the project. Sections included in this chapter are Overview, System architecture, Application layout, and Targeted platforms.

- Case Study — Foodiefy ***(chapter 8)***. This chapter shows the result of a practical case as a web application. It proposes an application concept after analyzing the context. Additionally, each part of the app and the interactions are shown in detail. Sections included in this chapter are Introduction, Features, Device responsiveness, Interaction overview, and Views.

- Conclusions and Future Work ***(chapter 9)***. This chapter details the achieved goals and outcomes done by the master thesis following key points developed in the project. Additionally, there are described the future lines of work and the repository to continue in the same way. Sections included in this chapter are Achieved Goals, Links of interest, Conclusions, and Future lines of work.

- Project Impact ***(Appendix A)***. This appendix shows the social, economic, environmental impact jointly the ethical an professional responsibility. As it is described, the project uses open source datasets and web scraping techniques to acquire data that are the main points to be highlighted. Sections included in this chapter are Social impact, Ethical and professional responsibility, Economic impact, and Environmental impact.

- Project Budget ***(Appendix B)***. This appendix describes the necessary project budget regarding material resources, human fees, and taxes involved. Sections included in this chapter are Material resources, Human fees, and Taxes involved.

CHAPTER 2

State of the Art

This chapter introduces the reader all necessary knowledge to understand the project and its different sections. It deals with the current state of food image recognition systems and the different methodologies to carry it out.

2.1 Food image recognition

Food-related studies propose multiple applications and services to improve health, guide human behavior and understand the culinary culture. The development of social networks has allowed the sharing of lifestyles through images, recipes, and diets, which have led to the collection of large datasets of information. This information grants obtaining the necessary knowledge of how food can help people to solve the current problems of society.

It is demonstrated that the foods of a diet have a direct impact on the health of people. In recent years, the number of people with obesity and overweight has been increasing progressively due to the widespread consumption of unhealthy diets [47]. This condition of lifestyle makes people prone to acquire chronic diseases such as cardiovascular, respiratory and cancer. Therefore, it is completely necessary to build tools to control food habits [48].

Food computing vision acquires and analyzes heterogeneous food data from disparate sources for perception, recognition, retrieval, recommendation, and monitoring of food. It proposes different approaches related to human behavior, medicine, biology, gastronomy, and agronomy. Technological advances together with existing data are transforming the way of analyzing and interpreting food. Therefore, it uses different areas such as network analysis, computer vision, machine learning and data mining to analyze this class of information [45].

2.1.1 Computer Vision

Recently, the computer vision community has focused its objectives on areas involved in the development of automatic systems for food analysis: food image detection, food item recognition, quantity or weight estimation, food localization, portion estimation and caloric and nutritional value assessment [14].

On the one hand, food detection is known as a binary classification problem, where the algorithm tries to distinguish whether an image represents food or not. Classical approaches try to extract features such as interest point descriptors from scale-invariant feature transform (SIFT), pool the features into a vector representation eg, a bag of words and Fisher Vectors and then use a clustering algorithm such as Support Vector Machine (SVM) for classification. Recent studies have shown that the best results obtained are based on Convolutional Neural Network (CNN), in comparison to previous works in conventional machine learning, CNNs produce better performance.

On the other hand, assuming that only one food is present in the image, food recognition, tries to solve the problem of categorical classification. Researchers have been working on food recognition using traditional approaches based on classical image features and machine learning. Investigations of traditional approaches using multiple techniques achieve between

28% and 65% accuracy for datasets of 50-101 classes, depending on the different algorithms used. In recent years, CNNs are also widely used in food recognition since it achieves better performance than conventional methods, achieving accuracy above 79% for datasets of 101 classes [58].

In the last couple of years, machine learning and specifically deep learning with convolutional neural networks have achieved significant technological advances for image classification and recognition tasks. Although they have been working on a food image, there is a long way to find a solution with high accuracy, considering that there are many difficulties due to the great variety and complexity of food. Bearing in mind that many foods are not distinguishable by the human eye because they have a similar color or shape, it is extremely difficult to recognize them correctly. Another real problem is when different foods appear on the same plate, which makes it more difficult to answer this problem. Therefore it is believed that it is sufficient to recognize general types of food, which can provide very useful information for diets, calories, etc [58].

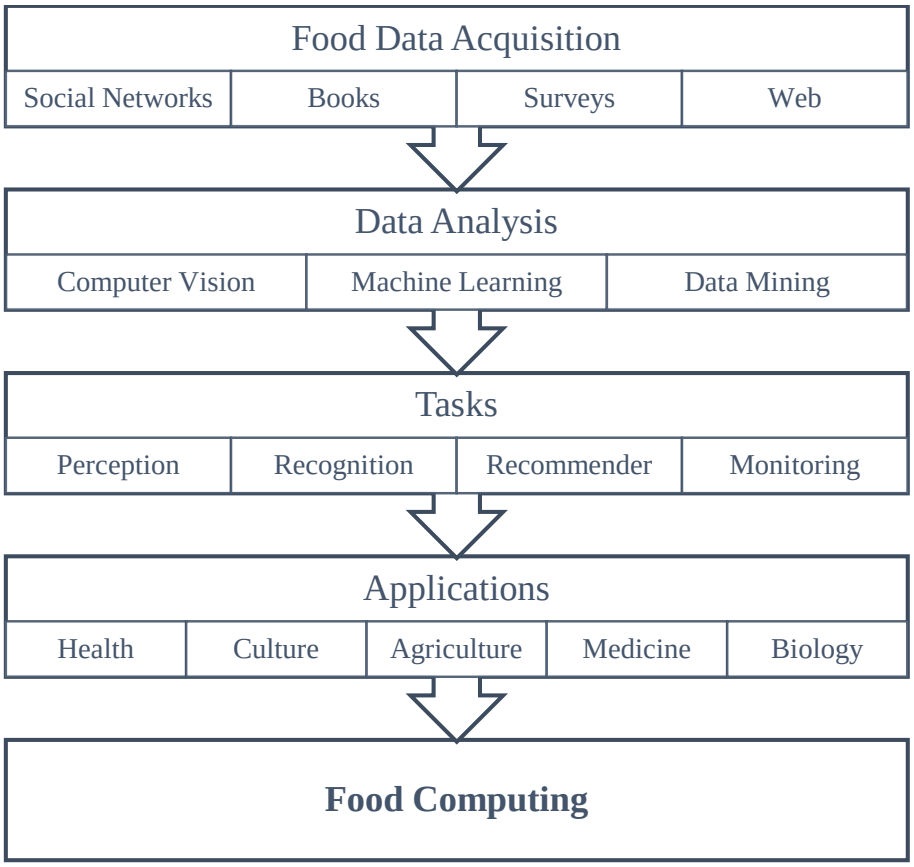


Figure 2.1: Food computer vision diagram

2.1.2 Food datasets

Currently, there are multiple datasets of food images [45] to perform tasks of recognition, quantity estimation, retrieval, detection, segmentation,... Nowadays, benchmark datasets are frequently released for the recognition of food images, whereas, researchers were focused on datasets with fewer classes and smaller scale. For example, UEC Food100 [44] consists of 14,361 images of Japanese food divided into 100 classes. With the rapid development of social networks, it is easier to get large amounts of images of food. For example, the largest dataset, Instagram800K [50] contains 808,964 Instagram food images of 43 classes. However, ETHZ Food-101 [16] is the benchmark dataset chosen for the task of recognizing food images since it has the highest ratio of images and categories. ETHZ Food-101 consists of a dataset of 101 food categories, with 101,000 images. For each class, there are 750 training images with noise and 250 manually cleaned test images.

The following Table shows popular accessible food datasets for recognition task, indicating the dataset name with link, number of images, classes of food, source, and references.

Dataset	Images	Classes	Sources	Ref.
UEC Food100	14,361	100	Web+Manual	[44]
UEC Food256	25,088	256	Crowd-sourcing	[36]
ETHZ Food-101	**101,000**	**101**	**foodspotting.com**	**[16]**
UPMC Food-101	90,840	101	Google Image Search	[64]
Dishes	117,504	3,832	Dianping.com	[67]
Instagram800K	808,964	43	Instagram	[50]
Food524DB	247,636	524	Existing datasets	[23]
ChineseFoodNet	192,000	208	Web	[18]

Table 2.1: Food datasets

2.1.3 Recognition applications

There are multiple applications [45] for image recognition tasks. Chronic diseases, control systems of traditional diets, food diaries, analysis of eating habits and the widespread use of smartphones have been the precursors of these applications. Most applications are focused on four types of algorithms:

- **Label food recognition:** supposes that every image has only one kind of food.
- **Food recognition on multiple labels:** tries to detect and classify multiple labels in one image.
- **Mobile food recognition:** focused on mobile technologies.
- **Specific food recognition:** focused on specialized image recognition for a specific function, like restaurant food recognition.

2.1.4 Related work

The following Table shows recent works of image recognition for ETHZ Food-101 dataset. For each research, it is shown the method name used and the Top-1 and Top-5 accuracy ordered from lowest to highest:

Method	Ref.	Top 1-Acc	Top 5-Acc
AlexNet	[16]	56.40	-
Modified AlexNet	[68]	70.41	-
Modified GoogLeNet	[40]	77.40	93.70
VGG16	[53]	79.17	-
InceptionV3 (stratospark)	[52]	86.97	97.42
InceptionV3	[29]	88.28	96.88
ResNet-200	[31]	88.38	97.85
Wide Residual Network (WRN)	[43]	88.72	97.92
Wide-slice Residual (WISeR)	[43]	90.27	98.71

Table 2.2: Related works accuracy for ETHZ Food-101

All investigations are based on pre-trained models, using fine-tuning to get new models, it can be deduced that using them is the best way to obtain better results. In addition, some of them use different techniques such as standard 10-crop and wide-slice residual technique to improve the accuracy of the models.

As a critic, note that all show the results obtained in articles but none of them makes their code accessible except Stratospark [52], which creates a completely transparent work.

2.2 Deep Learning for image analysis

2.2.1 Artificial Neural Networks

Artificial Neural Networks (ANN) [62] are systems modeled in biological neural networks that trying to simulate the behavior of human brain systems. The neural network is not regarded as an algorithm, but as a framework to process multiple algorithms working together.

The human brain is composed of neurons interconnected by dendrites receiving input signals to transform them into electrical signals that are transmitted by the axon. For computers, it is easier to process calculations with a great need for processing, while humans have the ability to recognize shapes and colors easily. Therefore, the focus is on solving problems benefiting both advantages [30].

Such systems learn through examples regardless of any specific feature. For example, to identify an object of an image you do not need to know the characteristics of it, but it automatically generates features by processing them.

The origin of artificial neural networks is due to several related studies [41]. In 1943 neurophysiologist Warren McCulloch and mathematician Walter Pitts conducted a study of the functioning of neurons. In 1949, Donald Hebb created the learning hypothesis based on neural plasticity, better known as Hebbian learning. In 1962, Widrow & Hoff developed a procedure based on the idea of the perceptron, while an active perceptron can have a big mistake, you can adjust the weight to distribute it over the network. Finally, neural network research was stalled in 1969 from machine learning by Marvin Minsky and Seymour Papert.

2.2.1.1 Network functionality

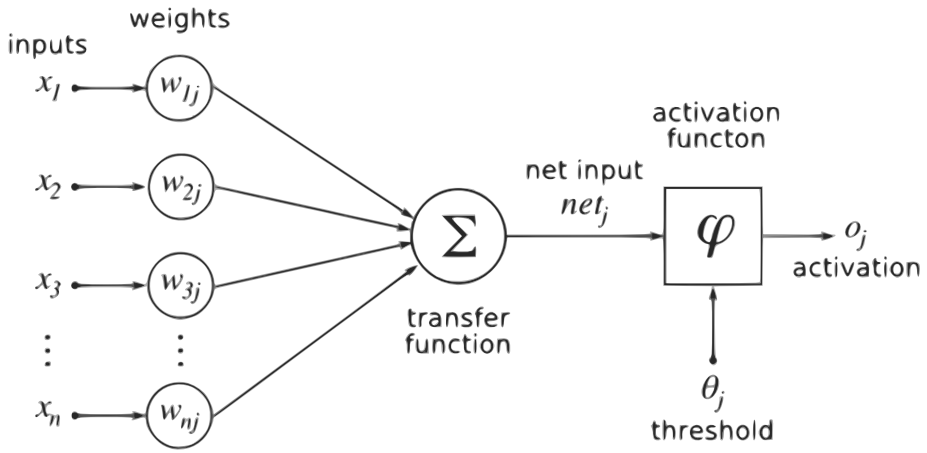


Figure 2.2: ANN functional interpretation extracted from [17]

An ANN is represented as connected nodes or artificial neurons that resemble that of the human brain. Each neuron can transmit a signal to another through the connections or edges. Both the nodes and the edges have a weight that adjusts the learning and also the first ones have to overcome a threshold signal to activate. In short, all these elements form a directed weighted graph formed by different hidden layers.

2.2.1.2 Component features

For making decision on whether or not the signal passes from one neuron to another activation functions are used. Among the most important are Sigmoid function, which transforms the output signal into a value between 0 and 1, Tanh function that transforms the output signal into a value between 1 and -1, ReLU function transforms the output signal into a positive value or 0 and finally Softmax function, which works in a similar way as Sigmoid but divides each of its outputs probabilistically, very useful for classifiers.

To minimize the cost function there are various algorithms to measure accuracy. Among the most popular are mean squared error and cross-entropy. Adjusting weights in the training phase in two phases, forward and backward, is performed by a series of techniques called optimizer. Among the most outstanding are the Stochastic gradient descent, as a variant of gradient descent, RMSprop, Adam, as a generalization of Adagrad. The most used for training images are Adam and SGD for its rapid convergence [28].

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs / ConvNets) [35, 72] are very similar to Artificial Neural Networks explained previously, based on neurons with weights and biases. They are conceived as a class of deep neural networks that is normally applied to image analysis.

CNNs are inspired by biological processes simulating the pattern of connectivity between neurons to simulate the visual cortex. It uses a variation of multilayer perceptrons that need preprocessing little capacity compared to other image classification algorithms. Unlike traditional ANNs, ConvNet starts from the fact that the input signals are images, which allows processing some properties in the architecture, achieving a more efficient forward function and reducing the number of parameters of the network.

Among the most recent applications in which this type of architecture is used, there are image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing.

The basics of CNNs are on the neocognitron introduced by Kunihiko Fukushima in 1980 and later improved by Yann LeCun in 1998 by introducing the method of learning backward. From 2012 it was refined by performing deployments graphics processing units

(GPU).

2.2.2.1 Comparing architectures

Recapitulating, like the ANNs, as input signal it has a vector that would be transformed by hidden layers series. Neurons of a hidden layer interconnect with all neurons in the preceding layer and with none of the same level, reaching a final layer or output layer representing scores.

On the one hand, the problem of traditional ANN for image analysis is that as an input signal has a "wide x high x channels" image that for a 200x200x3 image would have in a single fully-connected neuron in the first hidden layer 120,000 weights. Therefore this type of connectivity is wasteful and would produce overfitting. The following Figure shows the architecture:

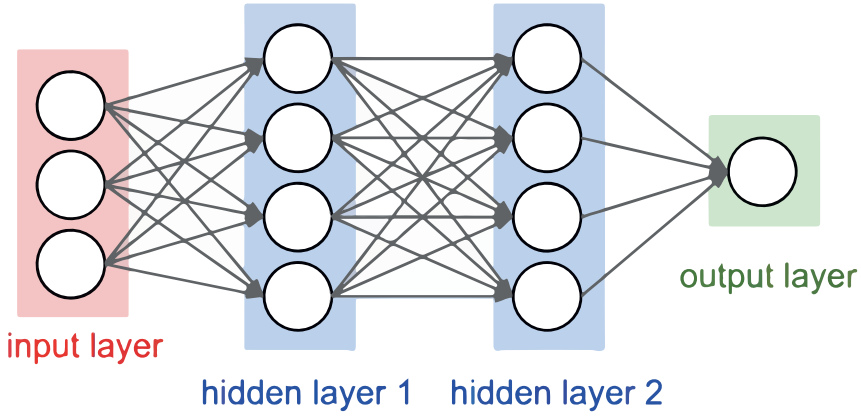


Figure 2.3: ANN layer architecture extracted from [35]

On the other hand, CNNs based on the premise that the input signals are images and layers containing neurons are organized in 3 dimensions: width, height, and depth. It transforms an input 3D volume to an output 3D volume with some differentiable function that may or may not have parameters. The following Figure shows the architecture:

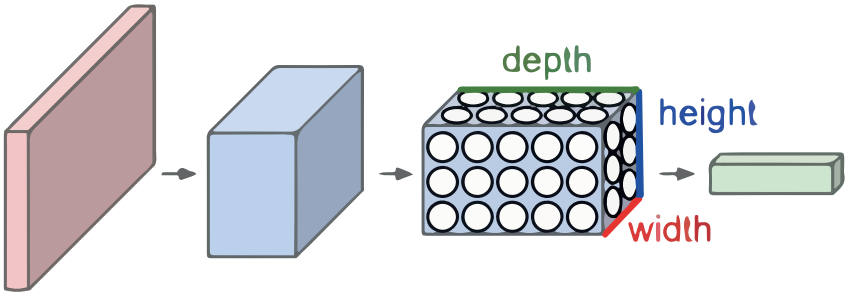


Figure 2.4: CNN layer architecture extracted from [35]

2.2.2.2 Architectural layers

CNNs consist of input, hidden and output layers [65]. Hidden layers are composed of convolutional layers, ReLU layers, pooling layers, fully-connected layers, and normalization layers.

- **Convolutional Layer:** They are the central building block of CNN and where the greatest computational expense occurs. They apply the convolution operation to the input and pass the result to the next layer. Every entry can be interpreted as an output that only take into account one small region. When is needed to deal with high dimensional data like images is not effective to connect neurons to every neuron in the next layer so it the strategy consist in connecting a subregion of the input volume.

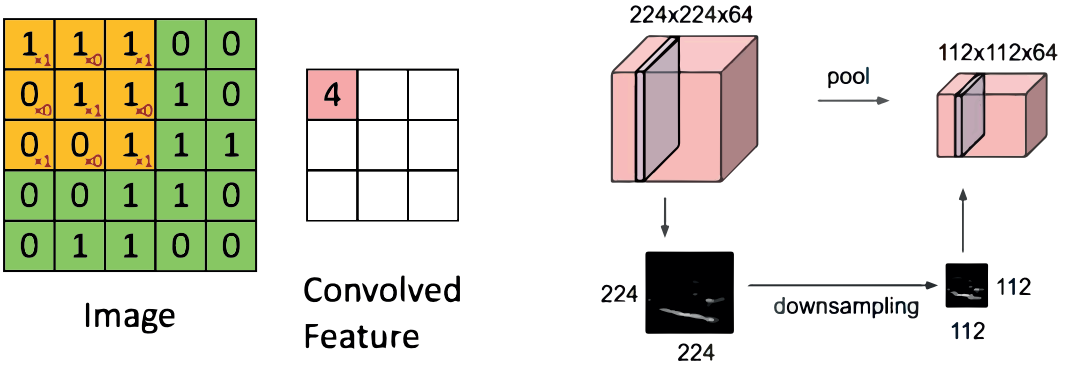
  Parameters are a set of filters (matrix) with learning capacity, each of these small filters (size defined by receptive field) extend through the entire depth of the input volume. Therefore, when performing the convolution of each of the filters, it results in a matrix called Feature Map which is used for edge detection, blur detection, and sharpen detection.

  Three hyperparameters are used to explain the number and organization of neurons to control the size of the output volume. The depth of the output volume is a hyperparameter that corresponds to the number of filters that will be used. The stride refers to the number of pixels that is filtered. Finally, zero-padding refers to the size of the padding applied around the edge.

- **Pooling layer:** Commonly, pooling layers are inserted after a convolutional layers, whose function is to reduce the spatial size of the representation progressively and thus reduce the number of parameters and avoid overfitting.

  It takes a filter and a stride of the same length. Then, it applies to the input volume and outputs the number depending on the technique in every subregion that the filter convolves around. The two most used techniques are max pooling that selects the maximum value of the cluster and average pooling that takes the average value of the cluster.

  The following Figures shows the difference between convolution and pooling operation and resulting effect:



(a) Convolutional layer operation

(b) Pooling layer effect

Figure 2.5: Convolutional and Pooling functionalities extracted from [35]

Figure (a) shows convolution operation over the image before slide the filter for the next value and indicating the first number result into the output matrix. Meanwhile, Figure (b) shows the effect of the layer, pooling and down-sampling the image by half.

- **Normalization layer:** They have developed various types of normalization layers trying to simulate the brain thread behavior, however, for ConvNets has been shown to produce no signifiable improvement and are leaving to use. Its function is to normalize the activation of the previous layers in each batch.

- **Fully-connected layer:** Connects each neuron in one layer to all neurons in another layer, resembling a multi-layer perceptron neural network (MLP). For classification algorithms, Softmax function is usually used.

The next Figure shows a diagram of a standard convolutional neuronal network with all the layers used [69]. It shows an input image that passes through two stages of extraction of features composed of a convolutional layer and pooling layer. Finally, there is the Fully-connected layer that connects all the extracted features and provides the classification probabilities leaning on Softmax or a similar function.

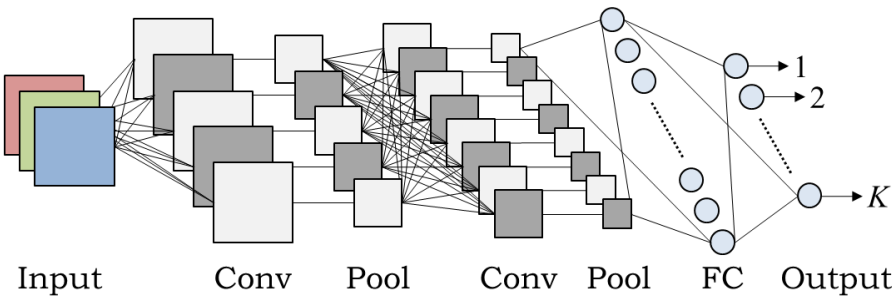


Figure 2.6: Layer diagram of CNN extracted from [32]

2.2.3 Predefined historical architectures

Creating convolutional neural network architectures can become very complex depending on the needs of our dataset classification. Therefore, there are predefined ConvNets that serve different types of image classification. These architectures are frequently accompanied by weights trained with a dataset, enabling direct use for prediction, feature extraction and fine-tuning.

ConvNet existing architectures [35, 34, 25, 24] can be divided into classic and modern. Among the classic highlight LeNet-5, AlexNet, and VGGNet, while modern would GoogLeNet, ResNet, and DenseNet.

Congress ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [54] has been key to the development of ConvNets. It is an annual competition organized since 2010 where a number of research teams test their algorithms to classify and locate objects using ImageNet dataset.

- **LeNet-5 [39]:** This model, the pioneer of the ConvNets, was developed by Yann Lecun in the 1990s with the aim of identifying zip codes for the postal service.

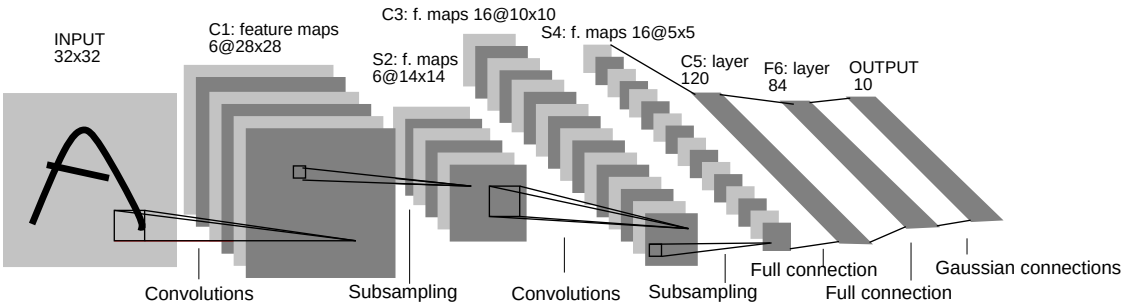


Figure 2.7: LeNet-5 architecture extracted from [39]

The architecture consists of two stages of convolutional and average pooling layers, followed by a flattening convolutional layer to end in two fully-connected layers and a softmax classifier. The number of parameters of this network is 60,000.

- **AlexNet [38]:** This model developed by Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton, popularized CNN in computer vision in the ILSVRC 2012 contest of ImageNet and convinced the computer vision community to use deep learning.

  The key points to win the competition were using ReLU layers, data augmentation techniques, dropout layers, and train the model using batch stochastic gradient descent on two GTX 580 GPUs for five to six days.

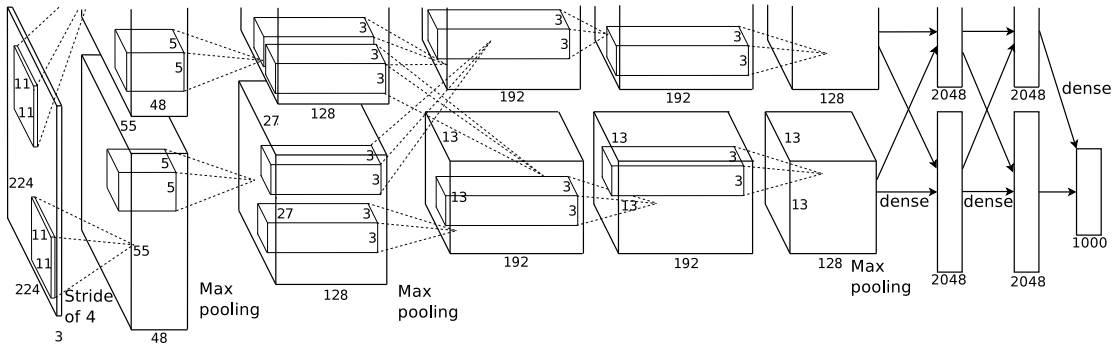


Figure 2.8: AlexNet architecture extracted from [38]

AlexNet architecture is similar to LeNet-5 but much larger and deeper. The network was made up of 5 convolutional layers, max-pooling layers, dropout layers, and 3 fully connected layers. The number of parameters of this network is 60 million.

- **ZFNet [71]:** ILSVRC 2013 winners were Matthew Zeiler and Rob Fergus with ZFNet. This network is very similar to AlexNet making changes in hyperparameter, specifically expanded the size of the convolutional layers in the middle and using smaller stride and filter.

  The key to winning this competition, in addition to modifications AlexNet points, was the use of ReLU layers to trigger functions, Crossy-entropy loss for the function error, the development of a visualization technique (Deconvolutional Network) and train the model using batch stochastic gradient descent on two GTX 580 GPUs for twelve days.

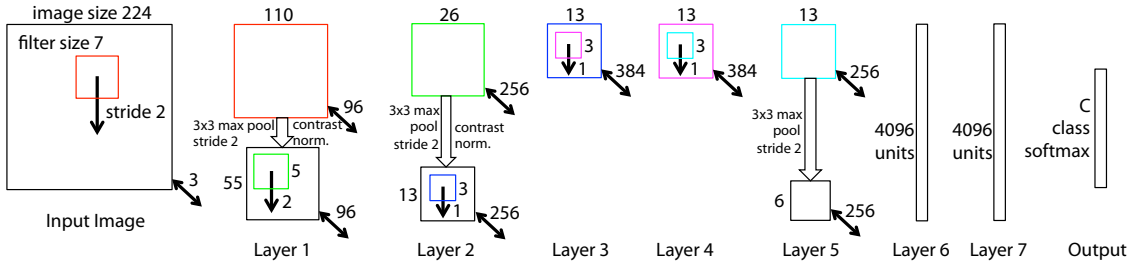


Figure 2.9: ZF-Net architecture extracted from [71]

- **VGGNet [57]:** The ILSVRC 2014 was not won by VGGNet, but Karen Simonyan and Andrew Zisserman did a great job of proving that the depth of the network is a critical component for good performance. The network is composed of 16 CONV/FC layers using 3x3 filters, along with 2x2 maxpooling layers. The number of parameters of this network is 138 million.

  The key points of this model were, besides those already mentioned, the use of scale jittering as data augmentation technique, Relu layers, and train the model using batch gradient descent on Titan Black four Nvidia GPUs for two to three weeks.

Figure 2.10: VGG-16 Net architecture

- **GoogLeNet (Inception) [60]:** In 2014, Google created this winning network ILSVRC 2014. His greatest contribution was the reduction of the number of parameters to 4 million in its first version since the v3 has 23 million parameters. It also uses average pooling instead of layers fully-connected at the top of the net. Although the architecture consists of 22 layers, the number of parameters is considerably small.

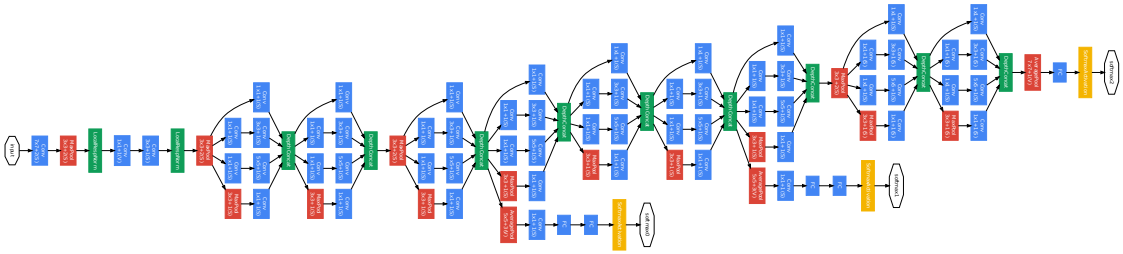


Figure 2.11: InceptionV1 architecture extracted from [60]

In the previous Figure, convolution layers are shown in blue, max-pooling layers in red, in yellow softmax, in green concatenation channels, in purple fully-connected layers, and in orange normalization layers.

- **ResNet [31]:** The architecture developed by Microsoft was the winner of ILSVRC 2015. It has become the default choice for using ConvNets. These networks are characterized by being much deeper reaching the order of hundreds of layers. To solve the problem of degradation the concept of residual blocks, which addresses the idea that an input (x) go through conv-relu-conv series resulting F(x), however, passes to the next block H(x) = F(x) + x instead of F(x). With this technique, the number of parameters that have ResNet50 is 25 million.

  Its main features were skipping connections, heavy use of batch normalization, remove the fully-connected layers at the end of the network, using ultra-deep network 152 layers, and stress that was trained on an 8 GPU machine for two to three weeks

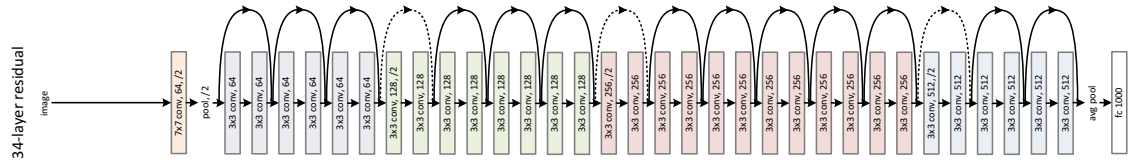


Figure 2.12: ResNet34 architecture extracted from [31]

Each group of layers of the same color represents the same dimension convolutions. Dotted lines represent residual connections in which 1x1 convolutions are used to adjust the size of the new block.

In this research, with wide residual networks, it was reached the conclusion that increasing network bandwidth (channel depth) can be more efficient than expanding the overall network capacity measure. As variant ResNet subsequently emerged ResNetXt with good evaluation in ILSVRC 2016. This architecture replaces the standard residual blocks by opponent's strategy "split-transform-merge" used in Inception.

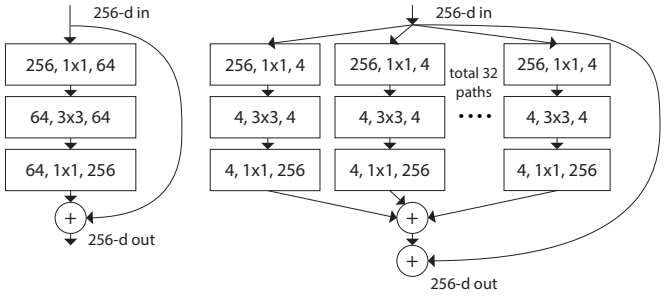


Figure 2.13: ResNet, ResNetXt block comparison extracted from [66]

The Figure shows a ResNet block on the left and on the right a ResNetXt block with almost the same complexity.

- **DenseNet [33]:** This architecture promotes a simple idea "it may be useful to refer to the maps from earlier in the network". Therefore, each successive layer inside the dense block is concatenated at the entrance. DenseNet varies between 0.8 and 40 million parameters according to its version.

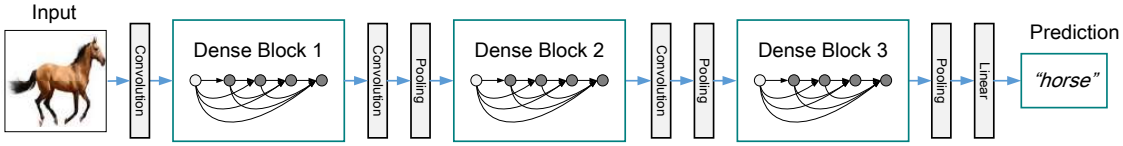


Figure 2.14: DenseNet architecture extracted from [33]

This architecture mimics ResNet but replacing the residual blocks for dense blocks, which enables the quantity to subsequent layers in the network take the above characteristics.

2.2.4 Transfer learning and fine-tuning

It creates a model for image classification can be very complicated. It is also possible that the tools and elements necessary for its realization are not available. However, there are techniques to transfer the knowledge of pre-trained models to your needs.

Transfer learning [42, 46, 55] is a computer vision method that allows creating accurate classification models in a short time. Instead of starting the training process from scratch you start with a series of patterns. This requires the use of pre-trained models trained on a large dataset and so to solve a similar problem.

It is assumed that the convolutional neural networks are composed of a first part for generating image features (convolutional base) and a second part serving classifier and provides probabilities (classifier). Furthermore, in the convolutional base, the lower layers generally refer to features (independent problem), while higher layers refer to specific features (dependent problem). Then, if you have a pre-trained model, you can take advantage of features obtained from the convolutional base to apply our classifier training entire or part of the model.

2.2.4.1 Fine-tune strategies

Depending on the type of training applied to pre-trained model there are three strategies. This technique of fully or partially training convolutional base is known as fine-tuning.

- **Entire model training:** This strategy is to train all the layers of the pre-trained model. Therefore, a large dataset and computational capacity are needed.

- **Partial model training:** As it has been seen previously, the lower layers and the higher layers contain information on general and specific features, respectively. Therefore, this technique consists of partially freezing some layers of the convolutional base and training the rest. This method is very effective if you do not have a large dataset or too much computing power.

- **Freeze entire convolutional base:** This technique consists in training the model freezing all layers of the convolutional base. It is effective if large computing capacity is not available, the dataset is small and the pre-trained model solves a similar problem.

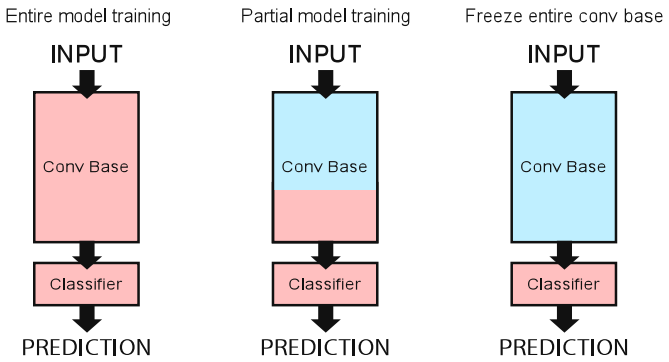


Figure 2.15: Fine-tuning strategies

Both the first and second strategy must be careful with the learning rate chosen because it can distort the weights and therefore it is advisable to use a small one.

The choice of fine-tuning strategy depends on two factors, the size of the dataset and the similarity with the pre-trained dataset. If you have a large dataset, when it is related or not with the trained dataset leads to training the model partially or whole respectively. While if the dataset is small, when it relates to the trained dataset or not, it is advisable to train the model partially in the first case and train the frozen model the entire convolutional base for the second case.

2.2.5 Current Keras applications

Existing and available on Keras predefined models are rated according to their accuracy, size and depth for ImageNet dataset with 1.2 million images belonging to 1000 classes. Among the models available are VGGNet, ResNet, Inception, MobileNet, DenseNet and NASNet plus all its variants.

The following Table has the most popular Keras architectures with accuracy, size and depth. The top-1 and top-5 accuracy refers to the model's performance on the ImageNet validation dataset.

Model	Size	Top 1-Acc	Top 5-Acc	Parameters	Depth
Xception	**88 MB**	**0.790**	**0.945**	**22,910,480**	**126**
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	**98 MB**	**0.749**	**0.921**	**25,636,712**	**-**
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
ResNeXt50	96 MB	0.777	0.938	25,097,128	-
ResNeXt101	170 MB	0.787	0.943	44,315,560	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	**215 MB**	**0.803**	**0.953**	**55,873,736**	**572**
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	**14 MB**	**0.713**	**0.901**	**3,538,984**	**88**
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

Table 2.3: Pre-trained models available in Keras with ImageNet weights extracted from [19]

CHAPTER 3

Enabling Technologies

This chapter offers a brief review of the main technologies that have made possible this project, as well as some of the related published works.

3.1 Machine Learning technologies

3.1.1 TensorFlow

TensorFlow (TF) [9, 12] is an open source software library for high-performance numerical data processing that uses dataflow graphs to represent computation, shared state and mutational operations of that state.

Originally developed by researchers and engineers from the Google Brain team within Google's AI organization was released under the Apache 2.0 open-source license on November 9, 2015. First version 1.0.0 came out on February 11, 2017, and last stable version 1.12.0 was released on November 5, 2018.

This software library operates at large scale and in heterogeneous environments. Its flexible architecture allows easy deployment of computation across a variety of platforms including multicore CPUs, general purpose GPUs and custom-designed ASICs known as Tensor Processing Units (TPUs). Therefore, it can work on devices from desktops to clusters of servers to mobile and edge devices. There are many APIs already included in TensorFlow, for example, TensorBoard, which is a suite of visualization tools in order to make it easier to understand, debug, and optimize TensorFlow programs. A large community of users and an infinity of project resources, implementations and research provide the stability and reliability of the library.

TensorFlow supports a variety of applications focused on training and inference on deep neural networks. Several Google services use TensorFlow in production, it was released as an open-source project, and it has become commonly used for machine learning/deep learning research. Some of the current uses of Google are Deep Speech, RankBrain, Inception Image Classification Model, SmartReply, Massively Multitask Networks for Drug Discovery and On-Device Computer Vision for OCR.

3.1.2 Keras

Keras [21] is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. The main factor is being able to go from idea to result with the least possible delay that is the key to doing good research.

This API was developed by project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) as part of the research effort, and its leading author and maintainer is François Chollet, who is a Google engineer. The first version came out on March 27, 2015, and last stable version 2.2.4 was released on October 3, 2018.

Keras is a user-friendliness API designed for human beings, not machines. It focuses on the user experience and follows best practices for reducing learning effort: it offers consistent y simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.

Its modularity shows a model understood as a sequence or a graph of standalone, fully configurable modules that can be plugged together with as few restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standard modules that can be combined to create new models. As it is easily extensible, new modules can be added in an easy way (as new classes and functions), and existing modules provide extensive examples.

It works with Python (2.7-3.6). No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

Actually, it claims over 250,000 users as of mid-2018. Keras is the favorite tool for deep learning researchers, coming in #2 in terms of mentions in scientific papers uploaded to the preprint server arXiv.org. It has also been adopted by researchers at large scientific organizations, in particular, CERN and NASA.

3.1.3 Colaboratory

Colaboratory [2] is a free Jupyter notebook environment that runs in the cloud and stores its notebooks on Google Drive and GitHub.

Following the launch of its Artificial Intelligence opensource library, Tensorflow, launched Colaboratory [27], a internal development tool completely free. The software works in a similar way as Google Docs as a collaboration tool but also allows you to run the code and see the answers.

Google's strategy to offer this service for free, as with TensorFlow, is to achieve standardization of software to teach students Data Science. Thus Colaboratory is focused as an educational tool for collaborative research.

Although the main feature of Colab is running in the cloud, allows the execution in a local environment. The environment allows code execution in python2 and python3, also comes preinstalled with the TensorFlow library and more.

The environment has a dedicated CPU per user for computing, Intel (R) Xeon (R) CPU @ 2.30GHz, with 12.72 GB of RAM and HDD 358.27 GB. In addition, it comes with GPU Tesla K80 with 12 GB GDDR5, useful to expedite TensorFlow training complex models faster.

3.1.4 Scikit-Learn

Scikit-Learn [7] is an open-source library built on NumPy, SciPy, and matplotlib, for machine learning on Python. It includes simple and efficient tools for data mining and data analysis like classification, regression and group analysis algorithms.

The project initially called "scikits.learn" was created by David Cournapeau as a Google Summer of Code project. Subsequently, INRIA assumed the leadership of the project and made the first distribution in 2010.

3.2 Data technologies

3.2.1 Pandas

Pandas [5] is a Python open-source library that provides high-performance, easy to use data arrangements and data analysis tools. This library, extension of NumPy, was released under the three-clause BSD license.The name is derived from "panel data" which is an econometrics term for naming datasets including multiple time periods observations.

The principal features and functionalities that provide pandas library for data manipulation are:

- Data manipulation with indexing on DataFrame object.
- Reading and writing tools in different file formats.
- Data alignment, handling missing data, reshaping and pivoting.
- Split, apply, combine, filter, merging and joining operations.

3.2.2 BeautifulSoup

BeautifulSoup [1] is a Python library designed as a quick analyze project for parsing XML or HTML documents and based on top parsers like lxml and html5lib.

This library implements methods for navigating, searching and modifying a parse tree, so this tool is useful for web scraping. Automatically transforms incoming documents to Unicode and outgoing documents to UTF-8.

3.2.3 JSON

JSON (JavaScript Object Notation) [4] is an open-standard file format that eases human reading and writing and machines parsing and generating. It is a lightweight data-interchange format based on a subset of JavaScript.

3.2.4 Selenium

Selenium [8] is a browser automation framework for the testing of applications. It is provided with tools to record/reproduce tests without using a test scripting language (Selenium IDE), driving a web browser natively (Selenium WebDriver) and controlling web browsers locally or in other computers (Selenium Remote Control / Grid). The different tests run on the web browsers desired for Windows, Linux and macOS platforms. It is an open-source software developed under the Apache 2.0 license.

The framework was developed in 2004 by Jason Huggins who was joined by other experts in the field. The name comes from its competitor Mercury as a joke since it was said that the poisoning caused by mercury could be healed by taking selenium supplements.

Selenium WebDriver does not require any special server to run the tests, the WebDriver creates instances of the browser and controls it. This software is fully implemented and supports Java Ruby, Python, and C#. The best uses and applications of Selenium are automated testing, scraping and crawling.

3.3 Web application technologies

3.3.1 W3.CSS

W3.CSS [10] is free to use and easy to learn modern CSS framework with built-in responsiveness. It is based on mobile-first design and it is smaller and faster than similar CSS frameworks.

The main interest between other frameworks is that fits all browsers, devices and applies CSS only (no jQuery or JavaScript). This framework was designed by Jan Egil Refsnes with w3schools support.

3.3.2 FontAwesome

FontAwesome [3] is a freemium font and icon toolkit based on CSS and LESS. This library was designed by Dave Gandy for Twitter Bootstrap and later implemented in BootstrapCDN. Nowadays has a team behind it and is one of the top font libraries only behind Google Fonts.

3.3.3 Fuse.js

Fuse.js [51] is an open-source lightweight fuzzy-search library for JavaScript. It allows to search inside JSON files with multiple configuration parameters and scoring results by similarity. This library works on native JavaScript, so it does not need any dependency.

3.3.4 PhoneGap

PhoneGap [6] is a framework for mobile application developing created by Nitobi and succeeding bought by Adobe Systems. PhoneGap allows developing hybrid applications for iOS, Android, and Windows using JavaScript, HTML5, and CCS3, solving multiple platforms problem. The interfaces are not native and do not use specific system graphic interfaces, instead of that, it renders the app with web views.

Apache Cordova includes PhoneGap as a distribution. Both systems have similar functionalities, but PhoneGap has a compile access in the cloud by Adobe. Apache Cordoba is an open-source project and with PhoneGap can be used without licenses.

3.3.5 Docker

Docker [11] is an open-source project to automatize application deployment inside software containers, providing an additional abstraction frame and application virtualization multi-device.

A Docker container image is a standalone, lightweight, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries, and settings. Container images can be built with docker build command, to become containers when they are executed with docker run command on Docker Engine.

CHAPTER 4

General System Structure

This episode summarizes the next chapters showing the general structure of the system. Modules, components, and connections are described and analyzed.

4.1 Architecture

The general system structure is composed of main and auxiliary architectural modules. Next chapters are assumed as main modules. In spite of auxiliary modules are described in the preceding chapters too, they are apart from them. The system architecture is composed of three main modules and three auxiliary modules.

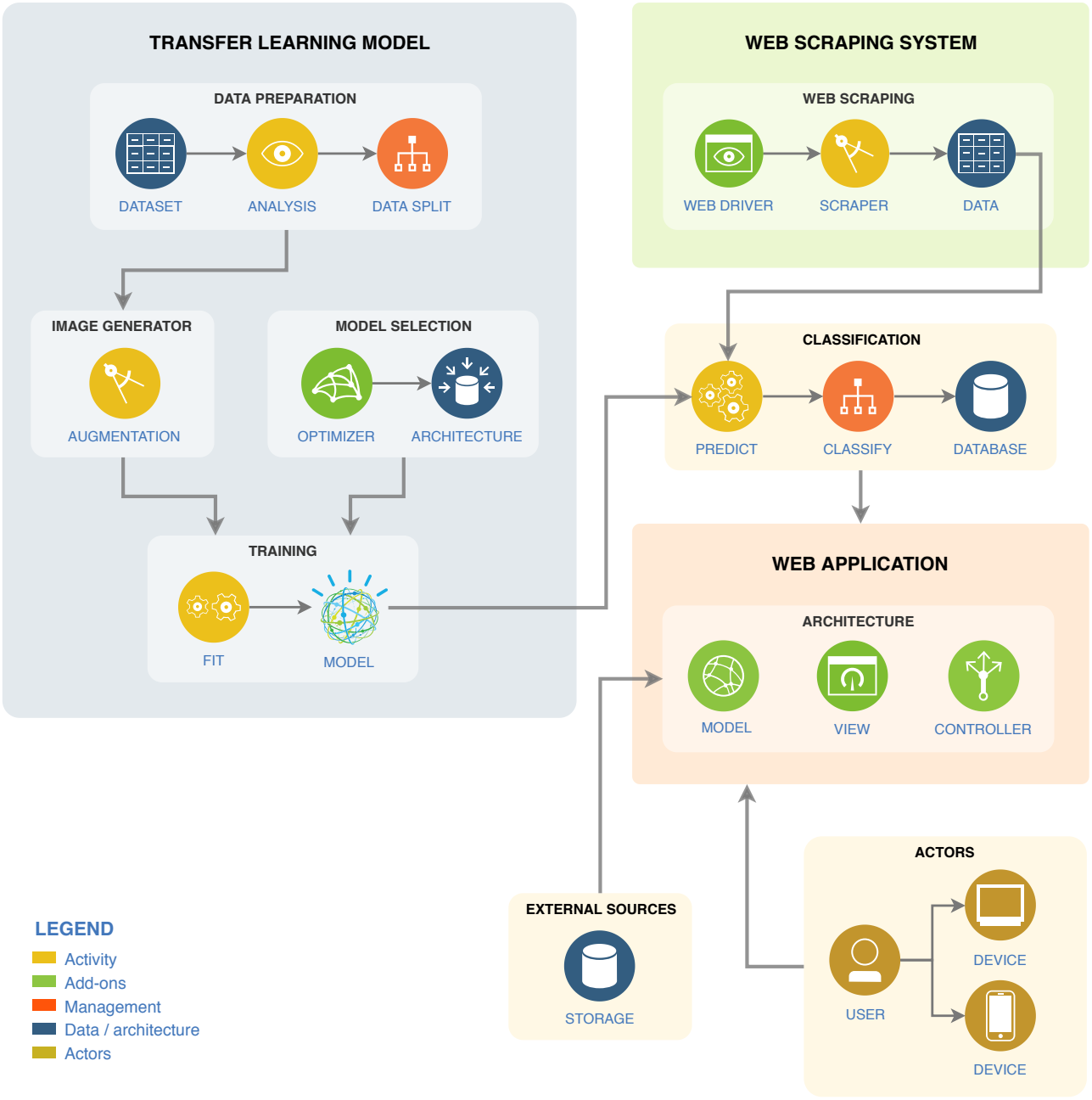


Figure 4.1: General system architecture diagram

The preceding architectural diagram represents the general system structure. It shows main and auxiliary modules defined by groups with specific functional colored details. Each color represents a functionality as can be seen in the legend.

4.2 Main modules

4.2.1 Transfer learning model

This module (chapter 5) prepares and generates data for training an image classification model using transfer learning of a pre-trained architecture. The resulting model is used in the classification auxiliary module.

4.2.2 Web scraping system

This module (chapter 6) uses a web driver for web scraping data of a website. Obtained data is used to generate a database in the auxiliary classification module.

4.2.3 Web application

Web application module (chapter 7) is composed of architectural components and uses the resulting database generated by the classification auxiliary module and external sources.

4.3 Auxiliary modules

4.3.1 Classification

Classification module (subsection 6.3.5) employs the generated model to recognize every image class of the web scraped data and builds a database for the web application module.

4.3.2 External Sources

External source module (subsection 7.2.1) refers to an external host of images used in the web application.

4.3.3 Devices

Devices module (subsection 7.2.2) define actors, devices and platforms that interact with the web application.

CHAPTER 5

Transfer Learning Model for Image Recognition

This chapter shows how to build a image classifier model transferring learning and analyzes multiple deep learning architectures. It describes the overall process of fine-tuning a model, with specific analysis of each architecture and comparison of results.

5.1 Introduction

This section will analyze and explain the methodology of creating an image classification model using deep learning methods of transfer learning.

Firstly, model building process and configurations are explained step by step, specifically applied to the ETHZ Food-101 image dataset, although it could be extrapolated to any other image dataset. Secondly, the architecture and features of the pre-trained models used to transfer knowledge are presented to our classifier. Thirdly, is carried out with the experimental phase the analysis of the pre-trained models used for transfer learning. Finally, the results obtained are analyzed and the optimal model for the project is chosen.

5.2 Methodology and configuration of the model building process

As a summary, the steps followed to build the model are shown in the following Figure:

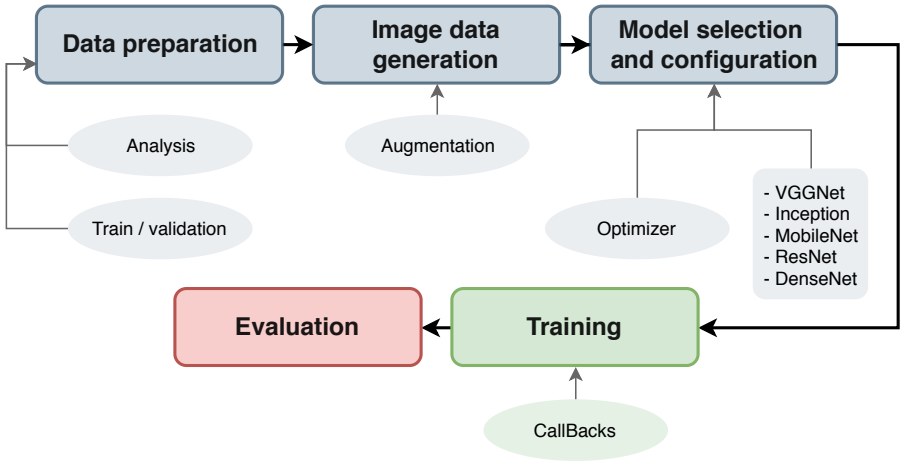


Figure 5.1: Model building process methodology and configuration schema

5.2.1 Data preparation

First of all, the content of the dataset used must be analyzed. In this case, ETHZ Food-101 [16] contains 101,000 images belonging equally to 101 categories, so there are 1,000 images per class. As a dataset competition, indicates for each class the images used for training and validation, being a ratio of 75% for training and 25% for validation. Training images present certain noise on purpose, and maximum image size is 512 pixels.

Having mislabeled images that are not even food, a blacklist [15] is used to improve results. This is intended to prevent images with high probability that do not correspond to the category being recognized. For example, ticket images are categorized into hamburgers, because there are training images of a ticket in that category.

Due to the nature of the dataset and the challenge, it is not necessary to divide the dataset into train, validation, and test, since the source images of train and validation are different. Additionally, fine-tuning this large dataset and image augmentation should not produce overfitting. Thus it is able to compare results with other implementations.

Starting from this information, it proceeds to divide the dataset into a directory for training images and other images for validation, being as follows:

Figure 5.2: Dataset directories split in train and validation

5.2.2 Image data generation with augmentation and pre-processing

A convolutional neural network must be invariant against different object orientations, which implies image augmentation for its training [20, 49]. This option is good for both when you have little data, and when you have a large dataset. On the one hand, more images are obtained for the training. On the other hand, more relevant data is obtained.

There are two methods of data augmentation, offline in which the dataset is increased with augmented images, and online where augmented images are generated during training. For small datasets, the offline mode is advisable, while for large datasets the online mode is recommended. When using a transfer learning method, the same image preprocessing function with which are obtained the weights of the pre-trained model must be used.

Image augmentation techniques used to train the models in this project are:

- **Rotate** the image between a random range of degrees.
- **Shift** or translate the image.
- **Shear** the image indicating a random range of degrees.
- **Scale** or zoom the image randomly.
- **Flip** horizontal or vertical the image.
- **Fill** or interpolate empty image parts using techniques like “constant”, “nearest”, “reflect” or “wrap”.

Once the dataset is separated into training and validation sets, it proceeds to configure the training image generator with preprocessing and augmentation of images. The validation image generator is advisable to not use data augmentation since different values of accuracy and loss would be obtained in each epoch and what is wanted is to evaluate the model with images that serve to check if training results improve or get overfitted. This statement invites us to think that it would train by producing overfitting, but when training the network with augmented images in each epoch is difficult to produce that effect.

The training image generator includes image augmentation techniques while the validation image generator only includes the preprocessing of the image. The following code fragment includes the configuration of the "ImageDataGenerator" class used to generate augmented training images, while the validation generator only preprocesses images.

```
datagen_train = ImageDataGenerator (
  rotation_range = 40 , #Rotate
  width_shift_range = 0.2 , #Shift
  height_shift_range = 0.2 , #Shift
  shear_range = 0.2 , #Shear
  zoom_range = 0.2 , #Scale
  horizontal_flip = True , #Flip
  fill_mode = 'nearest' , #Interpolation
  preprocessing_function = preprocess_input )
datagen_validation = ImageDataGenerator (
  preprocessing_function = preprocess_input )
```

This configuration mixes all defined techniques of image augmentation and randomly generates images like the ones shown in the Figure below (original image on the left, augmented images on the right):

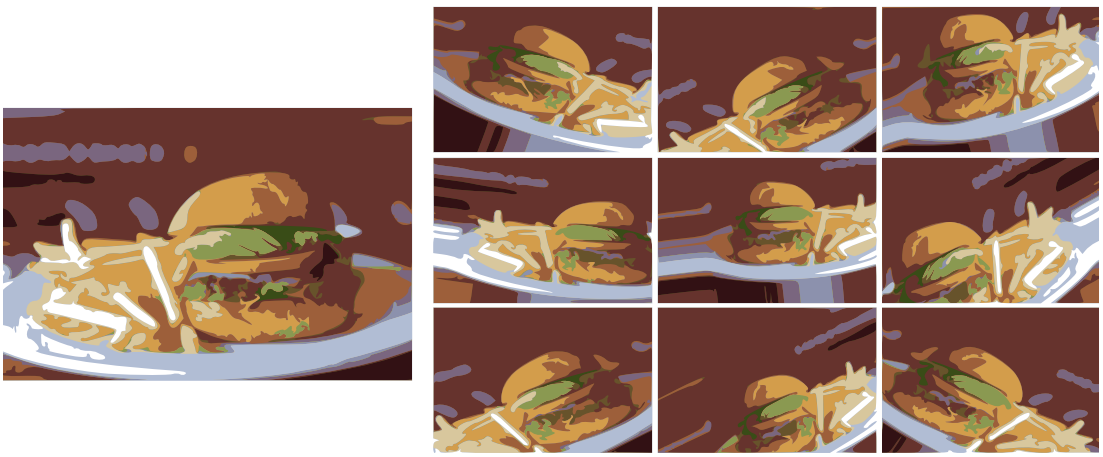


Figure 5.3: Augmentated images example with defined configuration

To generate the augmented online training images, a reader is created from the "ImageDataGenerator" class that obtains the labeled images from the directory. This generator applies augmentation and resizes them to the indicated value which will be the size of the images with which the model has been pre-trained. The generator must also define a hyperparameter called batch size.

Batch size indicates the number of training samples present in a single batch to be spread over the network at each training iteration. These samples are processed in parallel batch independently. It is usually better to have a greater number of images in batch in each training iteration to improve learning quality and convergence rate. However, it is not always possible since the available memory in the hardware is a limiting factor [37].

Therefore, larger batch sizes tend to converge faster and produce better performance. A larger batch size should improve optimization steps and obtain greater convergence speed. In addition, performance can be improved by reducing the number of communications between the GPU. It has been chosen batch size 16, 32 or 64.

5.2.3 Architecture selection and configuration

Once configured the data generators of training and validation, the pre-trained model loaded with the weights of the ImageNet dataset is selected. Keras offers several possibilities such as Inception, ResNet, VGGNet, DenseNet... to choose the most appropriate depending on the dataset and computational capacity.

After selecting the architecture, the convolutional base of the model is loaded without including the part of the classifier to be able to adjust it to the desired requirements. Subsequently, the classifier composed of pooling layers and dense layers is created, with the option of adding a dropout layer. Dropout layer will be necessary when the dataset with which it goes to fine-tuned is small. Dense layer indicates the number of classes to classify and the activation function used. This project uses the following configuration:

```
#Select model and get convolutional base
base_model = <model>(
  weights = 'imagenet' ,
  include_top = False ,
  input_tensor = Input(shape=(299, 299, 3) )
#Configure classifier
x = base_model.output
x = GlobalAveragePooling2D()(x)
#x = Dropout(rate = 0.2)(x) #to avoid overfitting if few samples
predictions = Dense(<num_classes>, activation = 'softmax')(x)
model = Model(inputs = base_model.input, outputs = predictions)
```

At this point, it is just needed to compile the model with the optimizer, losses function and training metrics.

Among the optimizers available in Keras are SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax and Nadam. As previously seen, the Most Used for training images are Adam and SGD for ITS rapid convergence. In addition, to fine-tune a model, SGD works very well as it converges rapidly. This project is set SGD with adaptive learning rate. The losses function is categorical cross-entropy because it is defining a classification system with several classes and the training evaluation metric chosen is the accuracy for having equitable data of each category.

```
#Compile the model
model.compile(
  optimizer = SGD(lr = 0.01, momentum = 0.9) ,
  loss = 'categorical_crossentropy' ,
  metrics=['accuracy'] )
```

5.2.4 Model training

To train the model, "fit_generator" function is used. It starts from the train and validation generator and also needs a series of parameters to configure.

The number of epochs refers to the number of times in which the entire dataset passes forward and backward through the neural network. There is no optimal number of epochs since a low value will produce underfitting while a too high will produce overfitting. With the configuration described in this project, it is unlikely that overfitting will occur, so it is about training the model until the validation accuracy does not increase more with training accuracy.

Steps per epochs refer to the number of training iterations in each epoch. This value will be the length of the training generator that is the number of training samples divided by the configured batch size. Validation steps are the same as above but for validation generator.

```
#Training the model
model.fit_generator(
  generator = train_generator,
  steps_per_epoch = len(train_generator), #train_samples//batch_size+1
  epochs = 25,
  validation_data = validation_generator,
  validation_steps = len(validation_generator), #val_samples//batch_size+1
  callbacks=[reduce_lr, csv_logger, tb_callback, checkpointer] )
```

Finally, this function allows the execution of callbacks. This project has used checkpoints to automatically save the results, CSV logger to save the accuracy and loss of each epoch, reduces learning rate on plateau to reduce the learning rate in the case that a metric does not improve compared to the previous epoch, and TensorBoard to display the parameters of training and validation.

The training process from scratch in this project is usually quite slow, lasting for weeks. However, using fine-tuning techniques can obtain a model with more than 80 % within hours. Depending on the pre-trained architecture, the batch size and the size of the input images each epoch have variable durations.

5.2.5 Prediction evaluation

Once trained the model, it can be used directly for getting a list of probabilities that belong to classes. Therefore, the category to which the image belongs or the categories to which belong are shown if there are doubts.

Based on the challenges of our dataset, evaluation methods are performed on the images of validation. Between the techniques of evaluating the accuracy and the loss, confusion matrix and other forms of scoring are used. To compare the models obtained it is taken into account the training time and some popular metrics from the scikit-learn library.

5.3 Analysis of the models

From the architectures available in Keras (2.2.5) with trained weights on ImageNet, different models that provide good performance at a low computational cost have been chosen. Due to the computational constraints of Colaboratory, it has not been possible to train the models with the same batch size and the largest possible one has been used for each architecture, speeding up the training process that can take up to a couple of days. In addition, the default input image size with the weights trained has been used for each architecture.

Depending on the complexity and the batch size, the training process time of each model varies considerably. Evaluation metrics are used to measure the quality of each model:

- **Accuracy:** Proportion of correct classifications. Is not reliable metric if the dataset is unbalanced.

$$Accuracy = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

- **Categorical cross entropy (loss):**

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} \log p_{model}[y_i \in C_{yi}] \tag{5.2}$$

- **Precision:** Percent of correct predictions.

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

- **Recall:** Percent of caught positive cases.

$$Recall = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{5.4}$$

- **F1-Score:** Percent of positive predictions correct.

$$F_1 = 2 * \frac{Recall * Precision}{Recall + Precision} = \frac{2 * TP}{2 * TP + FP + FN} \tag{5.5}$$

All of these metrics are used to compute the evaluation of every model taking special attention to the accuracy metric which is the best one because both training and validation sets are symmetrical, with the same number of samples of each class.

5.3.1 MobileNetV2

MobileNetV2 [56] is a small model, with low latency and power, designed for mobile devices. It is very useful to meet the computational restrictions of some use cases because of its lightness.

This model is very effective in executing object detection and segmentation. MobileNetV2 with respect to its previous version is 30% faster obtaining the same results. This improvement focuses on the use of linear bottleneck between the layers and shortcut connections between bottlenecks.

The following Table shows the main features of the MobileNetV2 pre-trained model with the weights of ImageNet dataset that is going to be used for transferring learning to our dataset.

Size	14 MB
Top-1 Accuracy	0.713
Top-5 Accuracy	0.901
Parameters	3,538,984
Depth	88
Default input size	224x224

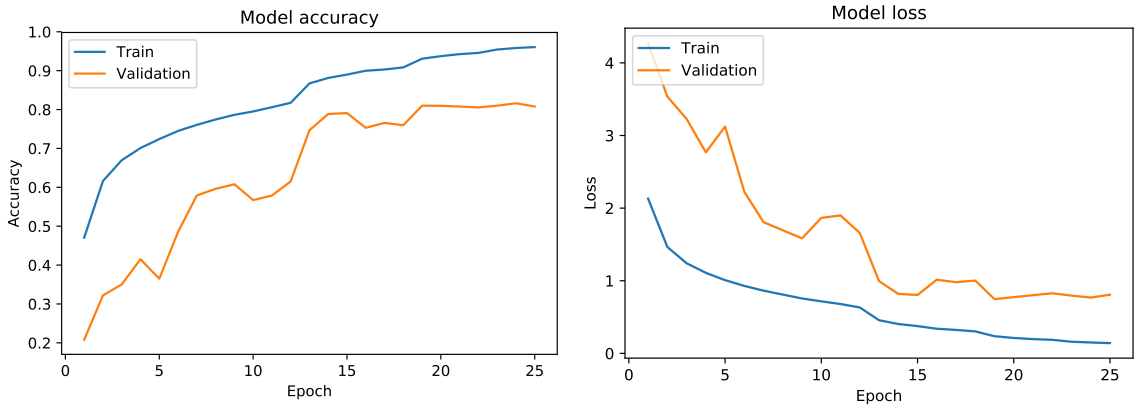
Table 5.1: MobileNetV2 ImageNet features

The following is the necessary configuration and training process for transfer learning using ImageNet weights on the ETHZ Food-101 dataset.

Training images	75328
Validation images	25189
Classes	101
Input size	224x224
Batch size	64
Epochs	25

Table 5.2: Parameter configuration MobileNetV2

MobileNetV2 is a light model and the default input size is not high, so the maximum batch size is 64 and the training process is very fast. As can be seen in the training process you may not need to train with so many epochs as this process is slow and each epoch has a duration of 1870 s in this case.



(a) Accuracy epochs

(b) Loss epochs

Figure 5.4: MobileNetV2 training accuracy and loss Food-101 dataset

From the above graphs, it can be highlighted that one epoch is not enough to achieve a good accuracy in the validation set and it is necessary to go for at least 15 epochs it achieves 79% accuracy.

The best of the models obtained for the validation set is reached around the epoch 19, so it would have been necessary to train the model during 19 * 1870 s or what is the same 9.86 h to reach an optimal model.

Metrics computed for model evaluation using the optimal model are shown in the following Table:

Accuracy	0.817
Loss	0.749
Precision	0.820
Recall	0.820
F1-Score	0.820

Table 5.3: Metric results for the optimal MobileNetV2 model obtained

As can be seen in above results, all metrics except loss (that is a different way of measuring quality of the model) have almost the same results, that is because the validation set have the same number of samples for each class so every sample is as important as any other.

5.3.2 ResNet50

ResNet50 [31] won the 2015 ILSVRC challenge. This architecture solves the problem of training deeper neural networks by introducing residual neural networks, which are deeper neural networks. It reformulates the layers as residual learning functions with reference in the input.

The following Table shows the main features of the ResNet50 pre-trained model with the weights of ImageNet dataset that is going to be used for transferring learning to our dataset.

Size	98 MB
Top-1 Accuracy	0.749
Top-5 Accuracy	0.921
Parameters	25,636,712
Depth	-
Default input size	224x224

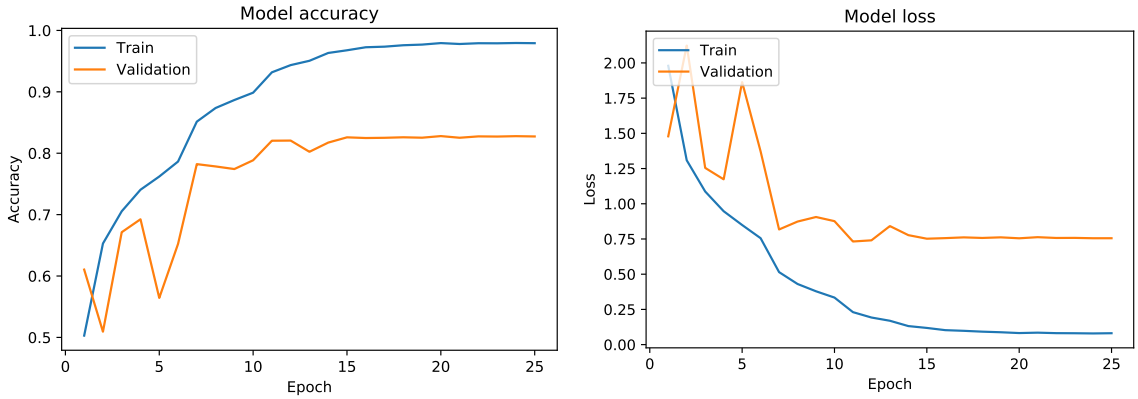
Table 5.4: ResNet50 ImageNet features

The following is the necessary configuration and training process for transfer learning using ImageNet weights on the ETHZ Food-101 dataset.

Training images	75328
Validation images	25189
Classes	101
Input size	224x224
Batch size	64
Epochs	25

Table 5.5: Parameter configuration ResNet50

ResNet50 is not a heavy model and the default input size is not high, so the maximum batch size is 64 and the training process is faster. As can be seen in the training process you may not need to train with so many epochs as this process is slow and each epoch has a duration of 2550 s in this case.



(a) Accuracy epochs

(b) Loss epochs

Figure 5.5: ResNet50 training accuracy and loss Food-101 dataset

From the above graphs, it can be highlighted that only one epoch achieves an accuracy of 61% in the validation set and in only 7 epochs it achieves 78% accuracy. In addition, it can be seen that both the accuracy and the loss validation are better than in training at the first epoch since it is a process of transfer learning.

The best of the models obtained is reached around the epoch 17, so it would have been necessary to train the model during 17 * 2550 s or what is the same 12.04 h.

Metrics computed for model evaluation using the optimal model are shown in the following Table:

Accuracy	0.827
Loss	0.755
Precision	0.830
Recall	0.830
F1-Score	0.830

Table 5.6: Metric results for the optimal ResNet50 model obtained

5.3.3 InceptionResNetV2

InceptionResNetV2 [59] achieves a new state of art in terms of accuracy. It emerges as a variation of InceptionV3 ideas using ResNet architecture. Residual connections allow networks to entailment deeper and simplify inception blocks. This model requires twice the memory and computational capacity than its predecessor InceptionV3. This architecture is the equivalent to InceptionV4 using ResNet ideas.

The following Table shows the main features of the InceptionResNetV2 pre-trained model with the weights of ImageNet dataset that is going to be used for transferring learning to our dataset.

Size	215 MB
Top-1 Accuracy	0.803
Top-5 Accuracy	0.953
Parameters	55,873,736
Depth	572
Default input size	299x299

Table 5.7: InceptionResNetV2 ImageNet features

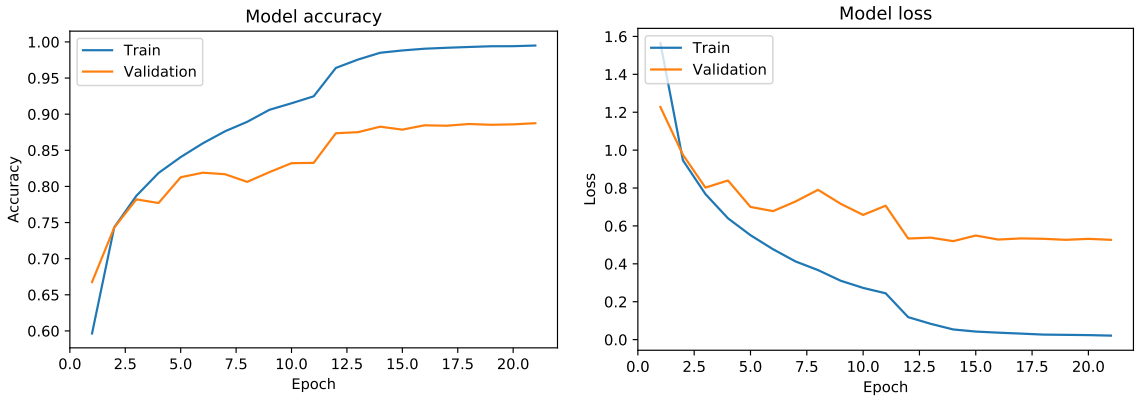
InceptionResNetV2 is a heavy model having a size of 215 MB and many parameters to train, plus the large input size affects the maximum batch size that can be used.

The following is the necessary configuration and training process for transfer learning using ImageNet weights on the ETHZ Food-101 dataset.

Training images	75328
Validation images	25189
Classes	101
Input size	299x299
Batch size	32
Epochs	21

Table 5.8: Parameter configuration InceptionResNetV2

As can be seen in the training process you may need to train with many epochs as this process is slow and it takes a long time to reach an optimum. Each epoch has a duration of 7000 s in this case.



(a) Accuracy epochs

(b) Loss epochs

Figure 5.6: InceptionResNetV2 training accuracy and loss Food-101 dataset

From the above graphs, it is worth noting that only one epoch achieves an accuracy of 66% in the validation set and in only 5 epochs it achieves 81% accuracy. In addition, it can be seen that both the accuracy and the loss validation are better than in training at the beginning since it is a process of transfer learning.

The best of the models obtained for the validation set is reached around the epoch 21, so it would have been necessary to train the model during 21 * 7000 s or what is the same 40.83 h to reach an optimal model.

Metrics computed for model evaluation using the optimal model are shown in the following Table:

Accuracy	0.887
Loss	0.527
Precision	0.890
Recall	0.890
F1-Score	0.890

Table 5.9: Metric results for the optimal InceptionResNetV2 model obtained

As seen in other models, all metrics except loss have almost the same results, that is because the validation set have almost the same number of samples for each class so every sample is as important as any.

5.3.4 Xception

Xception [22] is based on Inception modules, the intermediate step being in-between regular convolution and the depthwise separable convolution operation. This model may be the most balanced in terms of results and computational capacity since its size and accuracy provide a better ratio in ImageNet image training.

The following Table shows the main features of the Xception pre-trained model with the weights of ImageNet dataset that is going to be used for transferring learning to our dataset.

Size	88 MB
Top-1 Accuracy	0.790
Top-5 Accuracy	0.945
Parameters	22,910,480
Depth	126
Default input size	299x299

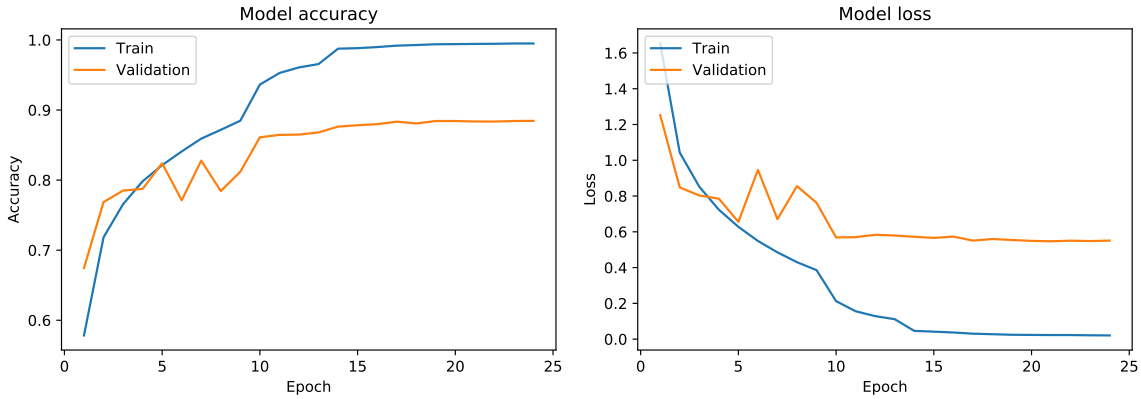
Table 5.10: Xception ImageNet features

The following is the necessary configuration and training process for transfer learning using ImageNet weights on the ETHZ Food-101 dataset.

Training images	75328
Validation images	25189
Classes	101
Input size	299x299
Batch size	16
Epochs	24

Table 5.11: Parameter configuration Xception

Although Xception apparently is not a heavy model, the input size directly affects the maximum batch size that can be used. As can be seen in the training process you may not need to train with so many epochs as this process is slow and each epoch has a duration of 6700 s in this case.



(a) Accuracy epochs *(b) Loss epochs*

Figure 5.7: Xception training accuracy and loss Food-101 dataset

From the above graphs, it is worth noting that only one epoch achieves an accuracy of 67% in the validation set and in only 5 epochs it achieves 82% accuracy. In addition, it can be seen that both the accuracy and the loss validation are better than in training at the beginning since it is a process of transfer learning.

The best of the models is reached around the epoch 15, so it would have been necessary to train the model during 15 * 6700 s or what is the same 27.9 h to reach an optimal model.

Metrics computed for model evaluation using the optimal model are shown in the following Table:

Accuracy	0.885
Loss	0.551
Precision	0.880
Recall	0.880
F1-Score	0.880

Table 5.12: Metric results for the optimal Xception model obtained

As seen in every model all metrics have the same behavior because of using a symmetrical dataset for validation.

5.4 Comparison of the models

Once obtained training results and transferring learning to the pre-trained models, they are going to be analyzed and compared. Epoch timing determines which architecture is better if there is no much time for training, while metrics comparison concludes the best model in terms of evaluation.

5.4.1 Epoch timing

The following Table shows the input size and the batch size maximum required to train each model. It is complemented with information about epoch duration, epoch number to achieve the best model and the time spent to get it.

Model	Input size	Batch size	Time/Epoch	Epochs	Time
MobileNetV2	244x244	64	**1870 s**	19	**9.86 h**
ResNet50	244x244	64	2250 s	17	12.04 h
InceptionResNetV2	299x299	32	7000 s	21	40.83 h
Xception	299x299	16	6700 s	**15**	27.9 h

Table 5.13: Epoch timing comparative

From these results, it can be determined that the faster training model with the same computation power is MobileNetV2 but that does not mean that is the best model, only it is the fastest architecture to obtain the local optimal model. InceptionResNetV2 is the lowest one but it wants to stand out that it achieves almost the same local optimal in much less time. To conclude, the Xception model obtains a local optimal in fewer epochs than the others because of using the lowest batch size.

5.4.2 Metric comparison

To discover the architecture that provides the best performance, the metrics and the time needed to reach the model are taken into account. The following Table shows each of the optimal models obtained, evaluated by metrics and the necessary training time:

Model	Time	Loss	Accuracy	Precision	Recall	F1-Score
MobileNetV2	**9.86 h**	0.749	0.817	0.880	0.880	0.880
ResNet50	12.04 h	0.755	0.827	0.830	0.830	0.830
InceptionResNetV2	40.83 h	**0.527**	**0.887**	**0.890**	**0.890**	**0.890**
Xception	27.9 h	0.551	0.885	0.880	0.880	0.880

Table 5.14: Metric comparison

From the preceding Table, it can be understood that depending on the available training time can be picked between MobileNetV2 or RestNet50 that obtain practically the same results and InceptionResNetV2 or Xception that also achieve similar results. From the first group, it is chosen MobileNetV2 since it gets almost the same results in 2 hours less, while from the second group it seems more practical to choose Xception since it obtains almost the same results in 15 hours less.

5.5 Analysis of the selected model: Xception

Xception is the selected model for the project case study since apparently, it is the one that renders the best metric results (with InceptionResNetV2). It has been chosen before InceptionResNetV2 because it is much more efficient (lighter and rapid convergence).

In this section is analyzed the entire model with its corresponding summary, the confusion matrix, and the classification report.

5.5.1 Feature summary

Accuracy	0.885
Loss	0.551
Time/Epoch	6700 s
Total time	27.9 h

Table 5.15: Summary of Xception model features

5.5.2 Confusion matrix

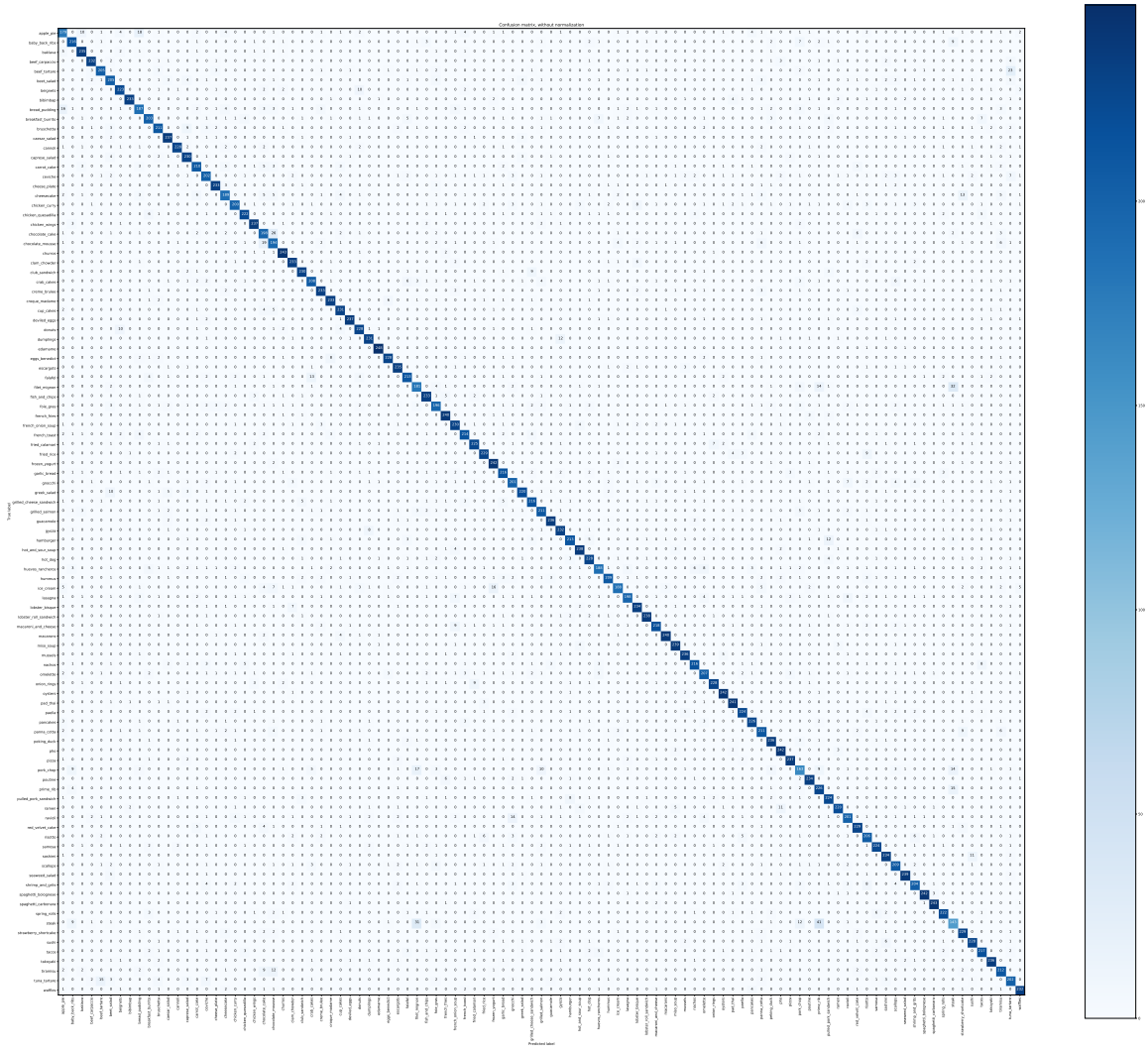


Figure 5.8: InceptionResNetV2 confusion matrix

5.5.3 Classification report

Class	F1-Score	Precision	Recall	Support
apple_pie	0.74	0.77	0.72	250.0
baby_back_ribs	0.86	0.86	0.86	250.0
baklava	0.92	0.9	0.94	250.0
beef_carpaccio	0.94	0.95	0.93	250.0
beef_tartare	0.85	0.89	0.82	250.0
beet_salad	0.82	0.81	0.82	249.0
beignets	0.89	0.89	0.9	249.0
bibimbap	0.96	0.98	0.93	250.0
bread_pudding	0.77	0.79	0.75	250.0
breakfast_burrito	0.84	0.87	0.82	249.0
bruschetta	0.85	0.86	0.84	250.0
caesar_salad	0.92	0.9	0.95	250.0
cannoli	0.94	0.97	0.92	249.0
caprese_salad	0.89	0.86	0.92	249.0
carrot_cake	0.85	0.86	0.85	248.0
ceviche	0.8	0.8	0.81	250.0
cheese_plate	0.93	0.93	0.93	250.0
cheesecake	0.8	0.83	0.77	247.0
chicken_curry	0.84	0.87	0.8	249.0
chicken_quesadilla	0.9	0.9	0.89	249.0
chicken_wings	0.94	0.94	0.95	250.0
chocolate_cake	0.77	0.74	0.8	249.0
chocolate_mousse	0.74	0.7	0.78	249.0
churros	0.95	0.93	0.97	250.0
clam_chowder	0.92	0.9	0.93	250.0
club_sandwich	0.92	0.92	0.92	250.0
crab_cakes	0.85	0.88	0.83	250.0
creme_brulee	0.92	0.91	0.94	249.0
croque_madame	0.92	0.91	0.94	249.0
cup_cakes	0.91	0.9	0.92	250.0
deviled_eggs	0.95	0.96	0.95	250.0
donuts	0.89	0.88	0.91	250.0
dumplings	0.92	0.92	0.92	250.0
edamame	0.98	0.97	0.99	250.0
eggs_benedict	0.93	0.94	0.92	249.0
escargots	0.93	0.91	0.94	249.0
falafel	0.87	0.88	0.85	250.0
filet_mignon	0.73	0.73	0.73	249.0
fish_and_chips	0.92	0.9	0.93	250.0
foie_gras	0.79	0.79	0.79	249.0
french_fries	0.94	0.92	0.96	250.0
french_onion_soup	0.88	0.85	0.92	250.0
french_toast	0.87	0.89	0.86	249.0
fried_calamari	0.9	0.91	0.9	250.0
fried_rice	0.92	0.93	0.92	249.0
frozen_yogurt	0.93	0.89	0.97	250.0
garlic_bread	0.86	0.86	0.86	250.0
gnocchi	0.82	0.81	0.83	243.0
greek_salad	0.9	0.93	0.88	250.0
grilled_cheese_sandwich	0.87	0.88	0.87	249.0
grilled_salmon	0.85	0.85	0.85	249.0
guacamole	0.95	0.95	0.94	250.0
gyoza	0.91	0.89	0.93	250.0
hamburger	0.88	0.91	0.85	250.0
hot_and_sour_soup	0.93	0.92	0.95	250.0
hot_dog	0.91	0.9	0.91	250.0
huevos_rancheros	0.81	0.86	0.76	248.0
hummus	0.87	0.89	0.84	248.0
ice_cream	0.82	0.9	0.75	250.0
lasagna	0.85	0.9	0.8	247.0
lobster_bisque	0.92	0.9	0.94	250.0
lobster_roll_sandwich	0.96	0.97	0.95	250.0
macaroni_and_cheese	0.89	0.89	0.89	246.0
macarons	0.97	0.97	0.96	250.0
miso_soup	0.94	0.93	0.96	250.0
mussels	0.93	0.92	0.94	250.0
nachos	0.89	0.92	0.86	250.0
omelette	0.84	0.86	0.83	250.0
onion_rings	0.92	0.93	0.91	250.0
oysters	0.95	0.93	0.97	250.0
pad_thai	0.94	0.92	0.96	250.0
paella	0.9	0.91	0.9	250.0
pancakes	0.9	0.9	0.91	249.0
panna_cotta	0.84	0.83	0.84	250.0
peking_duck	0.92	0.89	0.94	250.0
pho	0.93	0.9	0.97	249.0
pizza	0.95	0.95	0.95	249.0
pork_chop	0.68	0.71	0.65	250.0
poutine	0.93	0.93	0.94	250.0
prime_rib	0.81	0.73	0.9	250.0
pulled_pork_sandwich	0.88	0.86	0.9	250.0
ramen	0.91	0.94	0.88	250.0
ravioli	0.82	0.83	0.8	250.0
red_velvet_cake	0.91	0.9	0.92	250.0
risotto	0.82	0.81	0.84	246.0
samosa	0.91	0.91	0.9	249.0
sashimi	0.92	0.9	0.94	250.0
scallops	0.84	0.83	0.85	247.0
seaweed_salad	0.96	0.96	0.96	250.0
shrimp_and_grits	0.85	0.88	0.82	249.0
spaghetti_bolognese	0.97	0.98	0.97	249.0
spaghetti_carbonara	0.97	0.97	0.98	249.0
spring_rolls	0.9	0.92	0.89	250.0
steak	0.61	0.64	0.57	249.0
strawberry_shortcake	0.88	0.85	0.92	250.0
sushi	0.91	0.92	0.91	250.0
tacos	0.87	0.88	0.87	250.0
takoyaki	0.94	0.94	0.94	250.0
tiramisu	0.85	0.85	0.86	246.0
tuna_tartare	0.78	0.76	0.81	250.0
waffles	0.92	0.92	0.93	250.0
micro avg	0.88	0.88	0.88	25189
macro avg	0.88	0.88	0.88	25189
weighted avg	0.88	0.88	0.88	25189

Table 5.16: Xception classification report

From the classification report, it can be noticed that not all the categories have obtained the same metric results, and from the confusion matrix, it can be observed which groups are confused with each other. Some classes are easier to be recognized because of having different shapes, color, and density than other categories. On the one hand, edamame has 98% of F1-Score because there are not similar foods like beans, green bean or green peas, so it is easier to recognize it. On the other hand, steak has 61% of F1-Score getting confused with baby back ribs (9), filet mignon (31), pork chop (12), and prime rib (41) because they are similar foods.

CHAPTER 6

Web Scraping System for Data Extraction

This chapter presents the data extraction process using web scraping techniques and classifying results with a deep learning model. Additionally, it contains a previous analysis and database formatting for the case study.

6.1 Introduction

This chapter shows how to build a database by extracting data from web pages using web scraping techniques and applying a deep learning model for classifying. Based on the nature of the project, TripAdvisor web is selected. The main idea is to use TripAdvisor restaurants information and food images from reviews to model a dataset for the case study.

Firstly, an analysis of the web target is performed to adopt requirements. Secondly, are fixed the data prerequisites, and is performed database formatting thirdly. Lastly, is explained the entire process of web scraping, classifying and database formatting.

6.2 Previous analysis

6.2.1 Data prerequisites

From the case study, are known the restaurant fields needed for the project: name, reviews number, score, price, category, address, and a list of belonging images. These restaurants must belong to a specific area so will be filtered by city.

Once obtained this information, food class and accuracy of every image is predicted using the Xception model obtained for food classification.

6.2.2 Output database formatting

For the case study, extracted and predicted data must be formatted. As a result of the information obtained, there are two JSON tables. However, the case study needs the data merged into one table for its purpose.

(a) Tables obtained formatted

(b) Table merged for the case study

Figure 6.1: Database tables obtained

6.2.3 Web analysis

From the web page analysis, is known that the web is built using JavaScript for generating HTML code. Therefore, it is not possible to use common web requests because they are not able to run scripts. For that reason, is chosen Selenium web driver because it allows running scripts, like waiting until the document is ready and scrolling for loading all HTML code before receiving source page.

To parsing HTML requests and getting the required information, BeautifulSoup python library is used. This library allows searching for HTML components by class or id to ease data extracting task.

6.3 Methodology and configuration of the data modeling process

After the analysis, can be figured out the entire process diagram including the web scraping process to get all the required information, the classifier model process, and the database formatting. The following Figure shows the entire process diagram representing each process with a color:

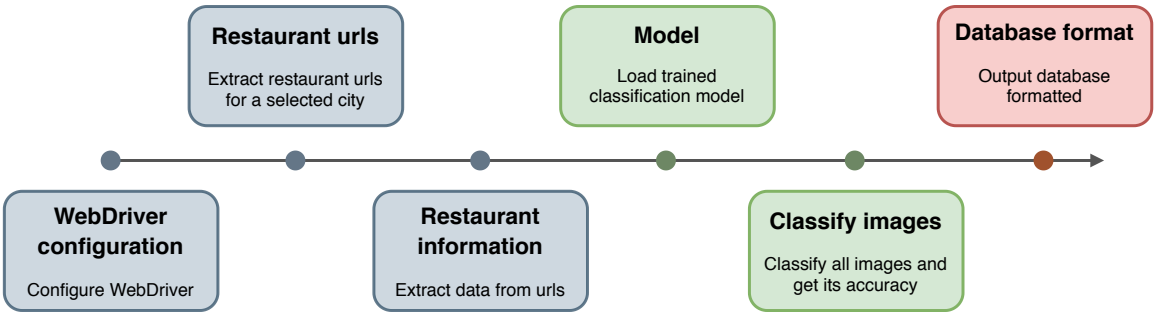


Figure 6.2: Modeling and web scraping diagram

6.3.1 Configuration of Selenium WebDriver

Selenium web driver (chromium) must be downloaded and included in the path to load configurations. In the project are defined two configuration methods explained below:

Web driver configuration logic follows this code fragment:

```
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument('--disable-dev-shm-usage')
driver = webdriver.Chrome('chromedriver', options=chrome_options)
driver.execute_script('return document.readyState')
```

Configuration logic runs a script ("return document.readyState") that waits until document and scripts ready. The meaning of each configuration arguments is:

- **headless:** It is a way to run Chrome browser in a headless environment which means running Chrome without chrome. It brings all modern web platform features provided by Chromium and the Blink rendering engine to the command line.
- **no-sandbox:** When running headless option in a container without a defined user, the chromeOptions environment property needs this argument or Chrome won't be able to startup.
- **disable-dev-shm-usage:** Allows to launch default flags and prevent from crashing.

Web scrolling logic allows to scroll down the web to load all scripts executed by scrolling. It executes scripts to get scroll height and scrolling down until the end. Web scrolling logic follows this code fragment:

```
scroll_height = 'document.documentElement.scrollHeight'
last_height = driver.execute_script('return '+scroll_height)
while True:
  # Scroll down to bottom
  driver.execute_script('window.scrollTo(0,'+scroll_height+');')
  # Wait to load page
  time.sleep(1)
  # Calculate new scroll height and compare with last scroll height
  new_height = driver.execute_script('return '+scroll_height)
  if new_height == last_height:
    break
  last_height = new_height
```

6.3.2 Extraction of restaurant URLs

The very first scraping process step is to get the desired data is to collect all restaurant links belonging to a city. In the case study, has been used restaurants from Majadahonda and surroundings so URL fragment must be the corresponding one.

The logic of the process consists in visiting the URL corresponding to the city, collecting restaurant URLs and detecting if there is a next page to iterate again. Special attention must be paid to "a" HTML tags belonging to a class and to the "href" parameter for obtaining the links. It is essential to add "time.sleep" functions to avoid blocking.

Once iterated through all pages, is obtained a list of restaurant links belonging to a city and surroundings and would be used in the next step.

6.3.3 Extraction of restaurant information

Extracting restaurant information from URLs is more complex than obtaining restaurant links. Special attention must be paid to every desired field, look for its HTML structure, and deal with missing tags/information. From the prerequisites analysis made, it is necessary to get the name, score, reviews, categories, price, address and image URLs, of each restaurant URL obtained before.

The web driver requests HTML document scrolled when it is ready to load more images. This procedure is looped through all restaurant pages using the restaurant URLs. To obtain information, attention is focused to "div", "h1", "span", and "a" HTML tags, and its corresponding classes and parameters. Must prevent from crashing dealing with missing and empty values, so it is more tricky.

Once iterated through all restaurant pages, it is obtained a dictionary that contains information of each establishment and includes images (restaurants without images are dispensable).

6.3.4 Configuration of the food classification model

At this point, it is required for the next step to load the food classification model built in the previous chapter and its configuration. Xception model obtained is loaded with Keras framework and all its configuration. It is going to be used to classify images in the next step.

6.3.5 Classification of the extracted images

All the image URLs obtained are opened and preprocessed for prediction using Xception model. The process consists of opening all images belonging to a restaurant and predict them saving the category and the accuracy.

Once iterated through all restaurant images, is obtained a dictionary containing image information.

6.3.6 Database formatting

Finally, it is required to merge the restaurant information table and restaurant images table into one for the case study and export it as JSON file orient to records. Merging tables into one gets some duplicate values, but, there is a small amount of data and, it is essential to make filters using the selected search engine.

CHAPTER 7

Food/Restaurant Searcher Web Application

This chapter presents the architecture, features and target devices of the web application for data visualization as the principal element of the case study and secondary part of the project.

7.1 Overview

Here a brief description of the web application for data visualization can be found. The application will allow the user to search restaurants by images of a certain kind of food. Additionally, resulting images can be filtered, and it is also implemented to search for images belonging to a restaurant. The objective of this chapter is to show the architecture, the main features and target devices of the web application.

- **Architecture:** Due to the nature of the application, it is possible to execute the entire application on the user's device except for image requests to the server, obtaining better performance and usability. The structure of the adapted MVC version developed, use cases, and functionalities are included in this section.
- **Features:** The application is user-centered designed, fitting all devices due to mobile first design and responsiveness as can be seen in the mock-ups of application views.
- **Target devices:** Application targets Android devices or devices with web browser access. This section shows differences between both platforms and additional files needed for compilation.

7.2 System architecture

The following Figure shows a diagram of the system architecture:

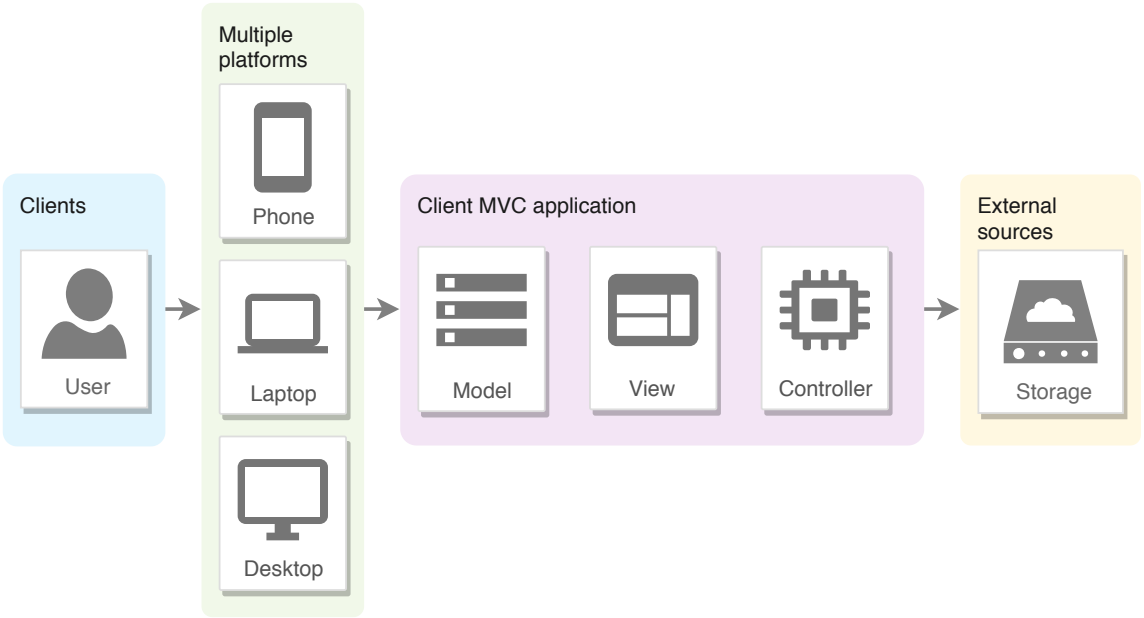


Figure 7.1: System architecture diagram

As can be observed in the diagram, the architecture of the system starts with users accessing their devices. There are two ways to reach the application, by android app or by a web browser. The application architecture follows an adapted version of MVC architecture and access to external storage by HTTP requests.

7.2.1 Structure

- **Clients:** Actor users that access the web application using any available platform and device.
- **Platforms:** Different devices that have access to the web application. Mobile, Laptop, Tablet, and Desktop are included.
- **Application:** The application develops an adaptation of Model View Controller (MVC). The view requests the controller and response visually, the controller requests data to the model and the model consult the database. The view corresponds to code that shows the business logic by using templates, the controller is composed by functions that manage the view and request data to the model, and the model is the representation of information to be accessed with queries.

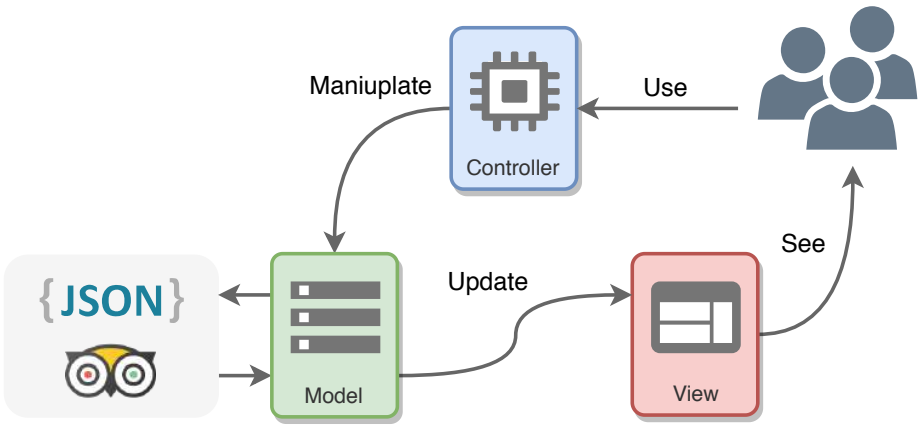


Figure 7.2: Model View Controller (MVC) architecture

- **External sources:** External storage hosts images demanded by HTTP requests.

7.2.2 Use cases

Application use cases interact with principal actors and secondary actors. On the one hand, the principal actor is the user that wants to search certain types of food images with filters, see image information, inspect restaurant food images, and visit the restaurant website. On the other hand, the secondary actor is TripAdvisor external entity.

In the following Table, each one of the involved actors is explained in detail, indicating their id, name, type, roll, and description:

ID	Name	Type	Role	Description
ACT-1	User	Primary	User	The user is the actor that wants to search restaurants by food images.
ACT-2	TripAdvisor	Secondary	Entity	TripAdvisor website hosts the images and the restaurant sites.

Table 7.1: Primary and secondary actors

The use case diagram is represented using Unified Modeling Language (UML). It is composed of a primary actor (user), a secondary actor (TripAdvisor) and application use cases. A complete (concrete) use case can be an extension of other adding features, and an incomplete (abstract) one, includes other use cases.

In this case, "Filter images" use case is an extension of "Search food images", as "Visit restaurant website" is an extension of "Show information" and "Inspect restaurant food images". "Search food images" and "Inspect restaurant food images" are abstract use cases, thus both of them must include "Compute statistics" and "Show information" required use cases to be completed.

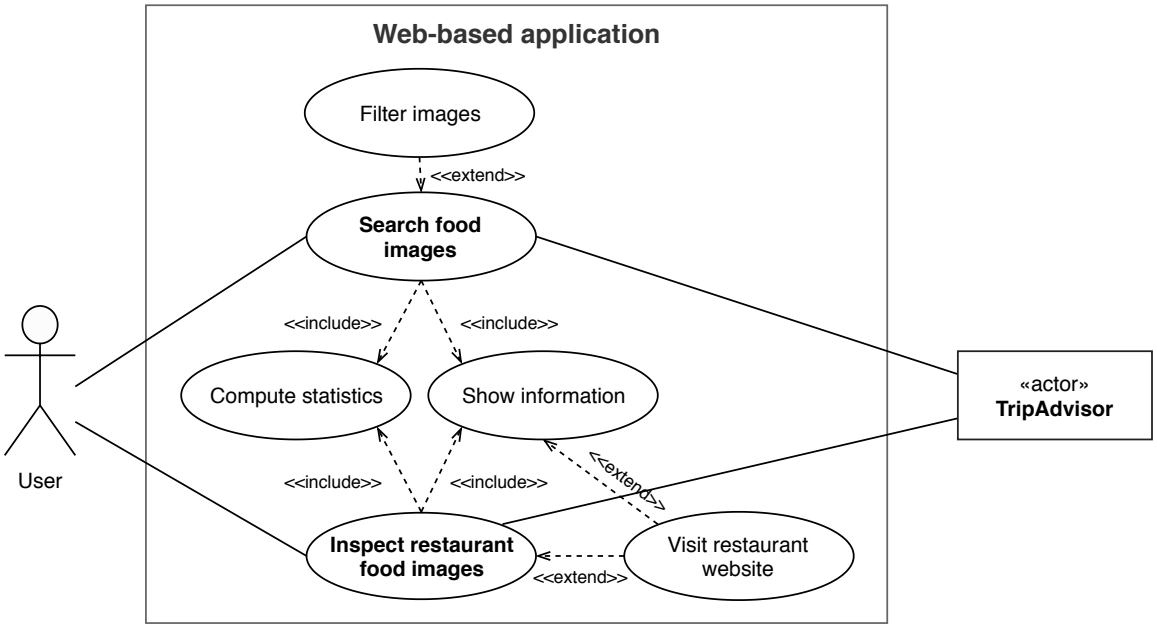


Figure 7.3: UML Use cases diagram

The following use cases include the user story with a descriptive table. Every use case descriptive table has the use case id, name, type, precondition, and events. Precondition refers to a necessary event that must precede the use case, and events refer to actor input actions and system response.

- **Search food images (UC-1):** As a user, I want to search for food images typing or selecting the kind of food in the search bar for obtaining a displayed grid of food images and statistics.

ID	UC-1
Name	Search food images
Type	Incomplete (abstract)
Precondition	-
Actor Input	Select available food in the search bar
System Response	Grid of searched food images and statistics

Table 7.2: Use case 1: Search food images

- **Inspect restaurant food images (UC-2):** As a user, I want to inspect restaurant food images from food image information for obtaining a grid of food images and statistics belonging to the restaurant.

ID	UC-2
Name	Inspect restaurant food images
Type	Incomplete (abstract)
Precondition	-
Actor Input	Search restaurant food images from image food information
System Response	Grid of restaurant food images and statistics

Table 7.3: Use case 2: Inspect restaurant food images

- **Filter images (UC-3):** As a user, I want to filter food image results by accuracy, score, price, and reviews for obtaining a displayed grid of filtered food images and statistics.

  This use case is an extension of "Search food images".

ID	UC-3
Name	Filter images
Type	Complete (concrete)
Precondition	-
Actor Input	Fill filter boxes and select available food in the search bar
System Response	Grid of filtered food images searched and statistics

Table 7.4: Use case 3: Filter images

- **Visit restaurant website (UC-4):** As a user, I want to visit the restaurant website of a founded food image so that I can book.

  This use case is an extension of "Show information" and "Inspect restaurant food images".

ID	UC-4
Name	Visit restaurant website
Type	Complete (concrete)
Precondition	Have selected a food image
Actor Input	Click TripAdvisor restaurant website button
System Response	Open a new tab to redirect to the corresponding restaurant website

Table 7.5: Use case 4: Visit restaurant website

- **Compute statistics (UC-5):** As a user, I want to see food image statistics like the number of results out of the total, the number of restaurants out of the total, location, and filter configuration.

  This use case is included in "Search food images" and "Inspect restaurant food images".

ID	UC-5
Name	Compute statistics
Type	Complete (concrete)
Precondition	Have made a search
Actor Input	Fill filters and select available food in the search bar
System Response	Show food image statistics

Table 7.6: Use case 5: Compute statistics

- **Show information (UC-6):** As a user, I want to see food image information like restaurant name, address, score, price, reviews, categories and accuracy of image prediction.

  This use case is included in "Search food images" and "Inspect restaurant food images".

ID	UC-6
Name	Show information
Type	Complete (concrete)
Precondition	Have made a search with food image results
Actor Input	Select a food image
System Response	Show modal food image and restaurant information

Table 7.7: Use case 6: Show information

7.2.3 Functional modules

In this subsection are analyzed functional classes as functional modules using Unified Model Language (UML). Functional modules to carry out the web application are included in the following UML diagram of classes:

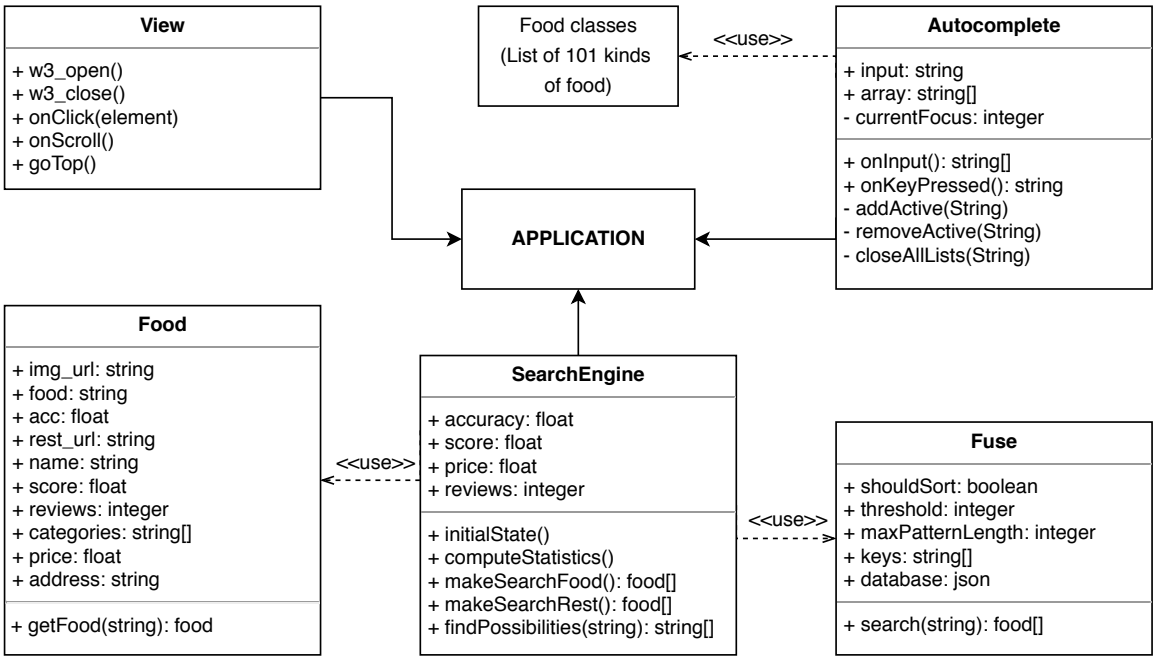


Figure 7.4: UML Classes diagram

There are three different classes and two auxiliary included in main modules that compose application logic.

- **View:** View class is composed of all visual functions that generate HTML views. This class is responsible for managing visual elements like modals, buttons, and effects.
- **SearchEngine:** This class is the search engine of the application completed by food and fuse classes. SearchEngine has filter parameters and includes fuse engine and food model to compute food image searches or a restaurant image searches. It has also added functions to find occurrences and manage state and statistics.
- **Food:** This class defines the food image model to get data from the database.
- **Fuse:** This class contains fuse search engine for JSON databases. It is composed of some same configuration parameters and a search function with a query parameter.
- **Autocomplete:** The objective of this class is to manage food possible inputs in the search bar. It has included a list of food classes as the input array.

7.3 Application layout

Design application has followed a set of standards and techniques for achieving user accessibility and multiplatform adaptability.

7.3.1 User-centered design

User-centered design (UCD) [13] philosophy aims to create products that meet the needs of end users to achieve the highest satisfaction and best user experience. This theory resembles a process using multidisciplinary techniques where, each decision, is based on needs, expectations, objectives, capabilities, and motivations of users. The main stages to achieve a user-centered design are:

1. Thorough knowledge of end users, using qualitative research or quantitative research.
2. Design a product that meets the needs of the user and adjusts to their capabilities, expectations, and motivations.
3. Test the design using user tests.

Following this philosophy have achieved valid outcomes in terms of accessibility, developing an application that gets high satisfaction and transcendent user experience possible with the minimum effort from the user.

The web application consists of simple use cases in which it is only necessary to write into the inputs so that the search is auto-completed, and intuitive buttons with a hover effect that helps to find functionalities.

7.3.2 Responsive web design

Responsive web design (RWD) [26] is a development philosophy that aims to adopt the view of the website to the device used to visit it. Each device, whether tablet, mobile, laptop or desktop, has a set of features such as screen size, resolution, CPU power, operating system or memory. Therefore, this concept pretends that everything is displayed correctly on any device with a unique web design.

A single HTML/CSS version can cover all screen resolutions hence the website will be optimized for different devices and screen resolutions. That improves user experience unlike what happens, for example, with fixed-width websites when accessed from mobile devices. Thus the production and maintenance costs are reduced for similar target devices.

Adaptive web design becomes possible with media queries in the properties of CSS3 styles. Media queries are a set of commands that are included in the stylesheet that tells the HTML document how to behave in different screen resolutions.

W3.CSS framework has been used to develop a responsive design. The application is oriented to desktop, laptop, tablet, and mobile devices, therefore, media queries have min-width parameter higher than 601 px and 993 px to build three views, particularly fixed to mobile design.

7.4 Targeted platforms

Architecture and file structure is different for each developed application platform. This application is targeted for web browser access and Android devices, thus each application has its own configuration and deployment.

The following Figure shows the Android directory structure and the web application content:

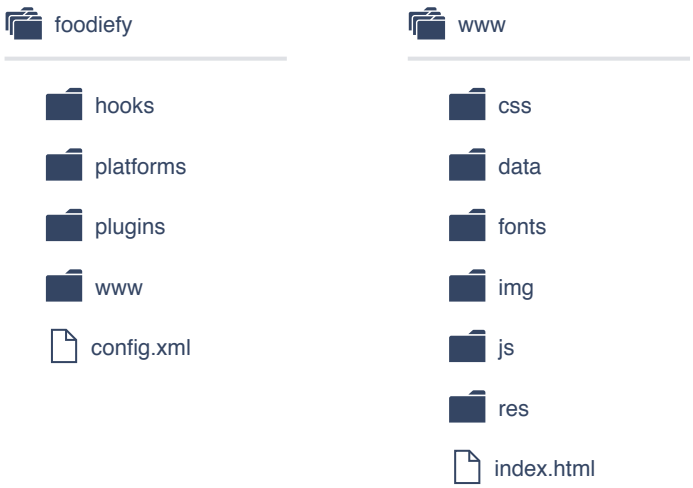


Figure 7.5: Application directory structure

On the one side, Android application needs all files and folders including Apache Cordoba plugins and Android PhoneGap configuration. On the other hand, web browser application just needs "www" folder including HTML, CSS and JavaScript files to be deployed as a web application. The web browser application is built as a Docker image to be deployed and run on the Docker Engine.

7.4.1 Mock-up

The following Figures show the two main mock-up views of the web application for every device including Desktop, Laptop, Tablet, and Smartphone. All mock-up views show responsiveness of the web application.

Figure 7.6: Application mock-up view

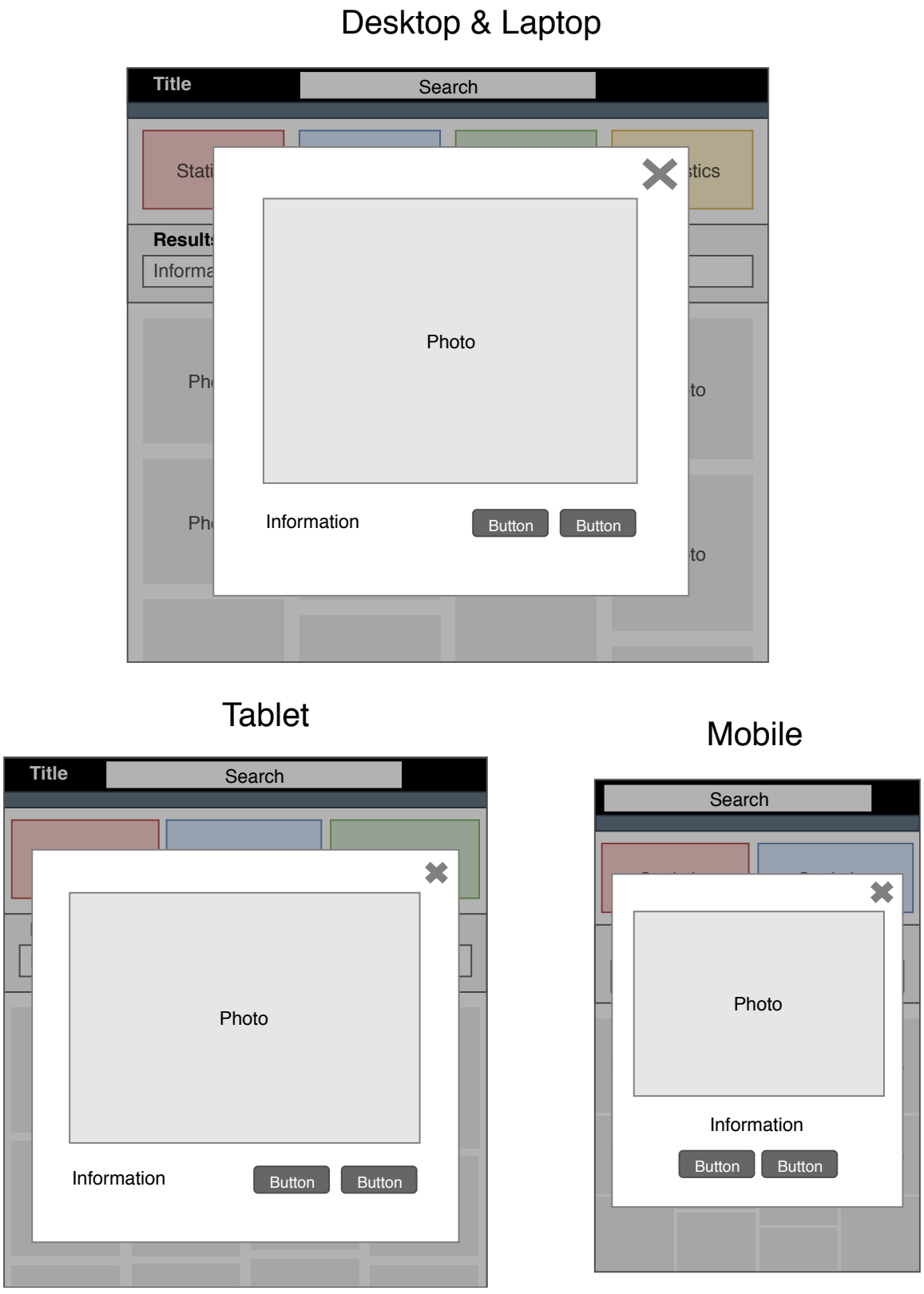


Figure 7.7: Modal mock-up view

In the application mock-up view is drawn the application structure, showing a header with title and a search bar, another header bar for filters, statistic spaces, result header with information and a grid of images. In the modal mock-up view is drawn the structure of the modal shown when clicking an image, including view restaurant food images and visit restaurant website buttons.

CHAPTER 8

Case Study — Foodiefy

This chapter shows the result of a practical case as a web application. It proposes an application concept after analyzing the context. Additionally, each part of the app and the interactions are shown in detail.

8.1 Introduction

8.1.1 Context of search engines

Search engines are undoubtedly the most used instruments to locate information on the Internet. A search engine is, in fact, a tool that manages databases of URLs with different contents. There are different types of search engines:

- **Search engines:** Usually, manage pages. It is search engines by content.
- **Thematic indexes:** They usually manage complete information resources composed of one or more web pages. They work as catalogs or directories by categories.
- **Metasearch engines:** Group or combine the potentials of several search engines.

The process carried out by any search system can be summarized in data collection, analysis (indexing/categorization), search, and recovery.

8.1.2 Application concept

Nowadays there are numerous search engines, going from finding websites to finding a partner. This case study arises from the idea of looking for restaurants avoiding the traditional search way and giving a twist to conventional systems.

Have you ever wanted to eat a type of food that has a certain aspect? This search engine allows you to search for images belonging to food categories and select them by its appearance, which makes it possible to find the restaurant with the desired food.

This concept is executed in the form of a web application using an image classifier model to categorize images extracted from TripAdvisor and index them to create the search engine.

Figure 8.1: Foodiefy logo brand

The name chosen for the food/restaurant search engine is Foodiefy composed by "foodie" who is a person with a particular interest in food and "-fy" ending. The logo brand is a fork contained in a two-color circular bluish-green background.

8.2 Features

As a summary, the different technologies and data sources used to carry out the application are listed below.

8.2.1 Technologies

The technologies, libraries, and frameworks used to perform the Foodiefy web application are listed below:

- **Keras — TensorFlow:** They have been used to build a model for food image classification.
- **Scikit-Learn:** This library has helped to choose the right model by scoring them with different metrics.
- **Selenium Web Driver:** The framework used to extract information from the website of TripAdvisor.
- **W3.CSS:** It is the base of the CSS view of the application for obtaining a responsive website.
- **FontAwesome:** All icons showed in the application come from this font library.
- **BeautifulSoup:** Useful to parse specific document files and extract information effortlessly.
- **JSON:** Output format of the database.
- **Fuse.js:** JavaScript library working as the search engine.
- **Android / PhoneGap:** The framework to build an android application with Apache Cordoba plugins.
- **Docker:** The open-source project to build Docker images and deploy the web application.

8.2.2 Data sources

The data sources that have been used to accomplish the project are:

- **ETHZ Food-101:** The labeled food image dataset to build Keras classification model.
- **TripAdvisor:** It is the website employed to extract and obtain information for the application.

8.3 Device responsiveness

The application of Foodiefy has been developed for all kinds of devices, going from smartphones to large screens. The following list shows the devices and resolutions with a screenshot of one of the main views are shown to display results.

- **Smartphone size**

  This is the limited appearance of the application omitting some buttons and functionalities. It is destined at small screens like a smartphone. The following screenshot shows the view for devices with a width resolution less than 601px:

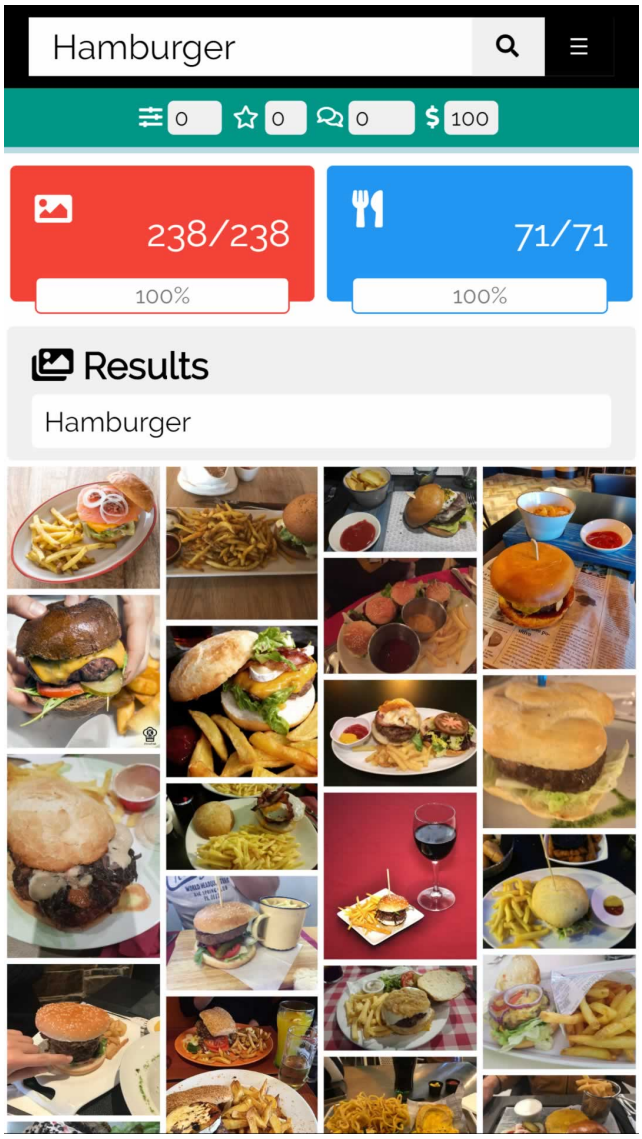


Figure 8.2: Smartphone screenshot responsive

- **Tablet size**

  This is the partial form of the application including almost all buttons and functionalities. It is destined at medium screens like a tablet. The following screenshot shows the view for devices with a width resolution between 601px and 993px:

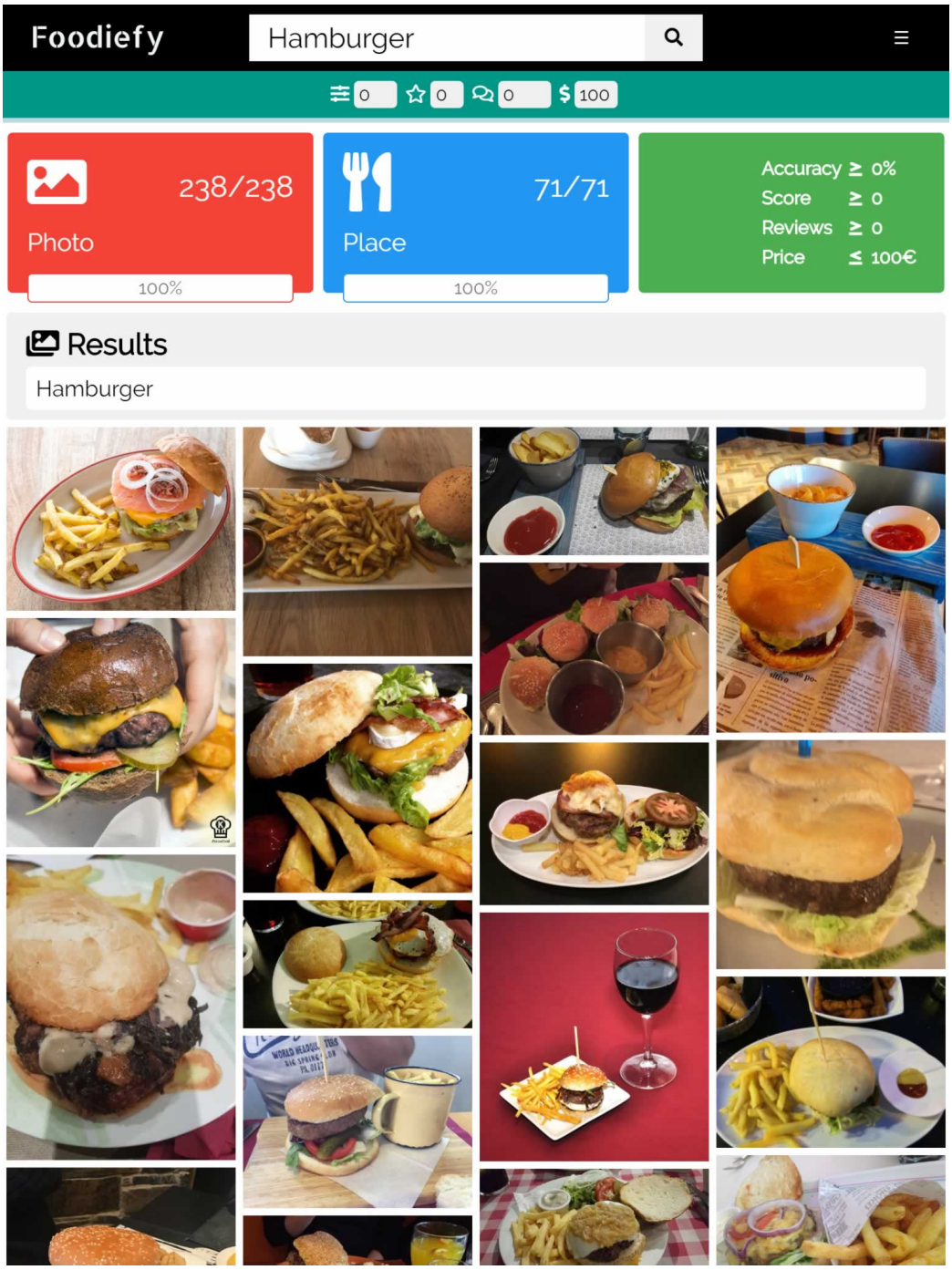


Figure 8.3: Tablet screenshot responsive

- **Desktop & Laptop size**

  This is the full look of the application including all buttons and functionalities. It is destined at big screens like laptop and desktop. The following screenshot shows the view for devices with a width resolution higher than 993px:

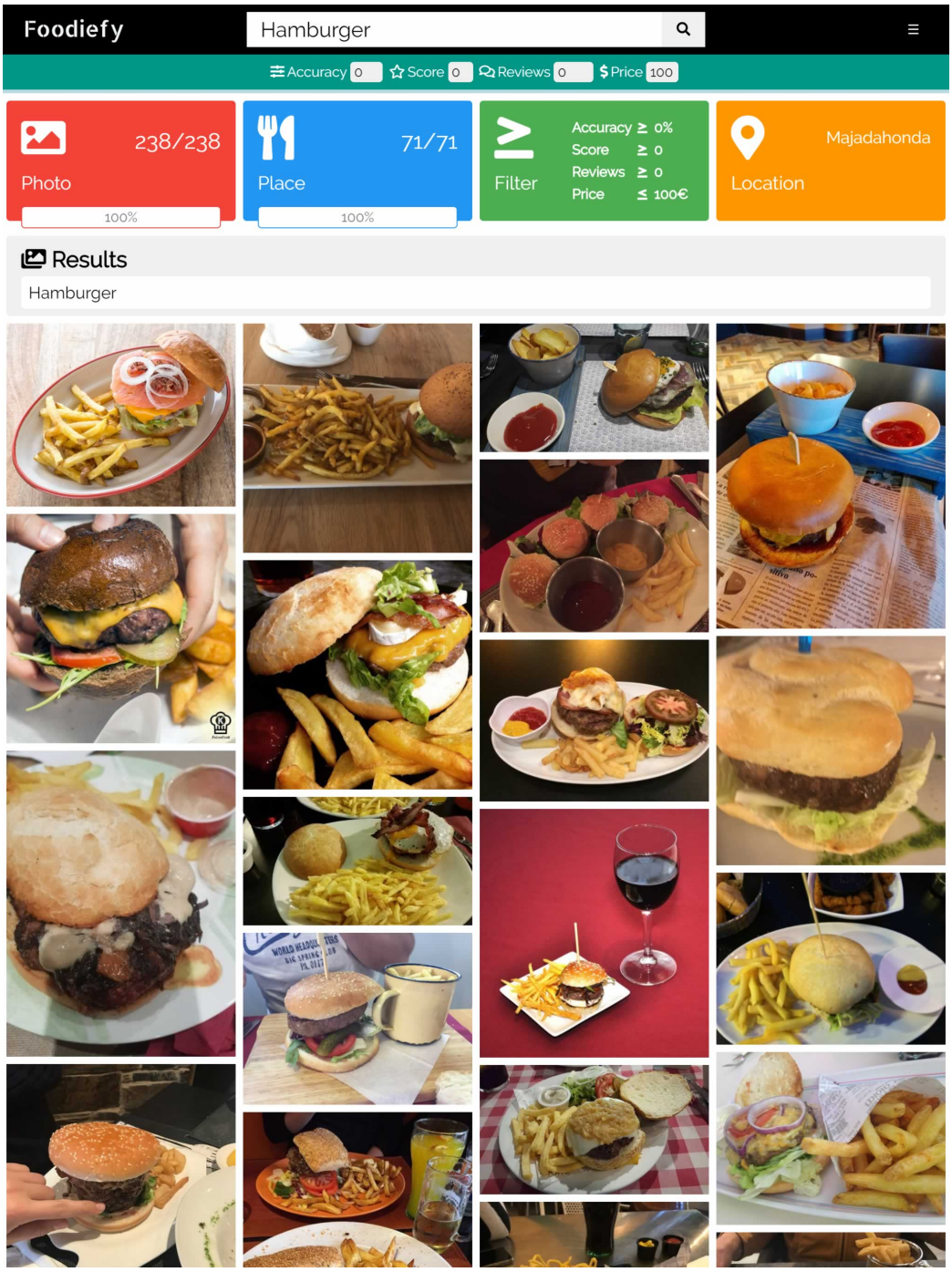


Figure 8.4: Laptop & Desktop screenshot responsive

8.4 Interaction overview

The following diagram shows the interaction overview of the Foodiefy web application:

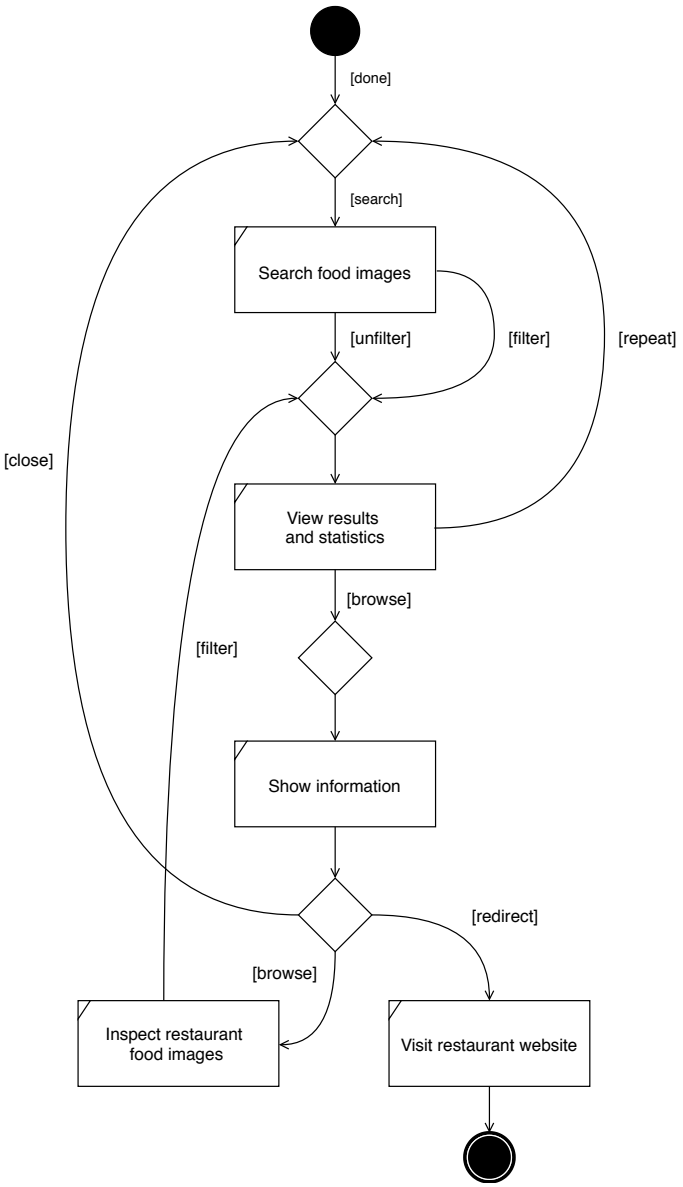


Figure 8.5: Interaction overview diagram

As can be seen, the interaction starts in the black dot when the application has been loaded. At that step, there is only one possibility, to make a search. The search can be filtered or unfiltered so that the results and statistics are shown. Now, there are two available options, to repeat the search, or to browse one image to show the information. Finally, in the modal view, there are three options, to browse and inspect restaurant food images, to close and return to the initial point, or to redirect and visit the restaurant website.

8.5 Views

The section of views exposes the functionality of each one of the aspects of the web application and shows for each device how it is rendered. All views are composed of the search bar with a button to display the menu, the filter bar, a search statistics area, the area where the results are displayed and the footer with information.

8.5.1 Home page view

The home page is the main view that is displayed when you open the application, ready to make a search with the desired filters.

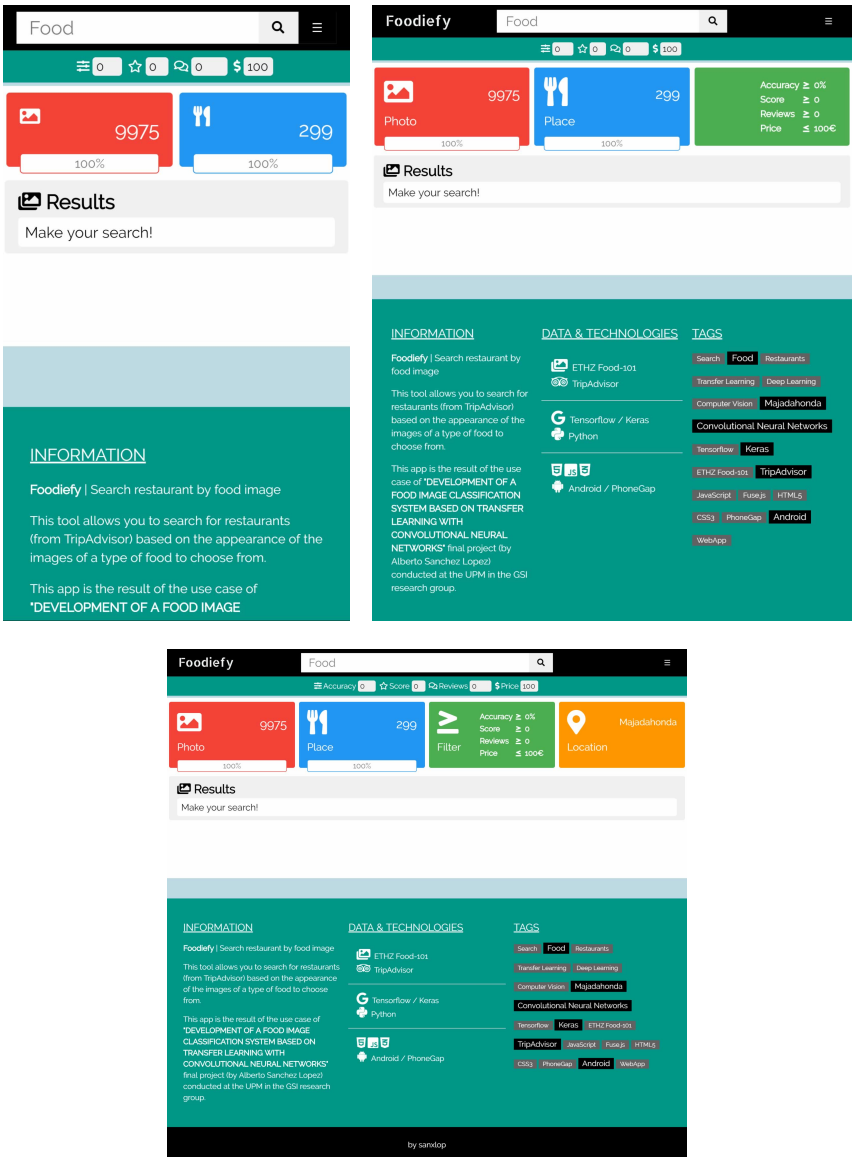


Figure 8.6: Home page view

8.5.2 Searched food view

The "searched food view" is the composition that is displayed after making a request in the search engine. The statistics show the number of images found for this type of food, the number of restaurants to which these images belong, the filters introduced and the selected location. Each of the results obtained as an image is clickable and opens a modal with information.

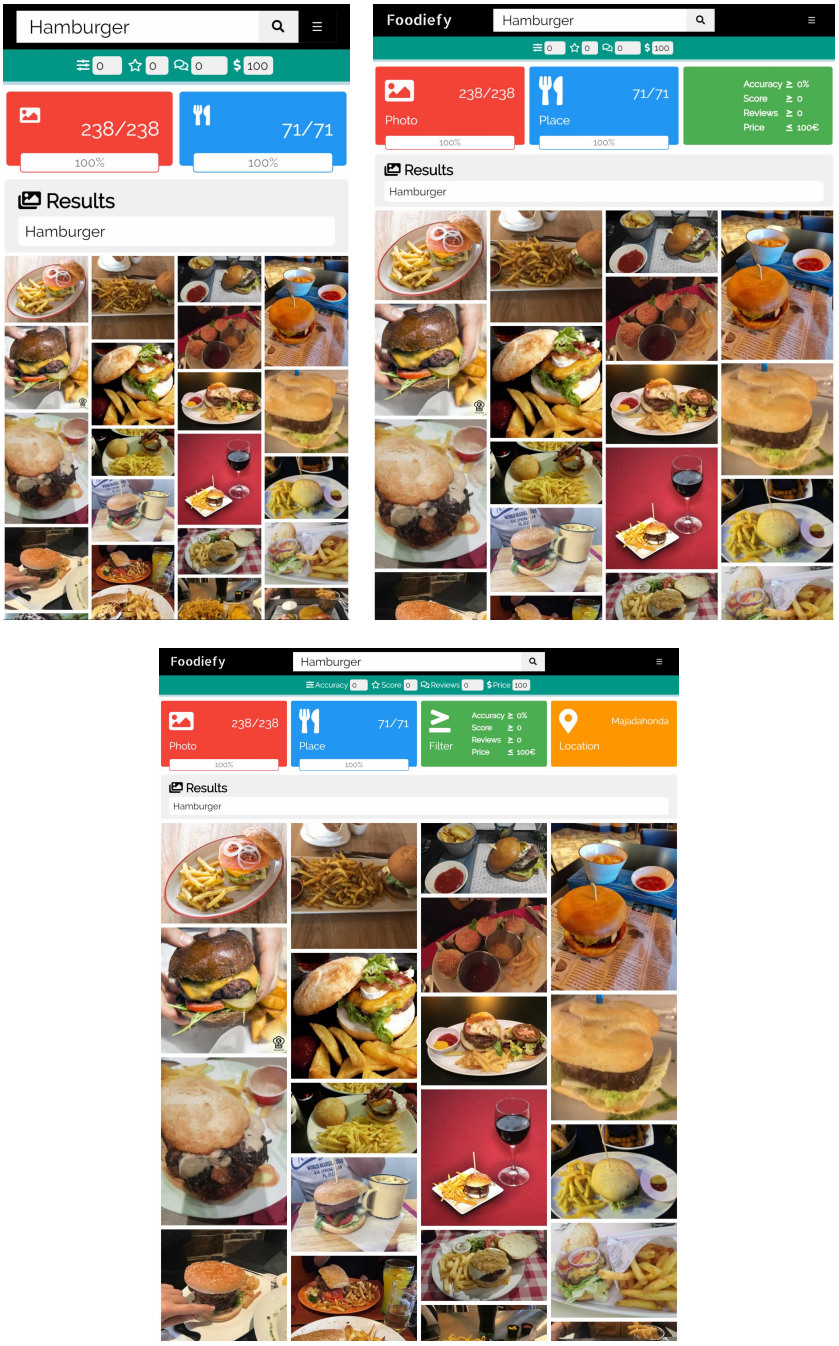


Figure 8.7: Searched food view

8.5.3 Selected food modal view

The "selected food modal view" is the composition that is displayed after clicking an image from the searched results. It shows the full-size image with accuracy prediction information and all restaurant information. Additionally, it includes three buttons, one for seeing more restaurant images in Foodiefy, another for redirecting to TripAdvisor website, and an extra one to close the modal.

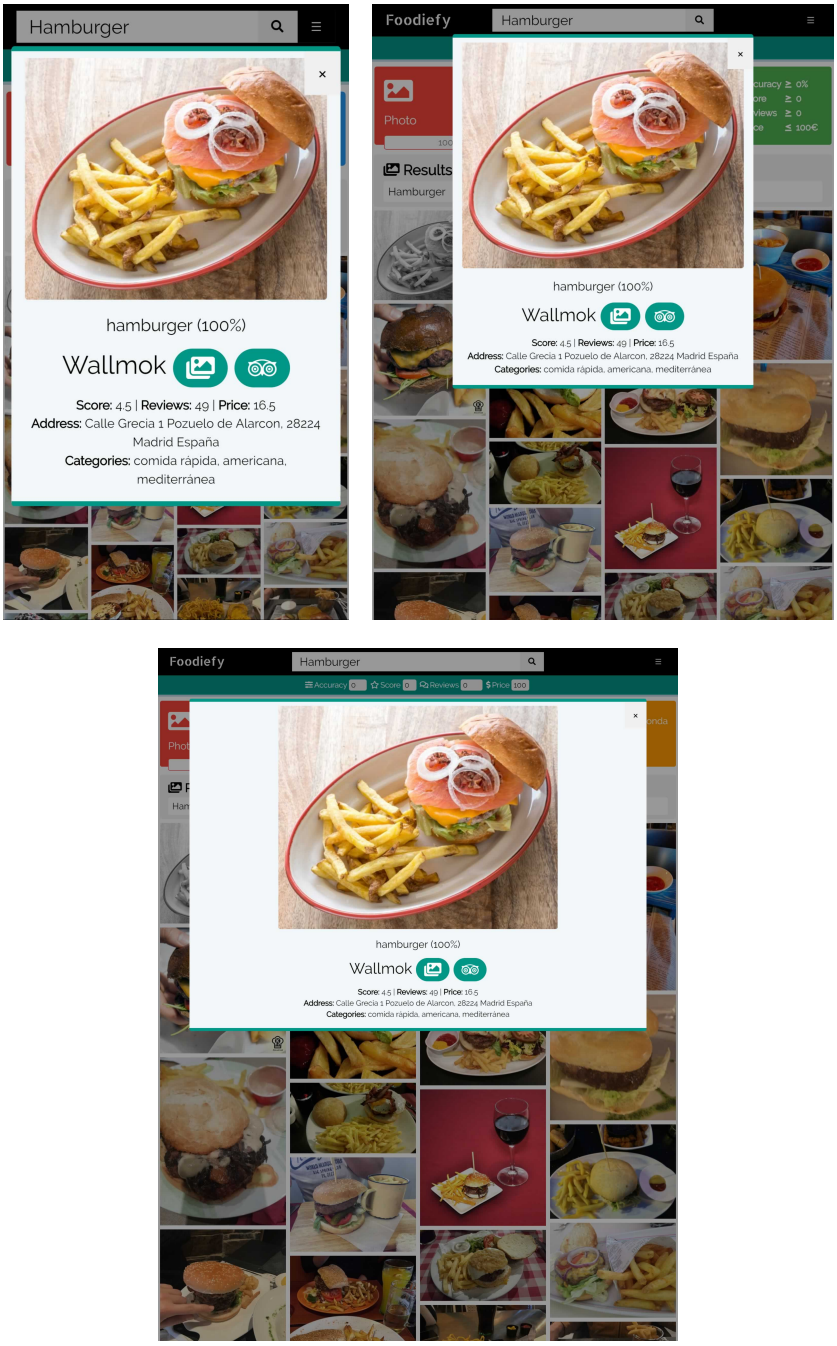


Figure 8.8: Selected food modal view

8.5.4 Restaurant food images view

The "restaurant food images view" is the composition that is displayed after clicking more restaurant information button. The statistics show the number of images belonging to the restaurant, the number of restaurants to which these images belong, the filters introduced and the selected location. Each of the results obtained as an image is clickable and opens a modal with information.

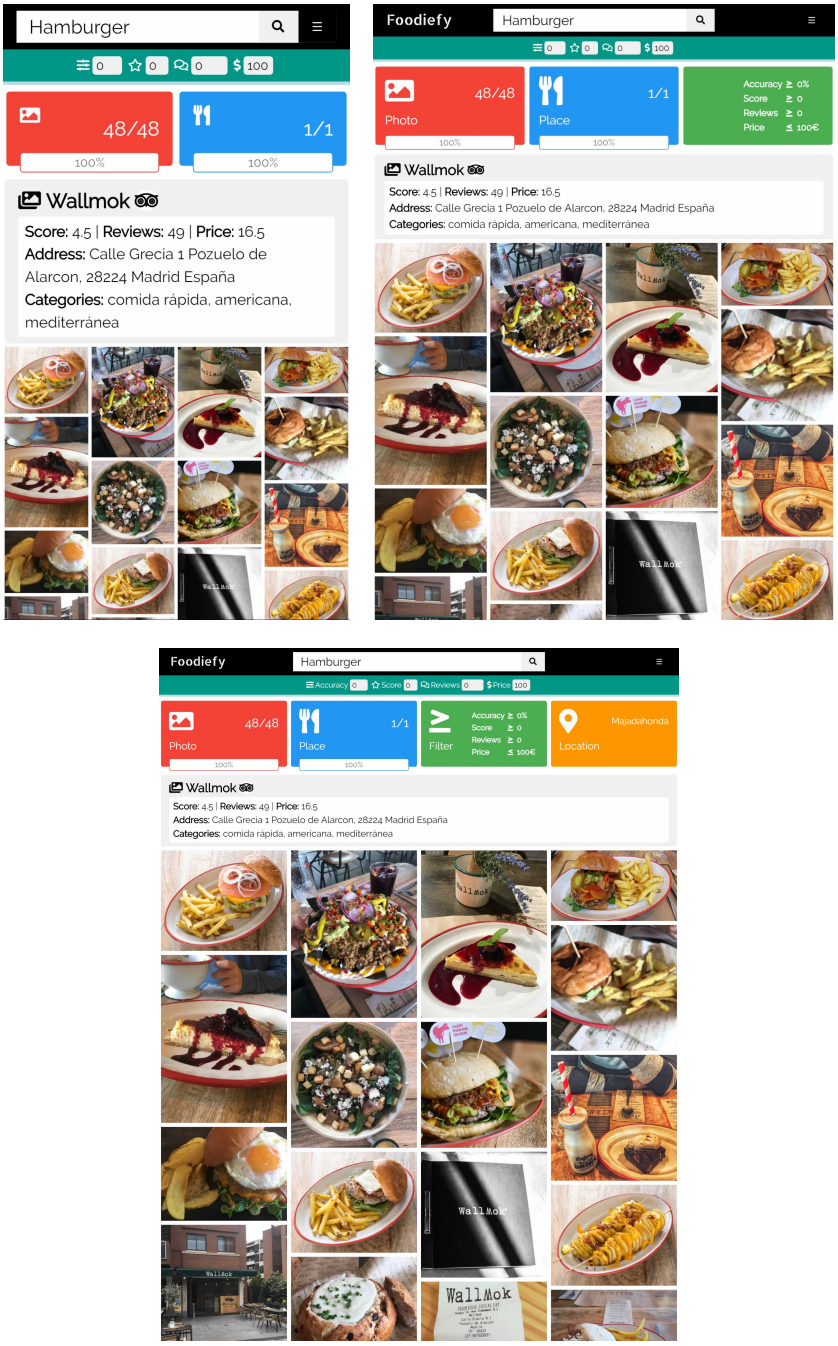


Figure 8.9: Restaurant food images view

8.5.5 Menu view

The "menu view" is the composition that is displayed after clicking the menu button at the search bar. It shows a slideable menu with index, about and reset search options. Index option goes to the top page, about option goes to the top of the page and reset search options restart the search, filters, and results.

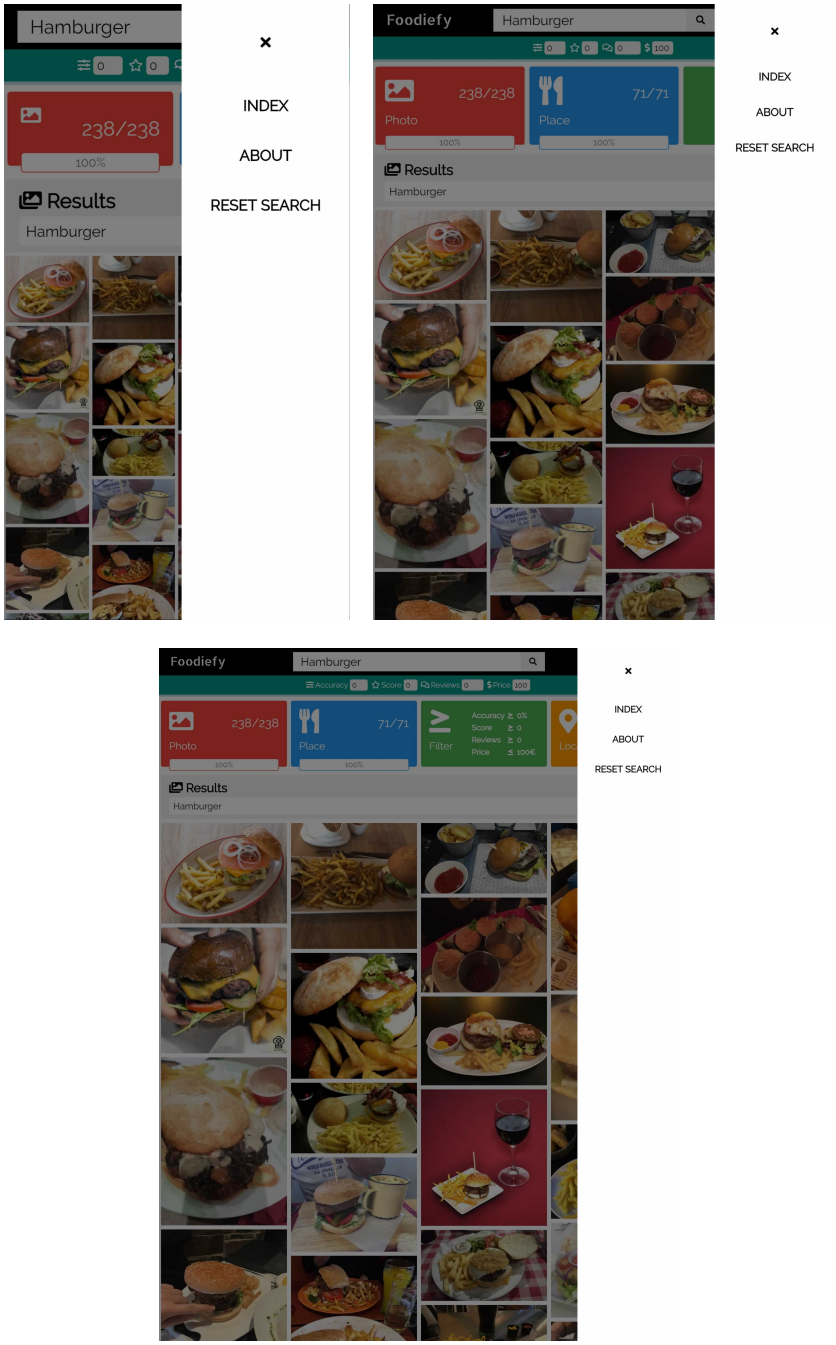


Figure 8.10: Menu view

As has been observed, each of the views is rendered correctly to each of the device sizes gratefulness to the responsive design. In addition, the simple use of the application is demonstrated with the interaction diagram commonly with the views.

CHAPTER 9

Conclusions and Future Work

This chapter details the achieved goals and outcomes done by the master thesis following key points developed in the project. Additionally, there are described the future lines of work and the repository to continue in the same way.

9.1 Achieved Goals

All the objectives raised at first, have been satisfactorily completed. Furthermore, the new proposals and ideas that have emerged during the development of the work have been incorporated by the additional value that they entailed.

In state of the art, it has been investigated all linked to image recognition and deep learning techniques for image analysis, going through the most important artificial network architectures. Next, in enabling technologies are described all methodologies and frameworks used in the project for image processing, data extraction, and web application tools. Techniques and methodologies proposed have been used in the next chapters and in the case study.

As a result of analyzing the different image recognition architectures for transferring learning, it has been trained a Keras model of image classification achieving almost 89% of accuracy for ETHZ Food-101 challenge dataset with Xception architecture.

Food images and restaurant information have been extracted using web scraping techniques with Selenium web driver from TripAdvisor's website. The model obtained has been applied to predict the labels of the food images extracted, so that extra value is reached in the dataset.

Finally, for summarizing, a general system architecture to fit each of the modules that make up the project has been done. To show project achievements and results, all jointly goals succeed in Foodiefy, a web application that has been developed for every device with a web browser or Android OS.

9.2 Links of interest

The code of the tools for building an image classification model using transfer learning techniques and the web scraping system developed during the project, is available in the following GitHub repository, including all obtained results:

`https://github.com/sanxlop/tfm_etsit`

Foodiefy web application can be downloaded in ***GooglePlay*** for Android devices and accessible in the link included in the readme file from the previous ***GitHub*** repository for every device with a web browser.

9.3 Conclusions

- Image recognition tasks are generally challenging for machines, while humans do not have any difficulty. Therefore, it is essential to use computational models, such as artificial neural networks that simulate the human brain.
- Building a convolutional neural network architecture for image recognition is a manageable task with Keras, the difficult job is to achieve an architecture with high validation accuracy. Using predefined and tested architectures solve this accuracy dilemma.
- Large datasets with several similar classes need very deep networks to find useful features. Training a very deep model for image classification is a process that requires significant time and costly computation capacity. However, transfer learning technique with pre-trained models solves those difficulties, so that is possible to obtain a model for categorizing 100 classes of similar images with 89% of accuracy in just 28h.
- In the training process, reaching a certain level of accuracy does not take too long, the slow task, is to improve that level.
- Adjusting model training and image generator hyperparameters is a tedious process since it consists of multiple configuration parameters, and it is not easy to find the appropriate values.
- Nowadays, the extraction of data to generate a dataset can become very complex due to the difficulties posed by large social networks. Web scraping techniques facilitate the extraction of data for one's own convenience.
- Designing an accessible application for all devices and adaptable to all platforms can be a problem, however, developing it as a web application, one single design works for all platforms.

9.4 Future lines of work

- Apply transfer learning technique developed on other architectures to generate new models based on other datasets or simply improve the model challenge accuracy.
- Implement the concept of this model in another architecture of recurrent neural networks for object detection.
- Develop another case study using the food image classification model generated, such as a real-time food image classification application with the camera of a device.

APPENDIX A

Project Impact

This appendix shows the social, economic, environmental impact jointly the ethical an professional responsibility. As it is described, the project uses open source datasets and web scraping techniques to acquire data that are the main points to be highlighted.

A.1 Social impact

The individuals and entities affected in this project are the potential users who use the application, the restaurants included in the database and the target website from which the information has been extracted. It can be accessed to the web application from any device with an Internet connection that has a web browser or Android operating system. The developed web application aims to generate social welfare trying to facilitate searches and reach the product in the most efficient way possible.

A.2 Ethical and professional responsibility

In this section, the ethical and professional issues of the technologies used in the project are treated. One ethical concept that applies the whole project is a copyright infringement, that has to do with the dataset sources employed to build the model, that in this case, is open source, and the stored data does not contain any risky information. Talking about technologies it can be highlight one main responsibility topic, web scraping. The different points that determine the legality of scraping [63] a website and the subsequent use of the data extracted are:

- The risk of incurring in a violation of the intellectual property rights of the owners of the website, proving the original structure of the database object of the web scraping, turning it into an intellectually protected work.

- The possible consideration of a conduct of unfair competition when the purpose carried out by those third parties that apply the techniques of web scraping is susceptible to be considered an imitation, by offering services similar to those provided by the website object of scraping, assuming a risk of confusion on the part of the users or the improper use of the reputation or effort of others.

- An eventual violation of the legal terms and conditions of use established by the owners of the website object of scraping, from the moment in which they are accepted by users who browse the web page and have access to the information contained therein.

- The potential breach of the regulations on data protection and the violation of the rights of holders of personal data subject to scraping, as remembered in personal data protection law, requires that you have the consent of the owner of the data at the time of proceeding with the storage and treatment thereof.

A.3 Economic impact

The economic impact is linked mainly to the case study developed as a web application that is the one that could generate some type of income directly like monetizing it. In addition, it can generate value any of the tools developed to generate models of image classification or data extraction. Automatization is one of the main concepts of the project by classifying data automatically and extracting data. Either of the economic models can be viable at the industry, with a relatively low maintenance cost, and with possible improvements and arising business ideas.

A.4 Environmental impact

The development of this project does not have any environmental impact except for the electrical consumption and the necessary technological components. The usual practice of this computational capacity is to use shared resources in the cloud on demand, optimizing consumption in this way, and being an environmentally friendly project.

APPENDIX B

Project Budget

This appendix describes the necessary project budget regarding material resources, human fees, and taxes involved.

B.1 Material resources

The material resources necessary to carry out this project would be the computational capacity necessary to generate the classification models and the servers to deploy the web application.

Since it has been used Google Colaboratory for this project, with free computational capacity, no expense has been required. Although, the equivalent would be the price of a computer with 12 GB of RAM, a 10 GB GPU and a mid-range processor, which corresponds to about 800 €, or rent the equivalent computational capacity.

The hosting server, in this case, has also been chargeless since the application is available on Google Play and access to the web application is hosted on servers of the research group also costless. The equivalent price would be the domain and hosting that is around 20 €/year.

B.2 Human fees

In this section is estimated all human fees necessary to develop the entire project according to the hours proposed by the rules of the master thesis of the university.

As the estimation of this project is 900 hours or equal 37.5 working days according to the ECTS credits of the master thesis, and considering an average salary of a software engineer as 31,000 €/year gross, the total human fees budget to develop the project is around 12,600 €, taking into account that the worker earns 2,583 €/month and to work 900 hours must be working for almost 5 months.

There must be a person in charge of the application maintenance, although this cost has been ignored.

B.3 Taxes involved

The fees paid by the company that finances the project would be the corresponding VAT established in the local country for material resources, while it must also pay the social security and some other taxes of the employees hired.

In the case that a second company wanted to buy the project, the regularization of the local country would come into play again, which in this case corresponds to the tax of a software product.

Bibliography

[1] Beautiful soup — we called him tortoise because he taught us. https://www.crummy.com/software/BeautifulSoup/. (Accessed on 04/18/2019).

[2] Colaboratory. https://colab.research.google.com/notebooks/welcome.ipynb. (Accessed on 03/04/2019).

[3] Font awesome. https://fontawesome.com/. (Accessed on 04/19/2019).

[4] Json. https://www.json.org/. (Accessed on 04/19/2019).

[5] Pandas — python data analysis library. https://pandas.pydata.org/about.html. (Accessed on 04/19/2019).

[6] Phonegap. https://phonegap.com/. (Accessed on 04/19/2019).

[7] scikit-learn — machine learning in python. https://scikit-learn.org/stable/. (Accessed on 04/18/2019).

[8] Selenium — web browser automation. https://www.seleniumhq.org/. (Accessed on 04/18/2019).

[9] Tensorflow. https://www.tensorflow.org/. (Accessed on 02/17/2019).

[10] W3.css framework. https://www.w3schools.com/w3css/. (Accessed on 04/19/2019).

[11] What is a container? — docker. https://www.docker.com/resources/what-container. (Accessed on 05/22/2019).

[12] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[13] Chadia Abras, Diane Maloney-Krichmar, Jenny Preece, et al. User-centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, 37(4):445–456, 2004.

[14] Eduardo Aguilar, Marc Bolanos, and Petia Radeva. Exploring food detection using cnns. In *International Conference on Computer Aided Systems Theory*, pages 339–347. Springer, 2017.

[15] Andrea Asperti and Pietro Battilana. Automatic point-of-interest image cropping via ensembled convolutionalization. *International Journal of Neural Networks and Advanced Applications*, 5:17–24, 2018.

[16] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European Conference on Computer Vision*, pages 446–461. Springer, 2014.

[17] Alex Castrounis. Artificial intelligence. `https://www.kdnuggets.com/2016/10/artificial-intelligence-deep-learning-neural-networks-explained.html`, 10 2016. (Accessed on 03/11/2019).

[18] Xin Chen, Hua Zhou, Yu Zhu, and Liang Diao. Chinesefoodnet: A large-scale image dataset for chinese food recognition. *arXiv preprint arXiv:1705.02743*, 2017.

[19] François Chollet et al. Keras. `https://keras.io`, 2015.

[20] Francois Chollet. Building powerful image classification models using very little data. *The Keras Blog, 05/06/2015*, 2016.

[21] Francois Chollet. *Deep Learning with Python*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2017.

[22] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[23] Gianluigi Ciocca, Paolo Napoletano, and Raimondo Schettini. Cnn-based features for retrieval and classification of food images. *Computer Vision and Image Understanding*, 176:70–77, 2018.

[24] Siddharth Das. Cnn architectures: Lenet, alexnet, vgg, googlenet, resnet. `https://medium.com/@sidereal`. (Accessed on 03/13/2019).

[25] Adit Deshpande. The 9 deep learning papers. `https://adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html`. (Accessed on 03/13/2019).

[26] Ben Frain. *Responsive web design with HTML5 and CSS3*. Packt Publishing Ltd, 2012.

[27] Dave Gershgorn. Nerds rejoice: Google just released its internal tool to collaborate on ai — quartz. `https://qz.com/1113999/`. (Accessed on 03/04/2019).

[28] Venu G Gudise and Ganesh K Venayagamoorthy. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706)*, pages 110–117. IEEE, 2003.

[29] Hamid Hassannejad, Guido Matrella, Paolo Ciampolini, Ilaria De Munari, Monica Mordonini, and Stefano Cagnoni. Food image recognition using very deep convolutional networks. In *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management*, pages 41–49. ACM, 2016.

[30] Simon S Haykin, Simon S Haykin, Simon S Haykin, Kanada Elektroingenieur, and Simon S Haykin. *Neural networks and learning machines*, volume 3. Pearson Upper Saddle River, 2009.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[32] Akinori Hidaka and Takio Kurita. Consecutive dimensionality reduction by canonical correlation analysis for visualization of convolutional neural networks. In *Proceedings of the ISCIE International Symposium on Stochastic Systems Theory and its Applications*, volume 2017, pages 160–167. The ISCIE Symposium on Stochastic Systems Theory and Its Applications, 2017.

[33] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[34] Jeremy Jordan. Common architectures in convolutional neural networks. `https://www.jeremyjordan.me/convnet-architectures/`. (Accessed on 03/13/2019).

[35] Andrej Karpathy et al. Cs231n convolutional neural networks for visual recognition. *Neural networks*, 1, 2016.

[36] Yoshiyuki Kawano and Keiji Yanai. Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In *European Conference on Computer Vision*, pages 3–17. Springer, 2014.

[37] Donald Kinghorn. Gpu memory size and deep learning performance. `https://www.pugetsystems.com/labs/hpc/`. (Accessed on 03/17/2019).

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[39] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[40] Chang Liu, Yu Cao, Yan Luo, Guanling Chen, Vinod Vokkarane, and Yunsheng Ma. Deepfood: Deep learning-based food image recognition for computer-aided dietary assessment. In *International Conference on Smart Homes and Health Telematics*, pages 37–48. Springer, 2016.

[41] Carl G Looney et al. *Pattern recognition using neural networks: theory and algorithms for engineers and scientists*. Oxford University Press New York, 1997.

[42] Pedro Marcelino. Transfer learning pre-trained models. `https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751`, 2018. (Accessed on 03/14/2019).

[43] Niki Martinel, Gian Luca Foresti, and Christian Micheloni. Wide-slice residual networks for food recognition. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 567–576. IEEE, 2018.

[44] Yuji Matsuda, Hajime Hoashi, and Keiji Yanai. Recognition of multiple-food images by detecting candidate regions. In *2012 IEEE International Conference on Multimedia and Expo*, pages 25–30. IEEE, 2012.

[45] Weiqing Min, Shuqiang Jiang, Linhu Liu, Yong Rui, and Ramesh Jain. A survey on food computing. *arXiv preprint arXiv:1808.07202*, 2018.

[46] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *arXiv preprint arXiv:1611.06440*, 3, 2016.

[47] Marie Ng, Tom Fleming, Margaret Robinson, Blake Thomson, Nicholas Graetz, Christopher Margono, Erin C Mullany, Stan Biryukov, Cristiana Abbafati, Semaw Ferede Abera, et al. Global, regional, and national prevalence of overweight and obesity in children and adults during 1980–2013: a systematic analysis for the global burden of disease study 2013. *The lancet*, 384(9945):766–781, 2014.

[48] World Health Organization et al. Diet, nutrition, and the prevention of chronic diseases. report of a who study group. *Diet, nutrition, and the prevention of chronic diseases. Report of a WHO Study Group*, (797), 1990.

[49] Bharath Raj. Data augmentation — how to use deep learning when you have limited data. `https://medium.com/nanonets`. (Accessed on 03/16/2019).

[50] Jaclyn Rich, Hamed Haddadi, and Timothy M Hospedales. Towards bottom-up analysis of social food. In *Proceedings of the 6th International Conference on Digital Health Conference*, pages 111–120. ACM, 2016.

[51] Kiro Risk. Fuse.js — javascript fuzzy-search library. `https://fusejs.io/`. (Accessed on 04/19/2019).

[52] Patrick Rodriguez. stratospark — deep learning. `https://blog.stratospark.com/deep-learning-applied-food-classification-deep-learning-keras.html`. (Accessed on 02/26/2019).

[53] Sharmili Roy, Zhao Heng, Kim-Hui Yap, Alex Kot, and Lingyu Duanb. Visual food recognition for dietary logging and health monitoring. *Learning Approaches in Signal Processing*, page 407, 2018.

[54] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[55] Ankit Sachan. Fine-tuning convolutional neural network on own data using keras tensorflow. `https://cv-tricks.com/keras/fine-tuning-tensorflow/`. (Accessed on 03/14/2019).

[56] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[58] Ashutosh Singla, Lin Yuan, and Touradj Ebrahimi. Food/non-food image classification and food categorization using pre-trained googlenet model. In *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management*, pages 3–11. ACM, 2016.

[59] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[60] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[61] TechnoReview. Artificial neural network: Beginning of the ai revolution. `https://hackernoon.com/artificial-neural-network-a843ff870338`. (Accessed on 03/11/2019).

[62] Marcel van Gerven and Sander Bohte. *Artificial neural networks as models of neural information processing*. Frontiers Media SA, 2018.

[63] Sonia Vazquez. Web scraping: ¿legal o ilegal? — ecija. `https://ecija.com/web-scraping-legal-ilegal/`. (Accessed on 05/19/2019).

[64] Xin Wang, Devinder Kumar, Nicolas Thome, Matthieu Cord, and Frederic Precioso. Recipe recognition with large multimodal food dataset. In *2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6. IEEE, 2015.

[65] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016.

[66] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017.

[67] Ruihan Xu, Luis Herranz, Shuqiang Jiang, Shuang Wang, Xinhang Song, and Ramesh Jain. Geolocalized modeling for dish recognition. *IEEE transactions on multimedia*, 17(8):1187–1199, 2015.

[68] Keiji Yanai and Yoshiyuki Kawano. Food image recognition using deep convolutional network with pre-training and fine-tuning. In *2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6. IEEE, 2015.

[69] Alex Yu. How to teach a computer to see with convolutional neural networks. https://towardsdatascience.com/@alexjy. (Accessed on 03/12/2019).

[70] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

[71] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[72] Wei Zhang, Kazuyoshi Itoh, Jun Tanida, and Yoshiki Ichioka. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Applied optics*, 29(32):4790–4797, 1990.