Master's Thesis

Applying Deep Learning for Food Image Analysis

UNIVERSITAT DE BARCELONA

Gerard Marrugat Torregrosa

Master in Foundations of Data Science

Facultat de Matemàtiques i Informàtica
de la Universitat de Barcelona

Advisor: Petia Radeva

Departament de Matemàtiques i Informàtica
de la Universitat de Barcelona

September 2, 2019

Contents

Chapter 1

Abstract

Food is an important component in people's daily life, examples of the previous assertion are the range of possible diets according to the animal or vegetal origin, the intolerance to some aliments and lately the increasing number of food pictures in social networks. Several computer vision approaches have been proposed for tackling food analysis problems, but few effort has been done in taking benefit of the hierarchical relation between elements in a food image; dish and ingredients.

In this project the highly performing state of the art CNN method is adapted concatenating an ontology layer, a multidimensional layer which contains the relation between the elements, in order to help during the classification process. Different structures for the ontology have been tested to prove which relations have the most beneficial impact, and which are less relevant. Additionally to structure, the value of the elements that compound this hierarchical relation layer play an important role, therefore the experiments performed contained different weighted relations between the components. The ontology layer is built with the labels of the multiple task in the dataset used to train the model. At the end, the results obtained will be compared to a baseline model without the ontology layer and it will be appreciated how hierarchical relations between tasks benefits classification. Finally, the result will be a model which will be able to simultaneously predict two food-related tasks; dish and ingredients.

Chapter 2

Food Analysis Problem

2.1 Inter-class Similarity and Intra-class Variability

Although the potential of food analysis systems is clear, several challenges need to be solved. In particular, if we compare food images to other visual analysis problems like object recognition, food classes have much higher inter-class similarity and intra-class variability, making any food analysis problem very difficult to solve. Considering the particular problems of food-related tasks, ingredients recognition can be of high difficulty considering that some ingredients can be present in several textures and shapes, and others can be invisible. To help the understanding of the previous concepts some examples are provided.

The two pictures below illustrate the inter-class similarity complication. The image on the left corresponds to a dish of chicken with curry, and the right one to spaghetti bolognesa. Curry sauce and tomato sauce present the same shape but do not represent the same ingredient, however they might be confused during detection step due their aspect similarity.

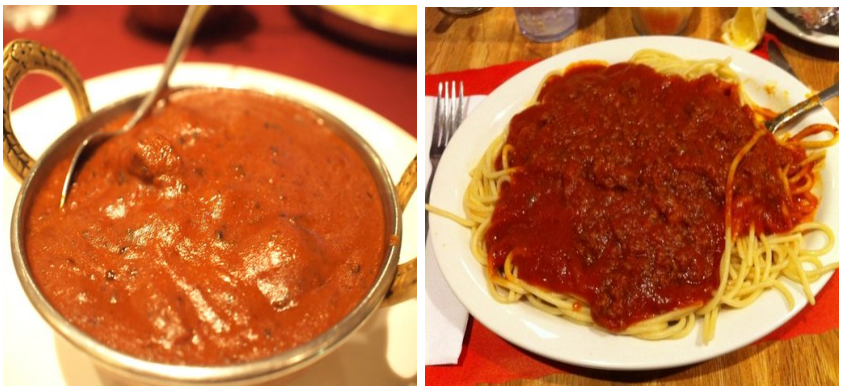


Figure 2.1: Inter-class similarity example 1: Curry Sauce and Tomato Sauce

Another good example of similarity between ingredients could be the next one; tuna tartare and beef tartare. Both represent the same dish but differ in the main ingredient; tuna or beef, which present a dice shape. Distinguishing tuna from beef in this example is a really difficult task for the classification algorithm, even for the human eye could not be feasible to accomplish.

Figure 2.2: Inter-class similarity example 2: Tuna Tartare and Beef Tartare

When talking about intra-class variability it refers to the possible shapes and textures that can manifest an ingredient itself. For example, the following images were taken to two apple pies cooked using different recipes. The first pie contains a cover made of apple slice whereas the pie on the right is filled in with apple compote. On the contrary to inter-class similarity, the ingredient is the same but the difficulty appears on learning the different aspects for the ingredient in question.

Figure 2.3: Intra-class variability example 1: Apple

The last images illustrate how variable can be the aspect of an ingredient. It may depend on the recipe, on the way of cooking, the ability of the cook or how well shown is in the picture among other possibilities. For example, avocado can be mashed to make guacamole, cut in dices to prepare makis and sliced to decor a dish of ceviche.

Figure 2.4: Intra-class variability example 2: Avocado

2.2 Related work

Applying deep learning models to food detection and classification or ingredients classification is part of the state of the art in Food Analysis, several approaches take advantage of pre-trained networks on large-scale datasets. However using external knowledge to build an ontology that provides relations of the elements that appear in the dataset has never been done in this field. Even so, previous studies have used hierarchical structure when dealing with image classification.

Convolutional Neural Networks in Food Analysis

Food Image Recognition: In [4] the author constructs a four-layer network, three convolutional-pooling layers and one fully-connected, as a proof of concept in order to demonstrate that deep learning models outperforms classical computer vision techniques in image recognition. A total of 5822 color images, representing ten-class food items, were collected from the ImageNet database.

Food Ingredients Recognition: Detecting the kind of food is not the only task that can be done in a food image. Based on the recipes of several dishes the authors of [5] trained a multi-label model with the purpose of being able to recognise which ingredients are present in an image, even if they are not visible to the naked eye. Morevoer, they made public a dataset for ingredients recognition built specifically for this work.

Multi-task Learning Model for Food Analysis: The work done in [1] expose the idea of the existent correlation between different tasks in food analysis problems. They propose a multi-task model, able to predict more than one task, e.g. dish and ingredients, with the particularity that they use the uncertainty of each classification task to weigh their corresponding losses as well as their correlations. Furthermore, they propose a Multi-Attribute Food dataset due the lack of these labeled image sets.

Ontology-driven image classifcation

Ontology is the knowledge which organized the data in a hierarchical structure. A hierarchy can have several levels, and the linked elements between levels establish a parent-child relation. These associations can not be learn directly from an image, an external special domain expertise is required. For example, [7] uses prior knowledge to achieve a transfer learning approach of Indian Monuments whether it is seen or not a discriminating region of certain monuments. Ontology is compound of weights which denotes how much related are two elements of different levels. In this case they use ontology weights as a regularization term in their objective function.

Other works have integrated the ontology in its model architecture. [9]

appends a two-layer hierarchy on the top of an Inception-V3. They did two ontology constructions based on ILSVRC2010 dataset; semantic and visual ontology. ILSVRC2010 is organized according to a high-level semantic lexical natural database called WordNet, so they used it to build a two-level ontology, where the high hierarchy level is considered a super-class category and the leaves related to each of them are image classes. On the other side, the visual ontology is build calculating the similarity among images, those of them which result similar are clustered into a non-leaf class, grouping some leaf nodes under it.

Graph Convolutional Neural Networks

Maybe the current project is not strictly related, but when talking about using the relation between entities it is required to talk about Graph Networks. Summarizing a graph is a collection of vertices connected among them. Regarding the notation, G = (V, x, ξ, A), where V is a finite set of vertices with size N, signal x $\in R^N$ - is a scalar for every vertex, ξ is a set of edges, A $\in R^{NxN}$ - is the adjacency matrix, and entry A_{ij} encodes the connection degree between the signals at two vertices. Graph Convolutional Neural Network propagates the input following the rule:

$$H = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l-1)}W^{(l)})$$

Where $\tilde{A} = A + I$, $\tilde{D} \in R^{NxN}$ is a diagonal degree matrix where $D_{ii} = \Sigma_j A_{ij}$, and H^l is the output of the l^{th}-layer, $H^0 = X$ (matrix of feature vectors of each node, input of the Graph Network)

Authors of [3] employed a GCNN for a practical case for a Large-Scale Bike Network to predict the state of each station. The feature vector x contains historical hourly bike demand values at station level, and for the construction of the adjacency matrix, which contains the relations between stations, they suggest two possible ways; pre-defining the matrix or initialize it with unknown parameters and allow fine-tunning during training. For the first approach they built four types of adjacency based on different criteria: spatial distance between stations, bike-demand from one station to another, average trip from one station to another and the demand similarity between stations. On the other hand, approach which they call Data-Driven Graph Filter, there is no need of degree matrix and the propagation operation can be reduced to:

$$H = \sigma(\tilde{A}H^{(l-1)}W^{(l)})$$

Another example of Graph Network related work is [8], which incorporates a semantic concept hierarchy into network construction. Their solution concatenates the graph hierarchy to the output of a CNN, from which the visual features are obtained, and during training phase for each training sample just one subgraph is activated, then the correlations between concepts of different levels are learnt.

Chapter 3

Research and Development

In the first section of this third chapter, before entering into details of the development, it is presented the technical background required to make possible the development of the project. The second part consists of an analysis of the datasets used, the difficulties that appear in these datasets regarding the food analysis and how the previous knowledge is applied to the data of the problem addressed. Finally, the third section contains the explanation of all the creative process; the description of the tools used, the construction of the ontology layer and the definition of the models evaluated.

3.1 Technical Background

3.1.1 Convolutional Neural Networks

In the field of computer vision, the appearance of Convolutional Neural Networks have meant an important innovation. When considering the analysis of an image with a computer it is necessary to define a computational representation of the image. With this purpose, images at the input of a CNN are represented as $n \times m \times d$ array of numbers, called pixels. The d refers to the number of channels in the image; being 3 if it is a RGB image or 1 if it is black and white. Values n, m refers to the height and width of the image in pixels. Each number in this array is given a value from 0 to 255 which describes the pixel intensity at that point. These numbers are the only inputs available to the computer.

To classify an image, this should be passed to the neural network in an array format, as explained before, and the output must be the probabilities of

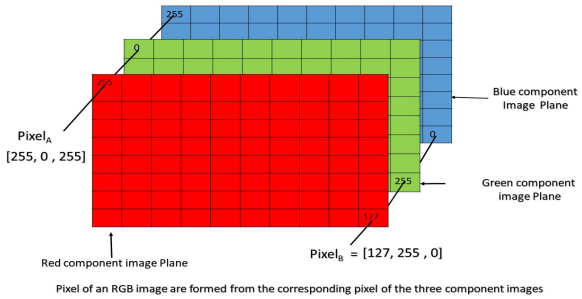


Figure 3.1: RGB channels. *Source: GeeksforGeeks*

the image being a certain class. The classification procedure could be performed by a classical neural network of dense layers, but it has some limitations that the CNN approach overcomes. The main one is that images with a high number of pixels would need a network with extremely large dense layers, and updating its weights would be an inconvenient in terms of time consumption. Another point that should be taken into consideration is the strong relation of a pixel with its neighbours. Moreover, natural images present location invariance property, which means that objects can be present at any place and at any scale in the image. Object location is not important, what is important for attaching a meaning to an image are the relative positions of geometric and photometric structures.

All these considerations leaded to the proposal of **Convolutional Layers**. A Convolutional Layer consists of a set of filters. Each filter is a small matrix(parameters extended along width and height) but extended through the full depth of the input volume. This layer is also called *Kernel*. When an image passes through the Kernel, it is said the Kernel convolves with the image. Convolution operation is basically to slide the filter over the image spatially, computing dot products at any position. As filter slides over the width and height of the input a 2-dimensional activation map is produced, and it gives the response of that filter at every spatial position. This operation allows the network to keep spatial features of its input and as it could be observed, the filter size plays an important role because it connects each neuron to only a local region of the input volume. The spatial extent of this neuron is an hyperparameter of the layer known as local receptive field.

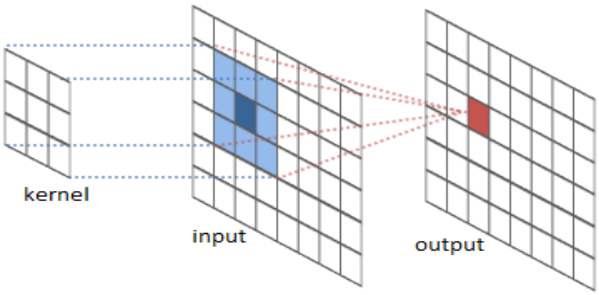


Figure 3.2: Convolutional Kernel. *Source: River Trail documentation*

Besides Convolutional Layers, CNNs are compounded of other types of layers. After each Convolutional Layer it is applied two non linear operations: **activation function**, such as ReLU or sigmoid, which enables the output of each parameter in the activation maps obtained from the previous convolution, and **pooling**, such as MAX or AVERAGE, whose purpose is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Finally, as in ordinary Neural Networks, **Fully Connected Layers** are present in Convolutional Neural Networks. They have full connections to all activations in the previous layer and are used to be the last layers in this kind of network architectures. The reason to use FC layers on top of a CNN is to fix the number of neurons in the last layer to be equal to the number of possible outputs in the classification problem.

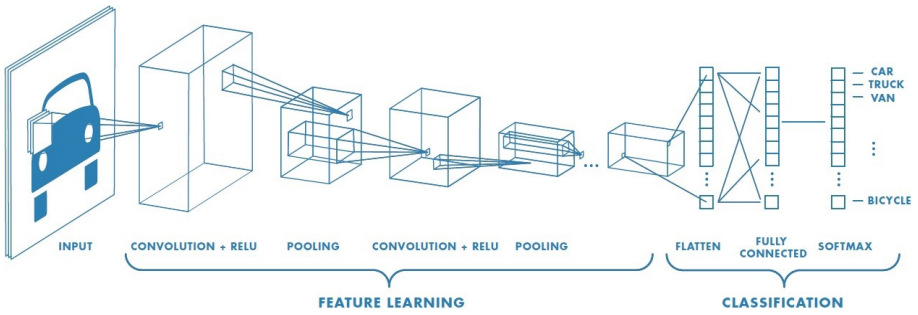
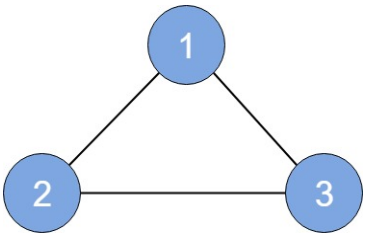


Figure 3.3: Example of CNN architecture. *Source: Medium*

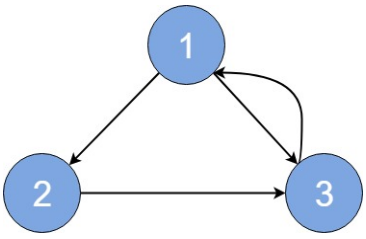
Several important advances have been done in the field of Convolutional Networks, some recognized architectures are **AlexNet** (2012, 8 layers), **VGG** (2014, 19 layers), **Inception** (2014, 22 layers) and **ResNet** (2015, 152 layers).

3.1.2 Graph Representation

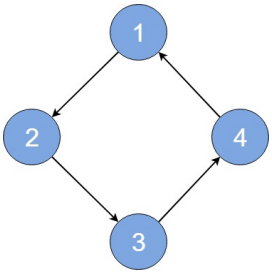
Graphs are structures that represent pairwise relationships between elements. A graph usually represents a flow in a relationship between various objects, moreover it also can represent a hierarchical relation between entities.



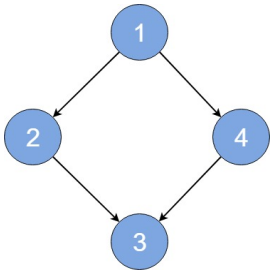
(a) Undirected Graph, all edges are bi-directional



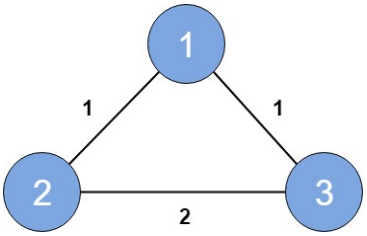
(b) Directed Graph, all edges are uni-directional



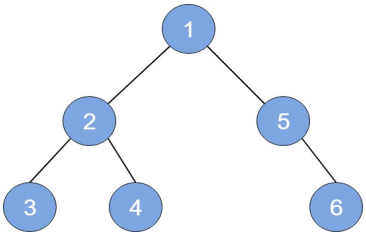
(c) Cyclic Graph, this kind of graph contains a path, called cycle, that starts and ends at the same vertex



(d) Acyclic Graph, it is a graph that has no cycle



(e) Weighted Graph, each edge has a weight



(f) Tree, it is an undirected graph in which any two vertices are connected by only one path. Each node has exactly one parent

Figure 3.4: Types of Graphs

Graphs are compounded by nodes and edges. Nodes are entities whose relationships are expressed using edges. Regarding the typology of a node, it can be **root node**, which is the ancestor of all other nodes and it does not have any ancestor, or **leaf node**, that represent the nodes that do not have any successors. Edges are the components used to link two nodes in a graph. An edge between two nodes expresses an unidirectional or bidirectional relationship between the nodes. Depending on the kind of edges present in the graph, they determine the type of graph.

Going into specific details, the type of graph required in this development is a weighted Directed Acyclic Graph, and the particularity of these graphs is the **topological ordering**, an ordering of the vertices such that the starting endpoint of every edge occurs earlier in the ordering than the ending endpoint of the edge.

3.2 Food Data Analysis

3.2.1 Data Understanding

In this section the datasets used for the experiment are described and analysed. An important limitation of food classification approaches is that there is no dataset that covers all existing recipes worldwide.

dish	total images
apple_pie	49
croque_madame	49
paella	49
gyoza	46
crab_cakes	50

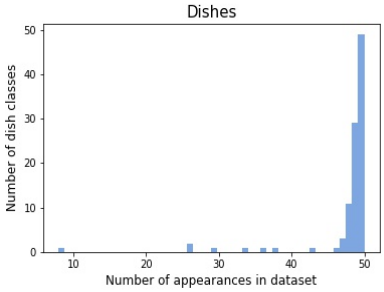
Figure 3.5: Head of Recipes5K dishes dataframe

Recipes5K is a dataset created for ingredient recognition task. It is composed by 4,826 unique recipes extracted from Food101 database with its associated images and ingredients list. It contains 3,123 unique ingredients for 101 dishes, but many of the ingredients are sub-classes (e.g. 'sliced tomato' and 'tomato sauce') of more general class (e.g. 'tomato'), thus, the authors proposed a simplified version by removing the overly-decriptive parts resulting in 1,013 ingredients. Due the ingredient labels are obtained from the recipes, invisible ingredients are also labeled.

Trying to understand how balanced is the dataset it is difficult to visualize correctly how many images per dish there are due the large number of different dishes, 101 in the Recipes5K case. As a solution the alternative proposed is as follows.Creating a dataframe with the number of images per each dish and visualize the histogram over it. In this way, it could be seen the dish class distribution per number of images. Ideally the

graph would be just a bin with a value equal to the total number of different dishes, 101.

Statistic measures over the images per dish dataframe shows that the first quartile of our dataset is 49, which means that the 25% of our dishes appears 49 or less times. Another perspective of this situation would be that the 75% of our dishes appears at least 49 times, and considering that the maximum number of appearances is 50, dish classes are quite well balanced.



(a) Histogram of dish appearances in dataset

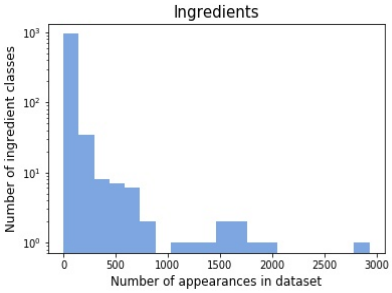
statistical measure	total images
count	101.0
mean	47.78
std	6.02
min	8.0
25%	49.0
50%	49.0
75%	50.0
max	50.0

(b) Statistical measures over dish appearances

Figure 3.6: Analysis of Recipes5k dish classes

Analogously and following the same procedure, the same analysis was done as well over the ingredients, the simplified version, in the dataset.

Here the unbalancing between ingrdient classes is clear. The 50% appears 3 times as much, while the upper 25% of ingredient population is over 17 times.



(a) Histogram of ingredient appearances in dataset

statistical measure	total images
count	1013.0
mean	47.54
std	193.28
min	1.0
25%	1.0
50%	3.0
75%	17.0
max	2933.0

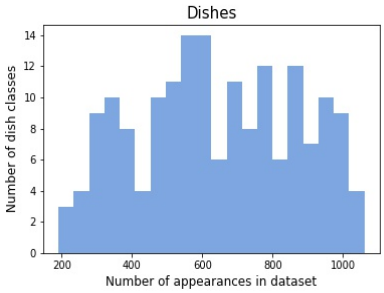
(b) Statistical measures over ingredient appearances

Figure 3.7: Analysis of Recipes5k ingredient classes

VireoFood-172 [2] is a multi-task dataset consisting in 172 popular chinese dishes collected by Baidu and Google image search. For the ingredients, it is only labeled the visible ingredients, contrarily to Recipes5K, resulting in a total of 353 ingredient labels.

As it was done with Recipes5k, an analysis of the number of dish and ingredient appearances on the dataset was performed.

Regarding dishes, the analysis results shows how unbalanced is VireoFood172 dataset. The histogram below displays the scattering of number of dish appearances. Looking at the statistical measures table next to it, specifically the three quartiles, it can be appreciated that the first 25% of our dish population is under the 480 appearances, the next 25% between 480 and 624, the third quarter between 624 and 824, and the remaining 25% between 824 and 1061 appearances.



(a) Histogram of dish appearances in dataset

statistical measure	total images
count	172.0
mean	640.94
std	224.01
min	191.0
25%	480.0
50%	624.0
75%	824.75
max	1061.0

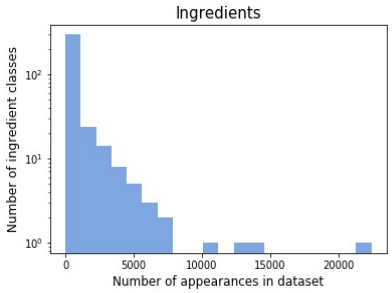
(b) Statistical measures over dish appearances

Figure 3.8: Analysis of VireoFood-172 dish classes

ingredient	total images
Minced green onion	22438
Pork chunks	1116
Seared green onion	2425
Crushed pepper	14261
Spiced corned egg	420

Figure 3.9: Head of VireoFood-172 ingredients dataframe

Considering ingredients, the range of number of appearances goes from 6 to 22438, wider than the Recipes5k ingredients (from 1 to 2933) due the larger size of the database. In view of the results plus the previous knowledge on ingredient appearances equality, it is clear the unbalancing of this task over the dataset. Just to exemplify the previous conclusion, the ingredient appearances dataframe shows the 420 appearances of the spiced corned egg contrasting with the 22438 and 14261 of minced green onion and crushed pepper respectively.



(a) Histogram of ingredient appearances in dataset

statistical measure	total images
count	353.0
mean	940.46
std	1974.54
min	6.0
25%	93.0
50%	437.0
75%	871.0
max	22438.0

(b) Statistical measures over ingredient appearances

Figure 3.10: Analysis of VireoFood-172 ingredient classes

3.2.2 Problem

Summarizing the previous data understanding part, it can be detected one of the complications when dealing with food images. It could be generalized for any dataset that ingredient detection is a tough task due to the strong presence of common ingredients in many dishes, like salt, oil or sugar, and the lack of rare ones, like panko, arame or dukkah.

Other problems, which have been already explained in this report, are the inter-class similarity and intra-class variability. As a recap, inter-class similarity is the similar aspect that different ingredients can present, and intra-class variability is the different shapes and textures the same ingredient can exhibit.

Finally, the last problem is the limited visibility, even invisibility, of some ingredients in an image. It makes difficult to the algorithm the detection of the ingredient in question.

3.2.3 Hypothesis

Taking as starting point models based on Convolutional Neural Networks, which classify images just using visual features, the idea of including non visual information arose. Thanks to the presence of multi task labels, dish and ingredients, in the datasets considered, the approach proposed was to find the relation between these task labels; Dish-Ingredient, Ingredient-Dish and Ingredient-Ingredient relation. In the next section more details will be provided about it.

Thinking on this approach as an improvement to reduce the negative impact of food analysis complications mentioned before, the result expected is an

increase in dish accuracy and in ingredient precision. Specifically, when there is a dish image with an infrequent ingredient label, the Dish-Ingredient and the Ingredient-Ingredient will help to make higher the probability of the rare ingredient. In case the algorithm confuses two ingredient, inter-class similarity, the relations will help to understand that the given dish image does not contain the mispredicted ingredient and it does the undetected one. For the invisibility difficulty, if there is a relation between a non visible ingredient with the rest of components of the picture, this may help in detecting even the ones that are hard to see.

3.3 Model Proposal

3.3.1 Software

To do the development of all model prototypes the high-level neural networks API Keras was used. According to the Keras website, it is an API written in Python, compatible with Python versions from 2.7 to 3.6, that can be run on top of TensorFlow, CNTK, or Theano. The backend for the present project was TensorFlow library. The advantadges that Keras offers and fulfilled the coding requirements of this development.

- Easy and fast prototyping, which enables a fast experimentation.
- Supports convolutional networks and also provides implemented layers for the development of these.
- Runs seamlessly on CPU and GPU.

3.3.2 Model Architecture

The base of the architecture of the model proposal is a pre-trained InceptionResNet. The weights of these previously trained network were obtained over ImageNet dataset[6]. Two dense layers, one per task, follows the CNN, and finally a layer which contains all the relations. This last layer, from now on called Ontology layer, consists on a squared matrix with dimensions equal to the number of different elements in the dataset. The output of this last layer will be an array of length equal to the dimensions of the matrix, thus the output is split in two; one array of dimension equal to number of dishes, and another equal to number of ingredients.

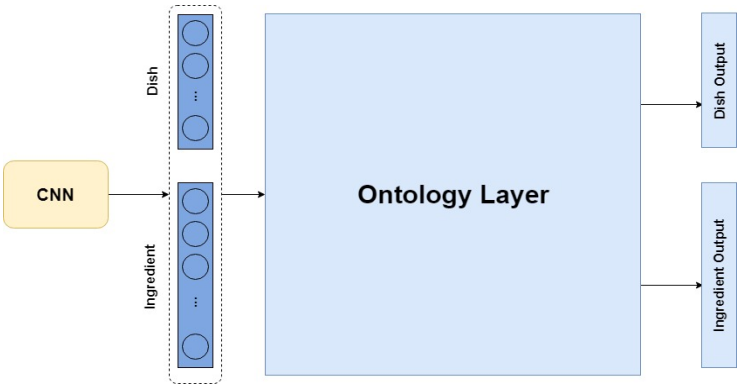


Figure 3.11: Model Proposal Architecture

3.3.3 Types of values in the Ontology layer

Each element on the Ontology matrix contains a value which defines if there is relation between the row item and column item. Different ideas on values that fill the matrix have been performed.

Ontology made of ones and zeros: when relation exists between two elements, it is represented with a one, on the contrary, if the two items in question does not have any link, there is a zero in the corresponding matrix position.

Ontology made of probabilities: this proposition maintains the zero value when does not exist connectivity, but instead of having the value one when an element is related to another, a weight value substitutes it. These weight values are probabilities obtained in a frequentist way from information in the dataset.

Probability of an ingredient given a dish:

$$w_{ij}^{dish-ingr} = \frac{\text{times ingredient j in dish i}}{\text{times recipe of dish i appears in database}}$$

Probability of a dish given an ingredient:

$$w_{ij}^{ingr-dish} = \frac{\text{times ingredient j in recipe of dish i}}{\text{times ingredient j appears in database}}$$

Ingredient coexistence probability:

$$w_{ij}^{ingr-ingr} = \frac{\text{times ingredient i and ingredient j appear together}}{\text{total number of pairs of ingredients}}$$

Ontology made of ones and negative ones: when there is a link between two items, the matrix element contains a one, like in the first ontology case. The characteristic change comes when two entities are not related. To help the model on learning when there is no relation, the idea of using negative values appeared. Then, in the current ontology, negative ones denotes no relation between elements.

Ontology made of probabilities and negative probabilities: The probabilities that relates two concepts are calculated as in the second ontology context, but now the links that represent no relation are intended to be small negative values. The proposal in this work is to assign the negative inverse of the total times a concept appears when no relation exists. Actually these values are not probabilities, they are just small values that describe disconnection between elements.

$$\textit{Negative probability of concept } i = \frac{-1}{\text{total number of times concept i appears}}$$

3.3.4 Ontology structures

Moreover the values contained in the ontology, it should be considered the structure of the layer. The possible formats differ one from another in which relations are included on the matrix. The intention of building various structures is to test all the possibilities and examine which relations benefits best the purpose of the experiment.

Following assignations will make the explanation easy to understand:

n_1 = number of dishes
n_2 = number of ingredients

From row one to row n_1 and from column one to column n_1, the ontology matrix contains **Dish-Dish (D-D)** relation values, which is in fact an identity matrix due that there are no relations between dishes. Sub-matrix $[1 : n_1, n_1 : n_1{+}n_2]$ is compounded by **Dish-Ingredient (D-I)** weights, and $[n_1 : n_1{+}n_2, 1 : n_1]$ is assembled with **Ingredient-Dish (I-D)** values. The remaining elements in sub-matrix $[n_1 : n_1 + n_2, n_1 : n_1 + n_2]$ includes the relation **Ingredient-Ingredient (I-I)**.

Following figure summarizes in a visual way the configuration of the ontology matrix. Below this paragraph, the expressions for the different structures tested are written. Be aware of the matrix values corresponds to the case of an ontology made of probabilities.

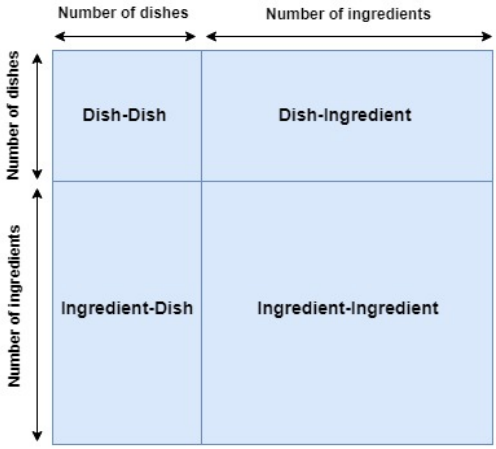


Figure 3.12: Ontology structure

Dish-Ingredient

$$W = \begin{cases} 1, & \text{if } i = j, \;\; i \leq n_1 \\ w_{ij}^{dish-ingr}, & \text{if } i \leq n_1, \;\; n_1 < j \leq n_1 + n_2 \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

where $w_{ij}^{dish-ingr} = \frac{\text{times ingredient j in dish i}}{\text{times recipe of dish i appears in database}}$

Ingredient-Dish

$$W = \begin{cases} 1, & \text{if } i = j, \;\; i \leq n_1 \\ w_{ij}^{ingr-dish}, & \text{if } j \leq n_1, \;\; n_1 < i \leq n_1 + n_2 \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

where $w_{ij}^{ingr-dish} = \frac{\text{times ingredient j in recipe of dish i}}{\text{times ingredient j appears in database}}$

Dish-Ingredient Ingredient-Dish

$$W = \begin{cases} 1, & \text{if } i = j, \;\; i \leq n_1 \\ w_{ij}^{dish-ingr}, & \text{if } i \leq n_1, \;\; n_1 < j \leq n_1 + n_2 \\ w_{ij}^{ingr-dish}, & \text{if } j \leq n_1, \;\; n_1 < i \leq n_1 + n_2 \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

Ingredient-Ingredient

$$W = \begin{cases} 1, & \text{if } i = j, \;\; i \leq n_1 \\ w_{ij}^{ingr-ingr}, & n_1 < i, j \leq n_1 + n_2 \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

where $w_{ij}^{ingr-ingr} = \frac{\text{times ingredient i and ingredient j appear together}}{\text{total number of pairs of ingredients}}$

Dish-Ingredient Ingredient-Ingredient

$$W = \begin{cases} 1, & \text{if } i = j, \;\; i \leq n_1 \\ w_{ij}^{dish-ingr}, & \text{if } i \leq n_1, \;\; n_1 < j \leq n_1 + n_2 \\ w_{ij}^{ingr-ingr}, & n_1 < i, j \leq n_1 + n_2 \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

Full Ontology

$$W = \begin{cases} 1, & \text{if } i = j, \;\; i \leq n_1 \\ w_{ij}^{dish-ingr}, & \text{if } i \leq n_1, \;\; n_1 < j \leq n_1 + n_2 \\ w_{ij}^{ingr-dish}, & \text{if } j \leq n_1, \;\; n_1 < i \leq n_1 + n_2 \\ w_{ij}^{ingr-ingr}, & n_1 < i, j \leq n_1 + n_2 \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

where all the weights are calculated as previously seen.

Focusing on how the ontology layer interacts with the outputs of the dense layers concatenated to the pre-trained InceptionResNet, the operation which is implemented is a dot product between two vectors; the merged outputs of dish and ingredients dense layers, and each of the columns in the relations matrix.

The large part of the results have been obtained applying the dot product, but advanced experiments were developed in order to contrast the possibility

of other operations. Two variants of a new version were tested. Initially, the outputs of the dense layer are passed to probabilities adding the corresponding activation layers before the ontology. Then, instead of the dot product, element wise product is performed, and the result is a matrix with dimension equal to the number of concepts. At this point is when appear the two variations to calculate the final output of the model, the first one calculates the average of each concept row and add it to the corresponding probability value obtained at the output of the preceding dense layers. The second method finds the minimum of each row and add it to the previous probabilities. This last version only makes sense to be applied when there are negative elements in the ontology matrix that denotes no relation. The reason is to soften the negative output and not obtained extremely negative values that destabilize the output model.

Returning to the dot product version, a new idea about the use of ontology layer arose, adding another layer, but this one without its values fixed. With this new idea it could be seen if adjusting the weights like in a common trainable layer improves the output performance of our model. Both layers were initialized with the same ontology structure and components. Finally, and considering the case of a new scenario, it was desired to know what would happened if the two layers were trainable. And this is the last case of applying ontology to a food analysis multi-task model.

Chapter 4

Results and Evaluation

Experimental setup

The CNN architecture was trained using the categorical cross-entropy loss for the single label task, dish classification, and binary cross-entropy loss for the multi label task, ingredient classification. The total number of output neurons depend on the dataset (Recipes5k, VireoFoof-172) as well as the dimensions of the ontology layer. As it was mentioned before, the core of the model is a pre-trained During the training process data augmentation was applied to the train set. The optimizer used was Adam and the learning rate was set to $1e-3$, but the rest of hyperparameters vary for the different experiments done during the project.

Results

In order to evaluate the performance of the new approach on Recipes5k, it was decided to select ten different dishes and starting with a toy problem to see if the proposal overcomes the multi-task learning model baseline. The operation performed by the ontology with the result of previous layers was a dot product. In total, fifteen ontology models were trained during 50 epochs using a batch size of 8.

The metrics used to evaluate the results are *Accuracy* for the SL task (Dishes) and *F1-score* in case of ML (Ingredients). The *F1-score* value depends on the *Precision* and *Recall* measures, and for an unbalanced dataset, which is the case of the ingredients, these metrics shows a better coherence of the results. The table above shows the results for all combinations of ontology structure and the values that compound it. It can be observed that the configurations with better results are Dish-Ingredient and Ingredient-Ingredient, even in the case of D-I structure made of probabilities it overcomes the MTL base-

	No Ont		Ontologies					
			1,0		1,-1		Probs	
	Dish Acc	*Ingr F1*	*Dish Acc*	*Ingr F1*	*Dish Acc*	*Ingr F1*	*Dish Acc*	*Ingr F1*
MTL	73,75%	60,54%	-	-	-	-	-	-
D-I	-	-	78,75%	57,68%	76,25%	55,61%	75,00%	61,30%
I-D	-	-	67,50%	32,60%	72,50%	27,60%	70,00%	44,40%
D-I I-D	-	-	75,00%	42,23%	66,25%	37,28%	73,75%	46,22%
I-I	-	-	75,00%	58,91%	68,75%	57,70%	77,50%	59,21%
Full	-	-	67,50%	54,05%	61,25%	54,49%	75,00%	41,46%

Table 4.1: Results on Recipes5k dataset.

line performance. However, in the view of the results for the ontologies made of ones and negative ones, it denotes the poor conduct of these layers versions. To deep more into the details of the results, the graph below shows how evolves the ingredient F1-score over the test set for each of the models presented in the previous table during the training phase.

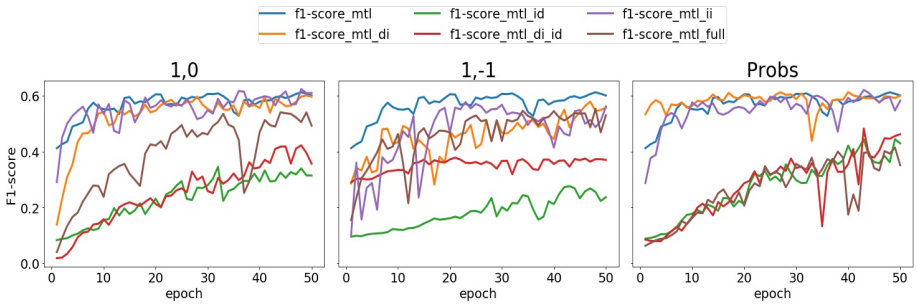


Figure 4.1: F1-score results for ingredients label on Recipes5k

It exemplifies more clearly what was said before, the one and negative ones matrices evidence a bad performance, and regarding the structures D-I and I-I compositions help the base network to make better predictions.

The subset contained 495 images in total, 349 for train, 66 for validation and 80 for test set. Independently of the small selected group of images, the mean number of images per class in Recipes5k is 47, which was considered not enough data for learning. With the purpose of repeating the experiments with a more complete food dataset, and continuing with the development of a model that uses relations between the elements in the dataset, it was decided to use VireoFood-172.

Summarizing, what it has been seen with Recipes5k and what will help with new probes is which structure configurations are more useful and which weight values best accomplish the project goal.

In order to go further with the search of the best model, the experiments on VireoFood-172 dataset were done using ontologies made of ones and zeros, probabilities and probabilities and negative probabilities, a new type. The ones and negative ones variants in the view of their results were not considered then. The operation performed by the ontology with the result of previous layers was a dot product. In total, fifteen ontology models were trained during 10 epochs using a batch size of 16.

			Ontologies					
	No Ont		1,0		Probs		Probs & probs neg	
	Dish Acc	*Ingr F1*	*Dish Acc*	*Ingr F1*	*Dish Acc*	*Ingr F1*	*Dish Acc*	*Ingr F1*
MTL	84,03%	67,50%	-	-	-	-	-	-
D-I	-	-	84,84%	51,26%	**85,45%**	**67,65%**	**84,82%**	**68,31%**
I-D	-	-	81,81%	15,57%	83,45%	57,14%	83,72%	57,56%
D-I I-D	-	-	81,31%	33,46%	83,19%	52,10%	83,90%	58,77%
I-I	-	-	84,06%	67,39%	84,60%	58,66%	**84,92%**	**68,15%**
Full	-	-	83,19%	53,60%	82,89%	42,54%	84,33%	62,41%

Table 4.2: Results on VireoFood-172 dataset.

In this table, the boldface values are the results above the ones obtained with the standard multi-task model. Specifically, the ontology made of probabilities configured with Dish-Ingredient structure obtains a Single Label Accuracy 1,42% higher in respect of the MTL, but just an improvement of the 0,15% for the Multi Label F1-score. Moreover, the Dish-Ingredient configuration whose values are probabilities and negative probabilities improves in a 0,79% the SL Accuracy and a 1,19% the F1-score. Finally, Ingredient-Ingredient structure built with positive and negative probabilities also shows a good performance; 0,89% and 0,65% above for the SL Acc and ML F1 respectively. Paying attention to these results, it could be claimed that the top-down relation between dish and ingredient, and the coexistence between two ingredients are the relations that benefit the most in the food classification task. In light of the previous assumption, another possible version of the ontology layer would be a combination of these two previous structures. Therefore, the experiments are repeated for the new structure with the same hyper-parameters configuration but this time omitting the ones and zeros relations.

	Ontologies			
	Probs		Probs & probs neg	
	Dish Acc	*Ingr F1*	*Dish Acc*	*Ingr F1*
D-I I-I	83,09%	59,70%	**85,22%**	**68,93%**

Table 4.3: Results on VireoFood-172 dataset.

As a consequence of these new attempt, it is obtained what is the best model up to the moment in terms of Ingredient F1-score, achieving a value of 68,93%, 1,43% above the baseline.

The plots on the right shows how the Dis-Ingredient Ingredient-Ingredient Ontology model impacts on the precision of certain dishes and ingredients. These are just some examples of the results, and unfortunately not all the components in the dataset have improved their detection. Exactly, the new model increase the prediction precision in 94 of the 172 dishes, and in 150 of the 353 ingredients.

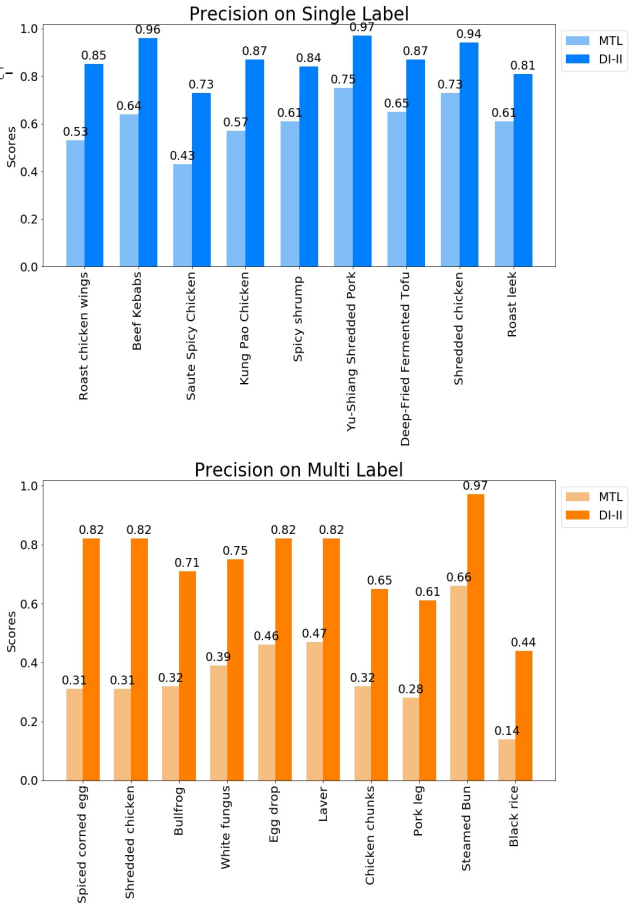


Figure 4.2: Comparison between MTL baseline and DI-II ontology model precision results

All these results are quantitative results, to demonstrate the abilities acquired by the model thanks to the use of an ontology it would be required qualitative results. For these reason, following these lines some images have been taken and passed to the Multi-task baseline model to see which are its predictions. In a similar way, it has been done with DI-II ontology made with positive and negative probabilities. To compare which model predicts better, the ground truth of each image is provided as well. Predictions include dish and ingredient class.

GT	MTL	DI-II
Dish: Yu-Shiang Shredded Pork **Ingredients:** Minced green onion, Crushed pepper, Black fungus, Shredded pork, Shredded bamboo shoots	**Dish:** Yu-Shiang Shredded Pork **Ingredients:** Shredded pork, Shredded bamboo shoots, Lentinus edodes slices, Dried pieces of bean curd, Mutton slices	**Dish:** Yu-Shiang Shredded Pork **Ingredients:** Shredded pork, Black fungus, Julienned carrot, Shredded pepper, Minced green onion

GT	MTL	DI-II
Dish: Kung Pao Chicken **Ingredients:** Minced green onion, Groundnut kernels, Crushed hot and dry chili, Brunoise diced chicken	**Dish:** Sautéed Sweet Corn with Pine Nuts **Ingredients:** Corn kernels, Pine nuts, Brunoise diced cucumber, Brunoise diced chicken	**Dish:** Kung Pao Chicken **Ingredients:** Brunoise diced chicken, Groundnut kernels, Minced green onion, Crushed hot and dry chili

GT	MTL	DI-II
Dish: Roast chicken wings **Ingredients:** Pepper slices, Chinese Parsleycoriander, Crushed hot and dry chili, Hob blocks of potato, Chicken chunks	**Dish:** Saute Spicy Chicken **Ingredients:** Bullfrog, Garlic clove, Chicken chunks, Chinese Parsleycoriander, Hob blocks of potato	**Dish:** Saute Spicy Chicken **Ingredients:** Hob blocks of potato, Chinese Parsleycoriander, Chicken chunks, Crushed hot and dry chili, Brunoise diced chicken

Finished with a round of experiments in which a successful new model was achieved, and thus the hypothesis about the benefits of using the relations between elements in a database is demonstrated. At this point other alternatives may be considered in order to open new lines of investigation and contrast with the good direction of the project that denotes the results.

A reasonable option could be trying with two ontology layers instead of one, then the size architecture of the model would be increased. But doubling the layer with no modification on it may makes no sense, so the idea that emerged was to make trainable the second layer. And to conclude with the two ontology models, an advance version which allowed the update of the weight in both layers was also trained. The experiment considered the best models reported; structures Dish-Ingredient, Ingredient-Ingredient and the combination Dish-Ingredient Ingredient-Ingredient whose elements are probabilities and positive and negative probabilities. In total, 8 ontology models were trained during 10 epochs using a batch size of 16.

	Ontologies			
	Probs		Probs & probs neg	
	Dish Acc	*Ingr F1*	*Dish Acc*	*Ingr F1*
D-I (one trainable layer)	83,52%	65,22%	82,35%	65,21%
I-I (one trainable layer)	-	-	82,94%	64,43%
D-I I-I (one trainable layer)	-	-	82,00%	64,81%
D-I (two trainable layers)	82,88%	63,74%	83,06%	65,43%
I-I (two trainable layers)	-	-	82,93%	64,25%
D-I I-I (two trainable layers)	-	-	83,30%	65,76%

Table 4.4: Results of two layer ontology models on VireoFood-172 dataset.

As it can be appreciated in the table the metrics values obtained are not bad results, however they do not represent a better option to think about in comparison with the one ontology layer models. This architecture with two layers could be interesting for datasets with more than two tasks.

To finish with the experimental part, an approach with another kind of interaction of the ontology with the previous layer. Remember that all models developed until this moment have used the dot product operation between ontology matrix and its input. Now, it is an element wise product between the output probabilities of the CNN and the ontology layer, followed by a row average calculation or find the row minimum, to finally add it to the previously obtained probability. In total, 6 ontology models were trained during 50 epochs using a batch size of 16. Due that the number of epochs was different that time, it was necessary to train an MTL baseline during 50 epochs.

			1,-1 (avg)		1,-1 (min)		Probs & probs neg (avg)	
	Dish Acc	*Ingr F1*	*Dish Acc*	*Ingr F1*	*Dish Acc*	*Ingr F1*	*Dish Acc*	*Ingr F1*
MTL	85,11%	70,64%	-	-	-	-	-	-
D-I	-	-	85,49%	70,72%	85,64%	70,28%	85,09%	70,72%
D-I I-I	-	-	-	-	-	-	85,53%	70,84%
I-I	-	-	-	-	-	-	85,43%	71,27%
Full	-	-	-	-	-	-	85,05%	68,1%

Table 4.5: Results of average and minimum ontology operations on VireoFood-172 dataset.

These new results seems to overcome the bests obtained until now, but it could be a mistake claiming that this is the new best model because these values have been obtained training during 50 epochs while the other experiments were done in 10. A future step to compare fairly the performance of different layer operations would be training the dot product models during 50 epochs.

Chapter 5

Conclusions and Future Lines

This Master project was developed with the initial hypothesis that using the relational information between the elements in a food picture, dish and ingredients, could improve the performance of the exceptional convolutional networks. Many experiments have been completed and contrasting their results with the initial assumption some conclusions can be made.

Six different ontology structure types have been implemented and subsequently evaluated; Dish-Ingredient, Ingredient-Dish, Dish-Ingredient Ingredient-Dish, Ingredient-Ingredient, Dish-Ingredient Ingredient-Ingredient and all of them together (Full). With the results obtained it can be claimed that Dish-Ingredient and Ingredient-Ingredient relations are the two that have the most positive impact, and combining the both of them the best Ingredient F1-score is achieved.

During the experiments it has been seen that the no relation between elements could symbolize another way to boost the model learning. A proposal of using very small negative values to denote this lack of association between dishes and ingredients has revealed an improvement on the performance.

Due the fact that there is no dataset that contains all the dish recipes in the world, the ontology layer might be limited to the dataset in question. However if an image of a dish external to the database is passed to the input of the ontology model, it can be able to detect ingredients and benefit the detection of other highly likely coexisting ingredients.

Regarding the possible future lines to improve this project, the approach

with small negative values in the ontology matrix could be considered. This initial idea of using the negative inverse of the total number of concept appearances in the dataset needs to be polished, a definition with a more theoretical background might benefit.

The attempt of using two ontology layers may have a positive effect on datasets with more than two tasks, having an ontology layer for each pair of tasks.

Finally, the models that performs the average or minimum operation over the result of element wise product between CNN output and ontology are not comparable to the rest of models because they were trained during more epochs. So, the best dot product models could be trained with a larger number of epochs, in this way it could be seen if the performance improves investing more time in training and at the same time all models would be comparable allowing to decide which operation is more appropriate.

References

[1] Marc Bolaños Eduardo Aguilar and Petia Radeva. "Regularized uncertainty-based multi-task learning model for food analysis". In: *Journal of Visual Communication and Image Representation* 60 (), pp. 360–370.

[2] Chong-wah NGO Jing-jing Chen. "Deep-based Ingredient Recognition for Cooking Recipe Retrival". In: *ACM Multimedia* (2016).

[3] Zhengbing He Lei Lin and Srinivas Peeta. *Predicting Station-level Hourly Demand in a Large-scale Bike-sharing Network: A Graph Convolutional Neural Network Approach*. URL: https://arxiv.org/ftp/arxiv/papers/1712/1712.04997.pdf.

[4] Yuzhen Lu. *Food Image Recognition by Using Convolutional Neural Networks(CNNs)*. URL: https://arxiv.org/pdf/1612.00983.pdf.

[5] Aina Ferrà Marc Bolaños and Petia Radeva. *Food Ingredients Recognition through Multi-label Learning*. URL: https://arxiv.org/pdf/1707.08816.pdf.

[6] Princeton University Stanford Vision Lab Stanford University. *ImageNet*. URL: http://www.image-net.org/.

[7] Santanu Chaudhury Umang Gupta. "Deep Transfer Learning with Ontology for Image Classification". In: *Institute of Electrical and Electronics Engineers* ().

[8] Hongfei Zhou Xiaodan Liang and Eric Xing. *Dynamic-structured Semantic Propagation Network*. URL: https://arxiv.org/pdf/1803.06067.pdf.

[9] Y. Qu Y. Zhang and C. Li et al. "Ontology-driven hierarchical sparse coding for large-scale image classification". In: *Neurocomputing* ().