Carrefour Hackathon

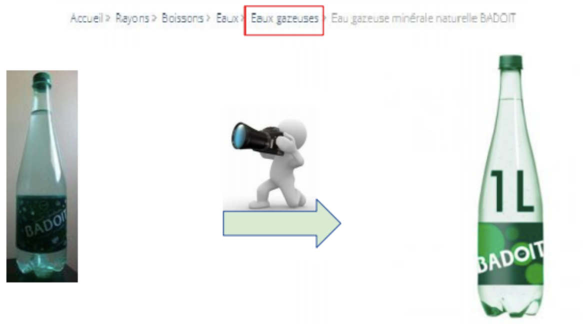
Vanessa Chahwan, Rania Ferchichi, Ashraf Ghiye,
Mohamed Ali Jebali, Karim Siala

March 21, 2021

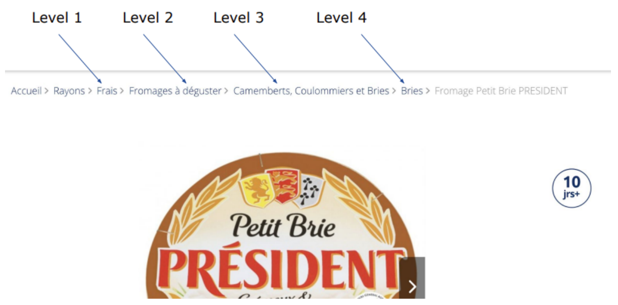
1 Introduction

The objective of this challenge is to be able to identify a product category from photos taken by the user.

Products can be categorized into different categories and through many levels. For example, a bottle of Badoit can be first classified as *Drinks* or (or *Boissons*), but it has also more nested levels such as Water (i.e. level 2: *Eaux*) and more specifically sparkling water (i.e. level 3: *Eau gazeuses*) and finally a fourth level that is very specific for this mark of water. Below is an image that explains this example.


Accueil > Rayons > Boissons > Eaux > Eaux gazeuses > Eau gazeuse minérale naturelle BADOIT
BADOIT
1L
BADOIT

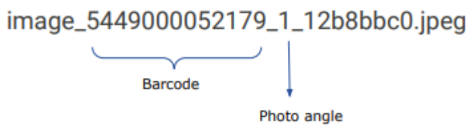
However, in the context of our challenge, we are only interested in predicting level 3. Below a second example.

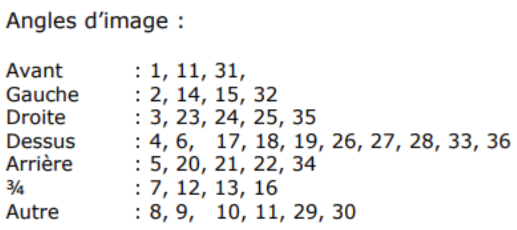

Level 1
Level 2
Level 3
Level 4
Accueil > Rayons > Frais > Fromages à déguster > Camemberts, Coulommiers et Bries > Bries > Fromage Petit Brie PRESIDENT
Petit Brie
PRÉSIDENT
10 jrs+

2 Data Exploration

Before going into our data visualization and statistics about the dataset, we were given at first a 180GB of product images stored on Google Cloud Storage. Also, the associated metadata (e.g. Barcode, product description, etc..) for all these images were given in a separate BigQuery table.

Filenames were decoded as follow:



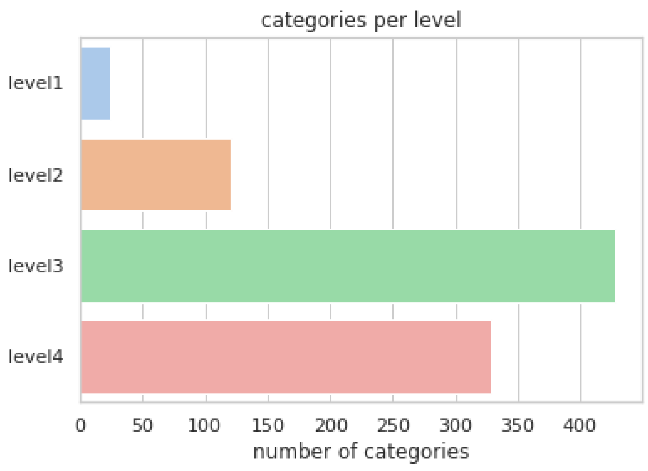

Angles d'image :

Avant	: 1, 11, 31,
Gauche	: 2, 14, 15, 32
Droite	: 3, 23, 24, 25, 35
Dessus	: 4, 6, 17, 18, 19, 26, 27, 28, 33, 36
Arrière	: 5, 20, 21, 22, 34
¾	: 7, 12, 13, 16
Autre	: 8, 9, 10, 11, 29, 30

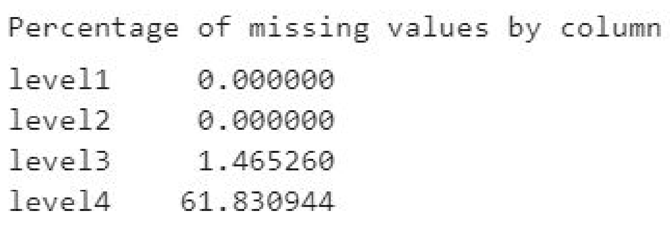
Note that for the same product, we can have multiple images in different angle of views, but also multiple images with same angle of view for one product exist. Also, a product can have multiple categories, but those latter are the minority thus we chose to only keep the main category for each product.

2.1 Levels and missing data

In the available metadata, each product is assigned to a category in one of the 4 different levels. Level 1 is the most general level where we have a small number of different categories whereas the level 3 contains 428 different categories and approximately 330 categories are in level 4. You can see below the number of classes for each level.



It is a bit strange to notice that level 4, which should be more specific than its higher level, contains less categories. This is explained by the high ratio of missing data for this particular level. As you can understand from the next figure, more than half of the labels of level 4 are missing.



```
Percentage of missing values by column
level1     0.000000
level2     0.000000
level3     1.465260
level4    61.830944
```

This is indeed a high value and it was obvious for us that it would be not be really efficient to take level 4 into consideration. So we choose to predict the **level 3** as our objective since it is still a very challenging task as we have 428 different class.

And since less than 2% of the data has unlabeled products at level 3, we've just decided to drop the rows with null values and only train our future model with the primary link for each product.

2.2 Labels distribution

Once we've set our labels, it is an imperative task to investigate on the distribution of our labels to take some basic conclusions. Here, we plot the top 30 cited categories at level 3. As seen here, there are a high number of products classified as "Maquillages", "Confiseries chocolatées" & "Vins rouges" with approximately 1500 products belonging to one of these categories.

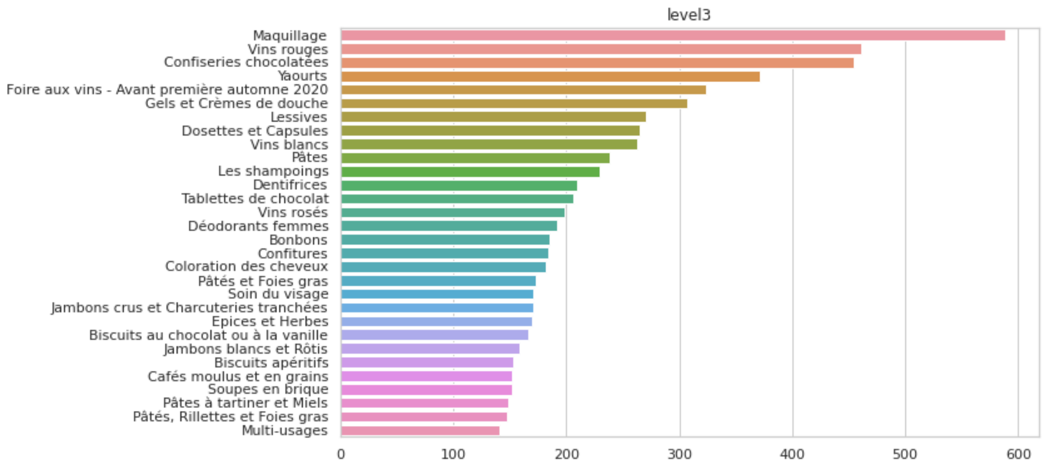
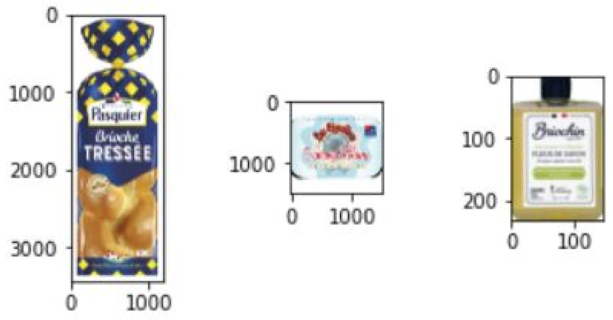


Figure 1: Labels' distribution for top 30 listed products

Other products have only very few examples in the dataset. For example, there are only 3 images of products belonging to "Agneau" and only 12 images of "Aide Culinaire". So it is clear that we are working with very **unbalanced** dataset and we should take this into account further steps of preprocessing the images in order to get more examples of less redundant classes and this is will be more explained in details in the "Data Augmentation" part which is useful to create artificial images from one image by some simple geometrical transformations or even other techniques.

2.3 Image characteristics

Another important point is that images are varying in size, contrast and other characteristics.
The first obvious problem is that in our dataset, there are images with 1024*1024 pixels while other have only 128*256 pixels. And as we will have to deal with Convolutional Neural Networks for prediction, we have to choose a standardizes input shape for all images and so we had to try various image resizing techniques with different techniques for padding or/and re-sampling.



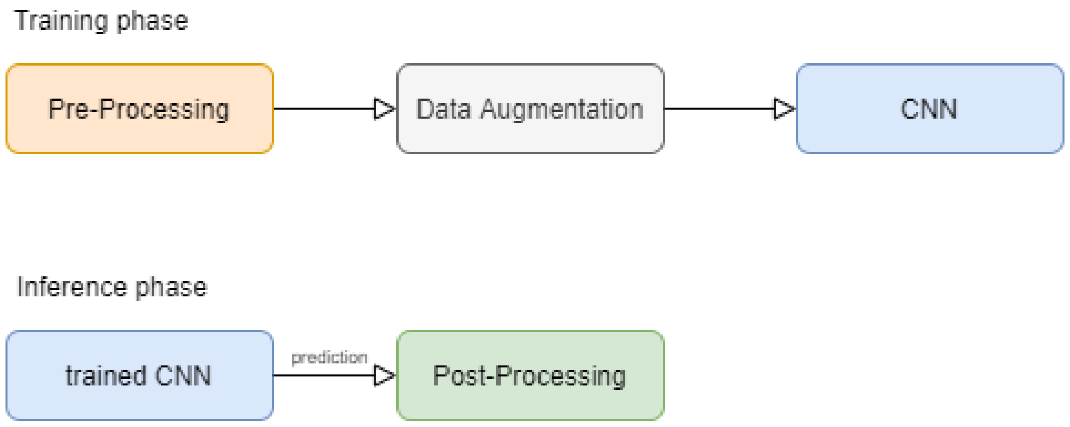
We have also noticed that most of the images have no external background and only have a white background. It was unclear for us if we had to include in our preprocessing task a technique to extract the background from the images of the products. But since we've only had the test data in the final days, we could not take it into account and this is maybe something we should have considered and treated it better.

In conclusion, we have a vary unbalanced dataset with very different images in terms of size, quality, resolution, background or lightening. All these drawbacks make it challenging to make a proper architecture before feeding our images to the fina models.

3 Approach

To proceed, we chose Deep Learning and Neural Networks to do the classification task since we have many examples and a large dataset.

We tried out many CNN models, some are designed by hand and trained from scratch and others were famous pre-trained networks like Xception and ResNet50, and did a fine-tuning on our samples by freezing the first layers and training only the last blocks of the network.



We recognize that this is a difficult task. One, because of the distribution of the classes, and second, because we do not have enough images representing each class, for the model to learn properly. However we have many ideas to improve our approach, see Enhancement section, that we did not have enough time to implement.

- **The data:** In view of our data exploration based on the metadata table and on the visualization of the images, we decided to work only on frontal views of products. The reasons behind this are many.

  The most available views for all products are of angle ids equal to 1, 11 or 31, which correspond to front view angles. Most of the products also have a few number of views from different angles, and again, most of these are frontal.

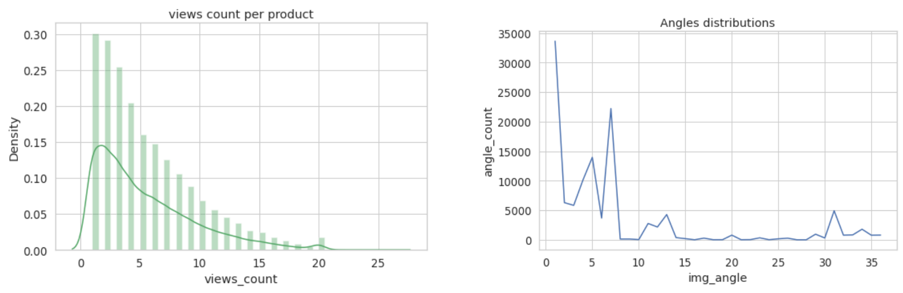


Figure 2: Distribution of the views count per product on left and image angles on right

Then, we also kept one image per product. This is to avoid **data leakage**. Because one product may have multiple views, sometimes corresponding to the same angle, some of them may fall into the validation set. Because of this, our final accuracy would be inflated and misleading.

In the end, we had a total 41315 front views images. And kept only 19106 by keeping one image per product. These will be used so we can be able to validate our performance with the validation *real* accuracy on top of the visual evaluation.
Once we were confident about our architecture, we let it train on the images.

- **Pre-Processing:**
  - **Resizing:** The original images are of very high quality, and of different sizes. To be able to train our model, we need to resize them to a fixed shape. We chose to resize them to (299, 299, 3). Which will be used as our input size.
  - **Standarization:** Before feeding the images to our model, we need to normalize the pixels' intensity. We used a pre-trained model and thus we also made use of its pre-processing pipeline to apply on our images.The inputs pixel values are scaled between -1 and 1, sample-wise.
- **Data Augmentation:** The images also contain no background. The testing images may not be so clean. And because by taking one frontal view per product we reduced the number of images, we tried to use data augmentation. This would have also help with overfitting. Suprisingly though, it only made our accuracy worse, and our model slower.
  To combat overfitting, we used weight decay regularization and dropouts.
- **External data:** To get more images to train on and to represent better the classes, we used web scrapping to get more images of different products, with our same labels. This added a total 13470 which also contained images similar to the test set.
- **CNN model:** Our final model is based on Xception. Xception is faster than Resnet and has fewer parameters to learn. Xception is trained with imagenet dataset which is quite different from the set we're working on. To learn features related to our images, we unfroze the deeper layers and retrained them with our training data. To help speed up the process, we fed the data to the model on the fly while running. The tuning of the hyperparameters such as learning rate, the number of layers to train, the batch size, the regularizations were done manually. This is due to the time limit we have. More fine-tuning could be done.

4 Results

We reached a maximum of 63% accuracy on the validation set. With over 95% accuracy on the train test. Even with the techniques we used, the model could not help but overfit the data.

We'll use the whole dataset to train the model one last time. And use the final weights to predict the unseen testing set.

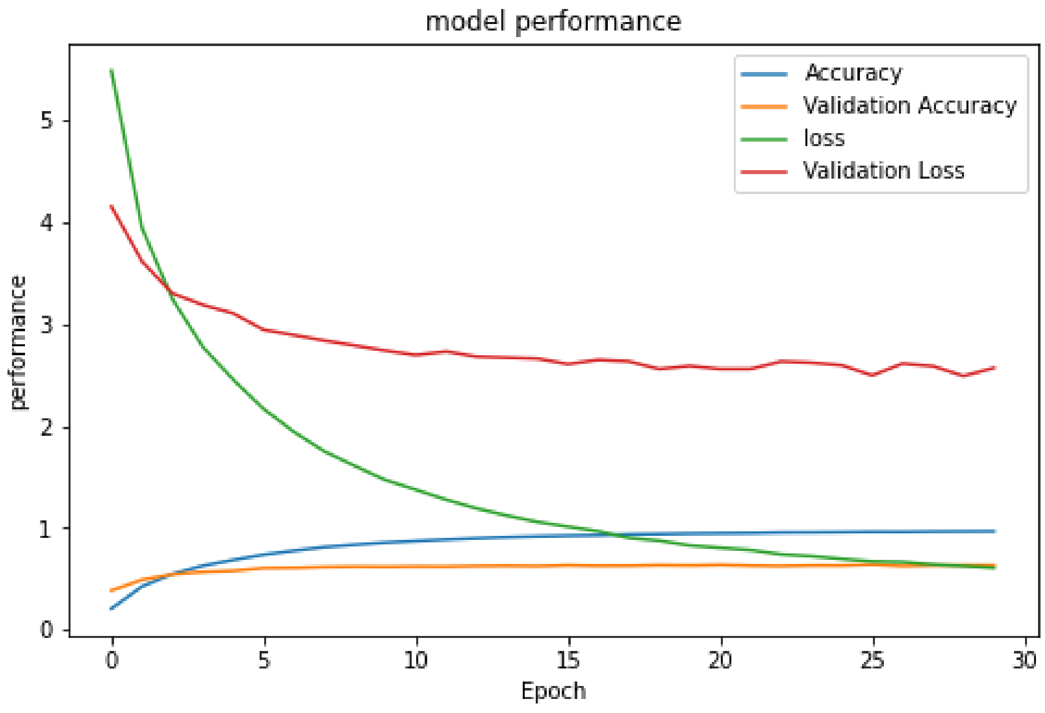


Figure 3: Model performance

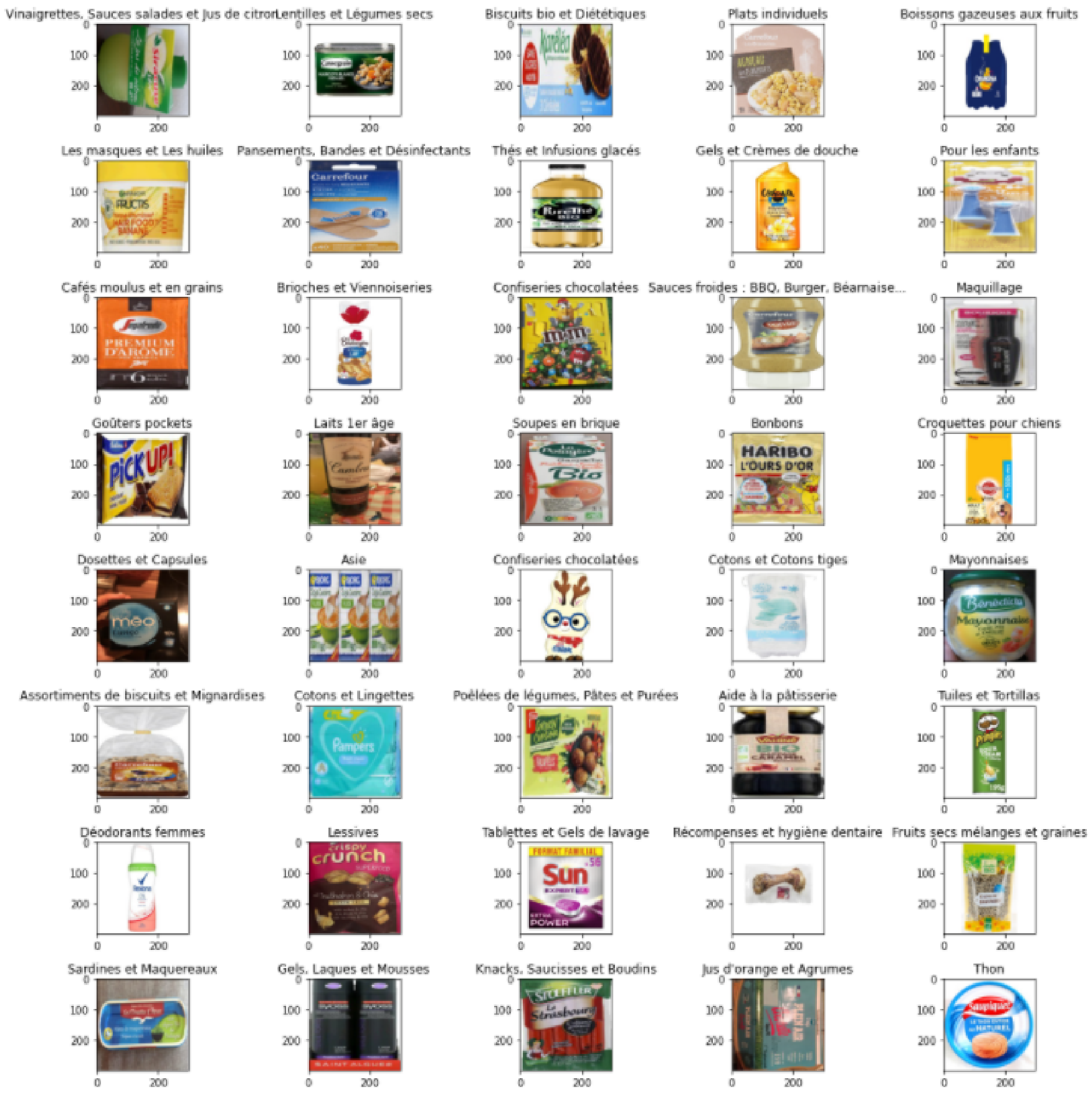

Vinaigrettes, Sauces salades et Jus de citron
Lentilles et Légumes secs
Biscuits bio et Diététiques
Plats individuels
Boissons gazeuses aux fruits
Les masques et Les huiles
Pansements, Bandes et Désinfectants
Thés et Infusions glacés
Gels et Crèmes de douche
Pour les enfants
Cafés moulus et en grains
Brioches et Viennoiseries
Confiseries chocolatées
Sauces froides : BBQ, Burger, Béarnaise...
Maquillage
Goûters pockets
Laits 1er âge
Soupes en brique
Bonbons
Croquettes pour chiens
Dosettes et Capsules
Asie
Confiseries chocolatées
Cotons et Cotons tiges
Mayonnaises
Assortiments de biscuits et Mignardises
Cotons et Lingettes
Poêlées de légumes, Pâtes et Purées
Aide à la pâtisserie
Tuiles et Tortillas
Déodorants femmes
Lessives
Tablettes et Gels de lavage
Récompenses et hygiène dentaire
Fruits secs mélanges et graines
Sardines et Maquereaux
Gels, Laques et Mousses
Knacks, Saucisses et Boudins
Jus d'orange et Agrumes
Thon

Figure 4: Results on the validation set

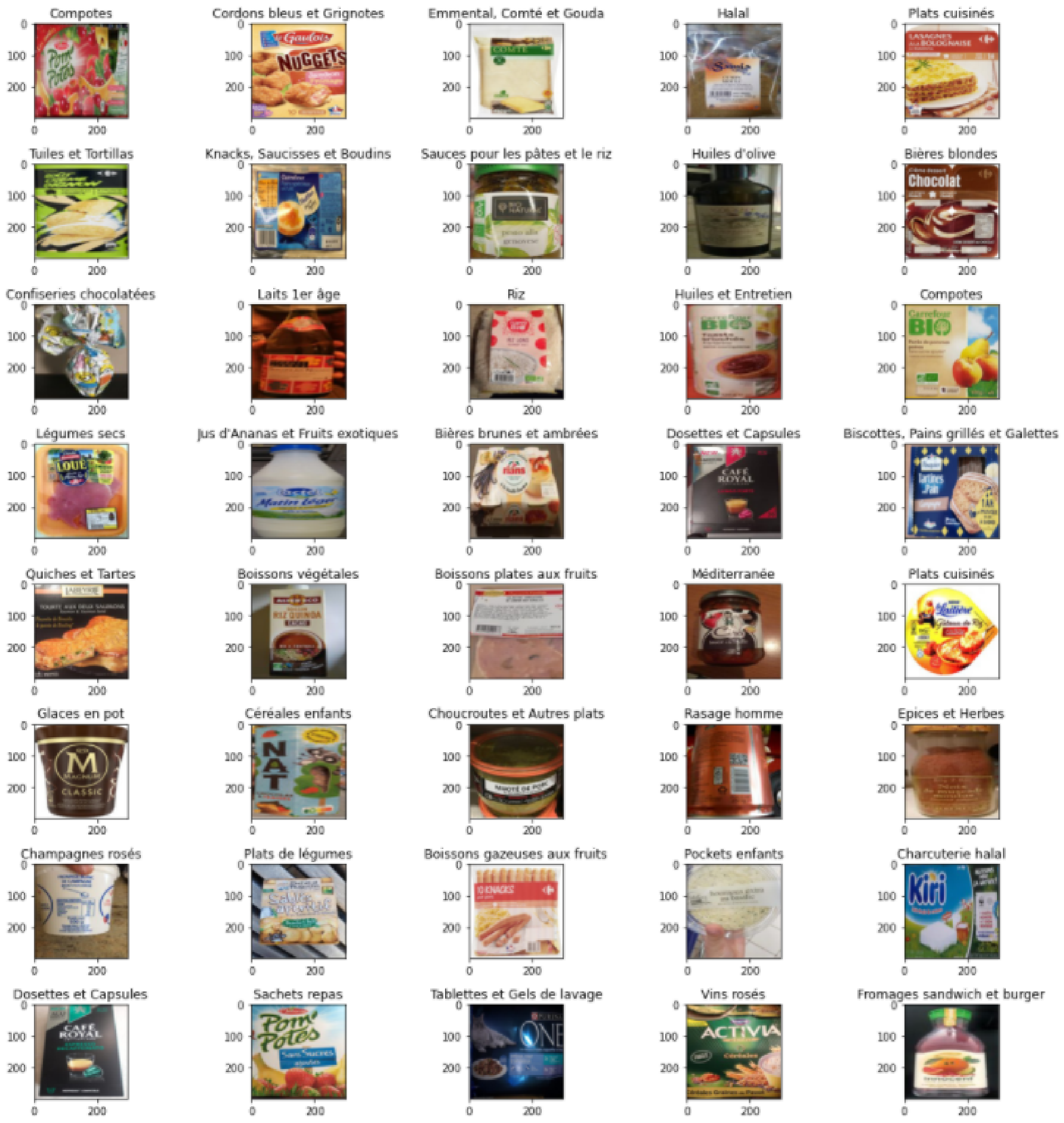


Figure 5: Results on the test set

5 Enhancements and Conclusion

We will propose some improvements which need more time to develop and apply. Because of the limited time we got to solve this challenge, we weren't able to put them into direct use, though we experimented with some of them.

5.1 OCR

Since our query is to classify products, and knowing that most of the products come with metadata that is informative about their category, we can extract them directly from the images. Some metadata examples are Brand names, Barcode and other text information that indicates the product's nature and identity.

Exploiting this metadata, which can be extracted using unsupervised methods (e.g. OCR), serves to redefine our problem in another way. However, it is hard to integrate this type of data (text) alongside the products images in the learning process.

One approach can be to use a multi-model, where each product is represented as a set of images and a set of extracted texts using OCR.

The other approach, which we focused on, was to build a hash-map where keys represented the different categories in level 3 and for each key associated an array of possible keywords extracted from the products images belonging to this category.

Since some brand names help to identify the category directly, e.g. "LU" are for known for biscuits, "Barilla" will mostly be pasta or sauces, Coca-Cola is unquestionably a "Soft Drink" and so on.

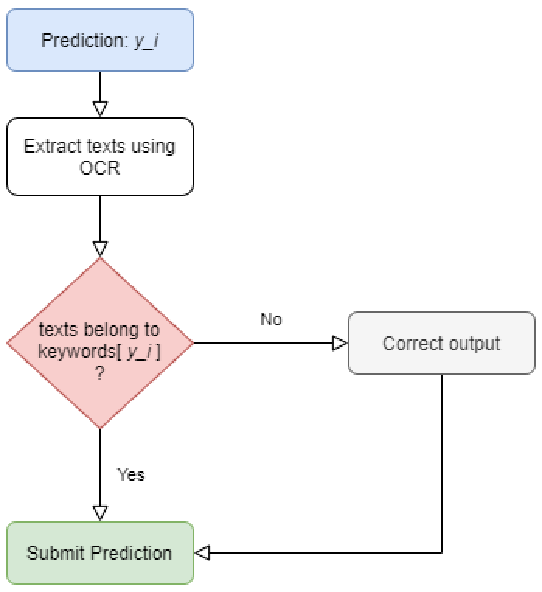


Figure 6: Post-Processing and output correction using OCR diagram

In that way, we are able to correct the output of the model by looping over the predictions and see if a recognized text belongs to the buckets of keywords for that predicted class. If not we are able to correct it manually in a semi-supervised fashion.

5.2 Pre-Processing

More pre-processing is needed before the inference step. The testing images are taken using the user's cameras and are a low-quality representation of what a product looks like. Thus we need more powerful techniques to clean the images before inference:

- **Image segmentation**: to eliminate the background and keep only the product's image.
- **Histogram transformations**: to enhance the lighting conditions and correct the colors.
- **Geometric corrections**: to resolve the distortions of some products.

5.3 Data Augmentation

Instead of using the same dataset to do data augmentation using rotations, zooms, or any other affine transformations, we figured that using the Barcode of the products at our disposal, we can crawl some sites like open food facts, which provides photos of nearly all the products taken by users and which are more realistic.

Hence, we will be able to train our model using user-taken images. These later are closer in nature to testing data. Training a model using both data (Crawled and from Carrefour) will improve the generalization capacity of the former.

To this end, we have presented our work for the Carrefour Hackathon 2021. The aim goal of this challenge was to identify the product's level 3 category. This task can be considered as a traditional multi-class classification. We solved the mentioned problem using deep learning and convolutional neural networks.