Food For Machine Learning Dataset

Fulop Bettina-Elena
Faculty of Exact Sciences and Engineering,
Advanced Programming and Database
"1 Decembrie 1918" University
of Alba Iulia,
Alba-Iulia, Romania.
bettynaelena@gmail.com

Cristea Ioan
Faculty of Exact Sciences and Engineering,
Advanced Programming and Database
"1 Decembrie 1918" University
of Alba Iulia,
Alba-Iulia, Romania.
five9652@gmail.com

April 24, 2020

Abstract

We introduce and describe a dataset with 1281 images of 423 foods taken from 3 angles. We also present the strengths and weaknesses of our database and encourage researchers to use it for testing the image analysis algorithms. We created a simple neural network for recognizing the images in the dataset.

Contents

1 Introduction

We introduce a new dataset of images containing some of the most common Romanian food dishes.

The dataset is named **Food For Machine Learning Dataset** (Food For Machine Learning) and can be downloaded from the addresses pointed by references [1] and [2], and it is constantly updated with images of new foods as soon as possible.

Unlike the datasets mentioned in section 2, the FFML covers a larger variety of foods, 423 to be more exact, divided into main courses, international dishes, traditional dishes, desserts (birthday cakes, sweets, ice cream, cookies), snacks, soups, salads, fast-foods, grills, garnishes, aperitifs and pizzas.

Instead of just covering a single category of foods, we choose to gather as many as we could get our hands on, that include all types of ingredients, in different forms: fruits, vegetables, meat, sauces, carbohydrates, pasta and so on.

As a second reason why our dataset is of a better quality is because all the images have been taken by us, carefully curated, modified to remove all the background noise and resized to a 512x512 format. This is a big plus, because most other datasets contain images that have both the dishes and the surrounding objects (like plates and tables) in them. If the background is clean, then an algorithm have a higher chance of detecting the correct type of food even if the surrounding objects are different.

The paper is structured as follows: Section 2 describes other existing datasets with images of foods. Section 3 explains, in great detail, the ways in which the images of the dataset were selected, processed and stored. In section 4 we have described a simple neural network that uses the TensorFlow library for recognizing the images in the proposed dataset. Section 5 concludes our paper.

2 Related datasets

This section describes similar attempts of creating databases with food images. Also mentions some attempts to train them.

Paper [3] contains a dataset of 101 standardized fast food categories, randomly sampling 750 images from `foodspotting.com`, and an additional 250 test images collected from each class, that were manually cleaned.

Paper [4] mentions the use of three image datasets, the first one (named *Food-5K*) containing 2,500 food images and 2,500 non-food images, the second one (named *Food-11*) 16,643 images grouped into 11 categories (Bread, Dairy Products, Dessert, Egg, Fried food, Meat, Noodles/Pasta, Rice, Seafood, Soup and Vegetable/Fruit), and the third one (named *IFD*) 4,230 food 5,428 non-food images. The food images, in the first two datasets, were selected from already existing and publicly available food image datasets.

Paper [5] describes a dataset of 251 prepared food categories split into 158,00 images gathered from the internet. For the training set, it uses 118,000 images

and, for validation, it provides human verified labels for 40,000 images.

The website referred by [6] contains photos of 100 categories of some of the most popular foods in Japan. "Each food photo has a bounding box indicating the location of the food item in the photo."[1]

Paper [7] introduces 4,545 still images, 606 stereo pairs, 303 360-degrees videos, and 27 videos of eating events as a part of a visual dataset. 101 different foods (including burgers, pizza, burgers, salads, etc.) from 11 popular fast food restaurants were gathered for use in this dataset.

[8] uses a 50-category food images set (aptly named *Food50*), built by gathering 100 relevant food images from the internet by hand for each category.

[9] features a dataset (*Food85*) of 85 kinds of foods, with a total of 100 images per category. Paper [10] details a dataset of food images taken in 10 restaurants across 5 different types of food (American, Indian, Italian, Mexican and Thai).

[11] is a large collection of Chinese food images. It contains 145,065 images for training, 20,253 images for validation and 20,310 images for testing.

Dataset	**Images**	**Classes**	**Background Noise**
Food-101 [3]	101,000	101	Yes
Food-5k [4]	5000	2	Yes
Food-11 [4]	16,643	11	Yes
FoodX-251 [5]	158,000	251	Yes
UEC FOOD 100 [6]	9,060	100	Yes
PFID [7]	1,098	61	Yes
Food50 [8]	5,000	50	Yes
Food85 [9]	8,500	85	Yes
CHO-Diabetes [10]	5000	6	Yes
ChineseFoodNet [11]	185,628	208	Yes
Food For Machine Learning (FFML)	**1282**	**423**	**No**

Table 1: Datasets for food recognition. The *Food For Machine Learning* datasets includes more classes compared to most of the prior work and and features background noise-free images.

3 The proposed Food For Machine Learning dataset

A good dataset is very important to the construction of an image recognition algorithm. It must contain a large number of high quality images to obtain an excellent performance for the algorithm.

As of now (04.24.2020) our dataset contains a number of 423 types of foods with a total of 1281 images.

[1] `http://foodcam.mobi/dataset100.html` [Online, accessed on 05.02.2020]

From the beginning we discussed the methods in which we would approach the creation of the dataset, mainly the ways in which we could obtain the images for a potential training algorithm. The options were to either search and download them from the internet or to take the pictures by ourselves. After all of this we reached the conclusion that in order to have full control over the angles and brightness, which are very important also for a high quality image, we would need to take the pictures whenever we cook or go in other places with food.

In photographing the dishes we captured them from different angles like:

- Above view
- Sides view
- Section view

For the above view we could have a perfect background preferably white or green but anyway not floral or drawings to capture the shapes of our food. Also we had to take into account the borders of our food, meaning that they have to be well defined and that they don't cut into the subject of the image. Thus, we will have a perfect above view of our dish.

Because one picture is not sufficient for the training algorithm, we also had to take side-view pictures. Depending on the type of the food, we like to choose the perfect side of view in order to capture the shape and all the ingredients included in the dishes.

The section view is important to capture the interior of the foods because the exterior can vary in its color, texture, shape or decorations. Even if these variations are small they count for the recipe of the dish.

After we defined all the rules for taking excellent pictures of a dish, we moved on to processing the images. But before we could begin this process we had to choose the best picture for each angle mentioned above from the collection of images taken. So now, we have a minimum of three images for each category of food, which means that we are now ready to modify the pictures.

We chose to use *Paint.net* in order to separate very well the food from the background and also because it's a very common application. The tools provided by this program are sufficient to make this process possible and easy:

- *Magic Wand*: we use this to select and remove easily most of the background. (See Figures 1 and 2)
- *Ellipse Select*: can help when we have a rounded shape of food like in the case of soups when they are served in bowls.
- *Eraser*: this tool is used to remove the pixels of the background left out by the previous tools and polish the borders. (See Figures 3 and 4)

After extracting the main subject of the images from the background, we had to trim the margins, by bringing them as close to the food as possible, applying a square aspect ratio and keeping the dish centered. The next step is to resize the

images to 512x512 pixels, for uniformity. These steps are very crucial so that the performance of the algorithm increases. We found that the best application in order to achieve this would be *Adobe Photoshop*, mainly because of its *Scripting* feature, which allows us to automate these repetitive tasks. The tools, settings and the order in which to apply them are written in a code form in a JavaScript file and placed into the *Presets/Scripts* folder in the application directory. The script is then binded to the *Open Document* Photoshop event, in the *Script Events Manager*, so that when opening the images in a food type, the steps mentioned above apply automatically, without any user input. The code of the script is given in Listing 1, and the results of applying it can be observed in Figure 5.

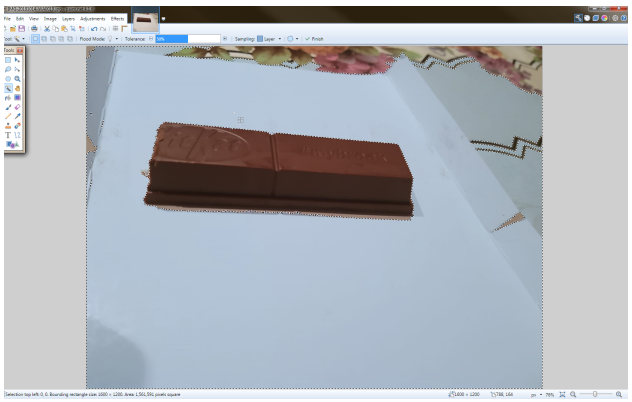

Figure 1: Roughly applying the *Magic Wand* tool.

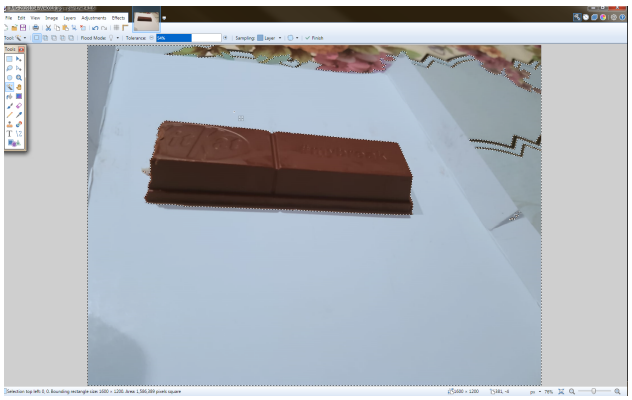

Figure 2: Fine tuning the range of selection for the *Magic Wand* tool.

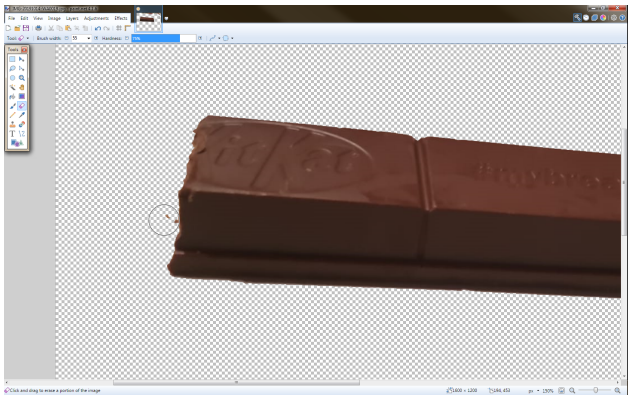

Figure 3: Using the *Eraser* tool to remove left out pixels.

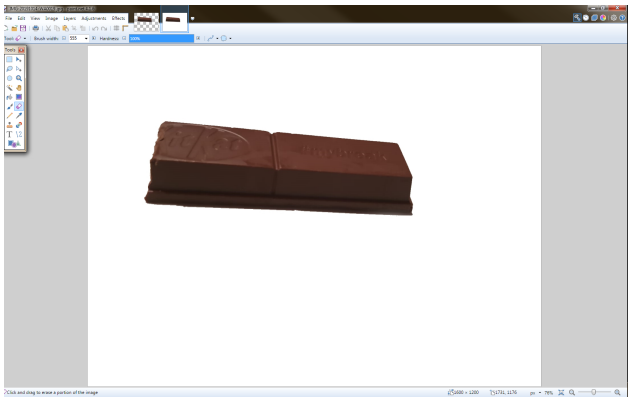

Figure 4: Final result of the process of separating the dish from the background.

```
// Store the active document in a variable
doc = app.activeDocument;
// Trim the edges of the document
doc.trim();
// Apply a square ratio
if( doc.height > doc.width ) {
  // If the height is greater than the width
  // then set the width equal to the height
  var size = UnitValue(doc.height, 'px');
  doc.resizeCanvas(size, size);
} else {
  // If the width is greater than the height
  // then set the height equal to the width
  var size = UnitValue(doc.width, 'px');
  doc.resizeCanvas(size, size);
}
// Resize the document to a 512x512 pixels
```

```
18      doc.resizeImage(UnitValue(512, 'px'), UnitValue(512, 'px'));
19      // Save the document
20      doc.save();
21      // Close the document
22      doc.close(SaveOptions.DONOTSAVECHANGES);
```

Listing 1: The code for the Photoshop script.

(a) Before. **(b)** After.

Figure 5: The results of applying the Photoshop script.

The final images are put into their respective folders named after the type of food that they represent (see Figure 6). These types of foods are then grouped by the category of which they belong to (i.e. main courses, soups, deserts, etc.). We chose to adopt a two-digit increment number convention for the images, which starts with *01.jpg*, continues with *02.jpg*, *03.jpg* and so on, and then resets with each new folder.

A public GitHub repository (see reference [1]) has been created to store the dataset and also to make it accessible for anyone wishing to use it in their own projects. The repository's structure is made up of:

- A *FOOD* folder, which contains all the categorized images.
- A *README.md* file that has a full list of food types, and also some basic information related to the dataset properties, description and history of updates.
- A *LICENSE.txt* file that details the preservation of copyright and license notices.

Whenever a new set of types of foods is added to the GitHub repository, we refresh the *History* section of the *README.md* file by entering the date and also a full list of the dishes included in the update.

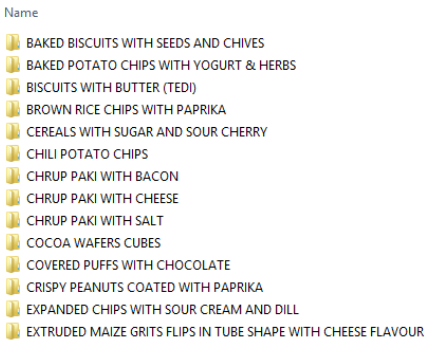

Figure 6: Folder structure.

Figure 7: File structure.

4 Numerical experiments

We have written a Python program that uses the version 2 of the TensorFlow library from Google [12].

We started the code by researching neural networks, Tensorflow, Keras and other tools that could help us to create an image training program. We found that accuracy is very important for training and learning. Accuracy must be 100% to give good results and to have a better artificial neural network.This means that we must have several layers, neurons on each layer. In our code the network is built in the "network" function.

For everyone's understanding, an artificial neural network means a computing machine that receives data at the inputs and returns data at the output. In the case of image classification, the inputs are the pixels of the image, and the output is a number that tells us what dish is in the image. For example: 0 = first meal, 1 = second. This network has several nodes connected to each other, and usually the nodes are arranged in layers. The layers are successively connected to each other. The nodes on layer x are connected to the nodes on the x + 1 layer, each with each other, but the nodes on the x layer are not connected to each other. However, any architecture (connections between nodes) is allowed. Usually we work with the so-called "feed-forward" and each connection between 2 nodes has a weight. Network intelligence is reflected in these weights. The entry in a node depends on the outputs of the nodes that enter it and weights.So, this entry is represented by a weighted sum.For example: x1 *

w1 + ... + x_n * w_n. There is a weighted sum which can increase indefinitely, so a limitation is placed and there are several threshold functions, and the input becomes: limitfunction (x1 * w1 + ... + x_n * w_n), where x_i are the outputs of the nodes related to the current node. The importance of weights is major because they influence the result. The weights represented by w_i are established by training. Initially the weights are random. If the network input is an image, it will return a value as a result of the output. If the value is correct for all the given images, then the network is perfectly trained. However, if there are images for which the classification is not correct, then the network has an error that must be corrected. This correction means changing the weights so that the error decreases. The algorithm that does this is called "back-propagation". How do we know if the value is correct or not? If the input is an image and the network outputs a number that can be [between 0 and the number of dishes we have -1]. This number corresponds to the picture sent to the input, so the number is an index for the dish. For example, if an image with soup is given at the input and the network outputs the number of an image with a steak, then it means that the network is not working properly and needs to be improved.

In our program we implemented the output as a binary vector and not a number in order to be easier to identify. This binary vector has a single position of 1 and the remainder 0. For example at the output: the vector 1,0,0 corresponding to the first image instead of the number 0, and for the third image the vector will be 0,0,1. So for the inputs we have the color of each pixel, and for the output vectors of 0-1. Only a 1 appears at the output corresponding to the image that is given at the input, otherwise 0. The number of dishes provided for training represents the number of outputs.

After the training period, the network can no longer have access to the training data because it is assumed that it has learned absolutely all the types of food represented in the training images. This memory remains stored in the network weights.

The neural network has an architecture with 8 layers as shown below:

- Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 1)))
- MaxPooling2D((2, 2))
- Conv2D(64, (3, 3), activation='relu')
- MaxPooling2D((2, 2))
- Conv2D(64, (3, 3), activation='relu')
- Flatten()
- Dense(64, activation='relu')
- Dense(430)

We have divided the dataset into 2 parts: a training set consisting of 848 images and a test set consisting of 433 images. The size of the images was

reduced to 32x32 gray-scale pixels, in order to improve the speed of the training. The dishes are distributed in folders of at least 3 images so the division is as follows: the first 2 images in each folder are for training, and the rest for the test.

We have trained our neural network on the test set and we have obtained 100% accuracy after just 3 iterations (epoch).

However, when we have fed the test set to our network, we have obtained a 15.94% accuracy. This means that about 15% of test images are incorrectly classified.

Training is the most important part of the algorithm and for this we improved the network structure and tried all kinds of layers, different training data and different values until we got the best accuracy both in training and in the test part.

5 Future work

In the future we plan to improve the dataset by adding more images of new classes of food.

Also we will improve the TF algorithm and we will test other machine learning algorithms on the FFML dataset.

References

[1] Fulop Bettina Elena, Cristea Ioan. *Food database on GitHub.* `https://github.com/cristeaioan/ffml_dataset` [Online, accessed on 23.10.2019].

[2] Fulop Bettina Elena, Cristea Ioan. *Food database on Kaggle.* `https://www.kaggle.com/cristeaioan/ffml-dataset` [Online, accessed on 23.10.2019].

[3] Lukas Bossard, Matthieu Guillaumin and Luc Van GoolFood, Food-101 - Mining Discriminative Components with andom forests, European Conference on Computer Vision, pp. 446-461, 2014.

[4] Ashutosh Singla, Lin Yuan, Touradj Ebrahimi, Food/Non-food Image Classification and Food Categorization using Pre-Trained GoogLeNet Model, International Workshop on Multimedia Assisted Dietary Management (MADiMa '16). Association for Computing Machinery, pp. 3–11, 2016.

[5] Parneet Kaur, Karan Sikka, Weijun Wang , Serge Belongie and Ajay Divakaran, FoodX-251: A Dataset for Fine-grained Food Classification, arXiv preprint arXiv:1907.06167, 2019.

[6] *UEC FOOD 100, website.* `http://foodcam.mobi/dataset100.html` [Online, accessed on 05.02.2020].

[7] Mei Chen, Kapil Dhingra, Wen Wu, Lei Yang, Rahul Sukthankar, Jie Yang, PFID: Pittsburgh Fast-Food Image Dataset, 16th IEEE International Conference on Image Processing (ICIP), pp. 289-292, 2009

[8] Taichi Joutou and Keiji Yanai, A food image recognition system with multiple kernel learning, 16th IEEE International Conference on Image Processing (ICIP), pp. 285-288, 2009

[9] Hajime Hoashi, Taichi Joutou, and Keiji Yanai, Image recognition of 85 food categories by feature fusion, IEEE International Symposium on Multimedia, Taichung, pp. 296-301, 2010

[10] Vinay Bettadapura, Edison Thomaz, Aman Parnami, Gregory D Abowd, and Irfan Essa, Leveraging context to support automated food recognition in restaurants, IEEE Winter Conference on Applications of Computer Vision, pp. 580-587, 2015

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

[12] TensorFlow library, `https://www.tensorflow.org/`, last accessed on 23.04.2020