Food Detection in Images

Daniel Frederico MASSON FURLAN

Résumé

This is the report of a study that aims to detect the presence of food in images. In order to assess the applicability of different solutions, we opted for an iterative approach, testing different machine learning algorithms, including deep neural networks. In addition, the Grad-CAM algorithm is used to visualize the most important features of the image when CNN models are classifying. 4 models were built and used for this purpose: a simple DNN, 2 pre-trained models (VGG16 with fully connected layers on top, VGG16 with SVM on top) and a new CNN architecture. For training and testing accuracy we obtained: DNN: 92%, 69%; VGG + fully: 98%, 90%; VGG + SVM: 98%, 89%; New CNN: 100%, 84% respectively.

Additionally, a bootstrap sampling approach using a Decision Tree Classifier (DTC) was used and new CNN architectures were also built and evaluated. Nevertheless, due to time constraints, the new CNN architectures are hard to compare with the 4 previous models. They were both (*DTC* and new *CNN*), therefore, put in a separated section (*Additional Models*).

keywords : Machine learning; food detection; CNN; deep neural networks; GRAD-CAM

Summary with hyper-link

Summary

1. Introduction

Image classification tasks have long been studied and it is an important field of machine learning and artificial intelligence. Normally associated with complex models, detecting and correctly classifying a figure/image is not trivial for the computer as it is for us, humans. For that reason, several models exist nowadays and deep neural networks gained their space with the advancements of microprocessors, overcoming time- and memory-constraints.

Despite the success of using convolutional neural networks for deciphering and understanding the subtle meanings of an image, simple and light classification models (e.g. SVM, Linear Regression) have always caught the attention for their interpretability and are still explored to modelize high-dimensional problems (such image classification).

But the problem scope goes beyond "how complex a model is". For instance, Joutout et al. [1] proposed a SVM classifier for such purpose and, despite the "simplicity" of the model, data acquired to predict spherical fruits involves a laser-scanning to obtain reflectance and range precision, outcoming the fruit shape and color.

Particularly, recognizing the presence of food items on images is also a challenge and Jimenez et al. proposed one of the first methods back in 1999 [2].

Although the difficult to classify food items, as they are strongly related to color and shape [3], novel methods have been tested and combination of multiple CNN models can already predict Mediterranean Diet food items with an accuracy of 52,71% [4].

2. Methodology

2.1 Models

4 approaches were used for training and evaluating:

- Build a simple DNN model (*1st)*
- Pre-trained models:
    - Build a fully connected layer on top of VGG16 *(2nd)*
    - Build a SVM classifier on top of VGG16 *(3rd)*
- Build new CNN architectures *(4th)*

2.2 Data processing

Although the image source for training and testing our models comes all from the same dataset (TRAIN and TEST folder), we prepared 3 types of processed data in order to train the different models.

- The data folder TRAIN was split into a train and validation dataset.
- The data folder TEST was entirely used for test evaluation.

Regardless of the model, due to computational process limits, the images had to be resized in 1/4 the original (from 240x320 to 60x80).

Furthermore, for training the new CNN, the images had also to be converted to grayscale. Despite this being a topic still in discussion in the scientific community, the accuracy for both methods (RGB and grayscale) don't differ too much [5][6] (Figure 1).

data_inet: for VGG16 model. input : (samples, 60, 80, 3)

data: for own architecture. input : (samples, 60, 80, 1)

data_stack : for simple DNN. input : (samples, 4800)

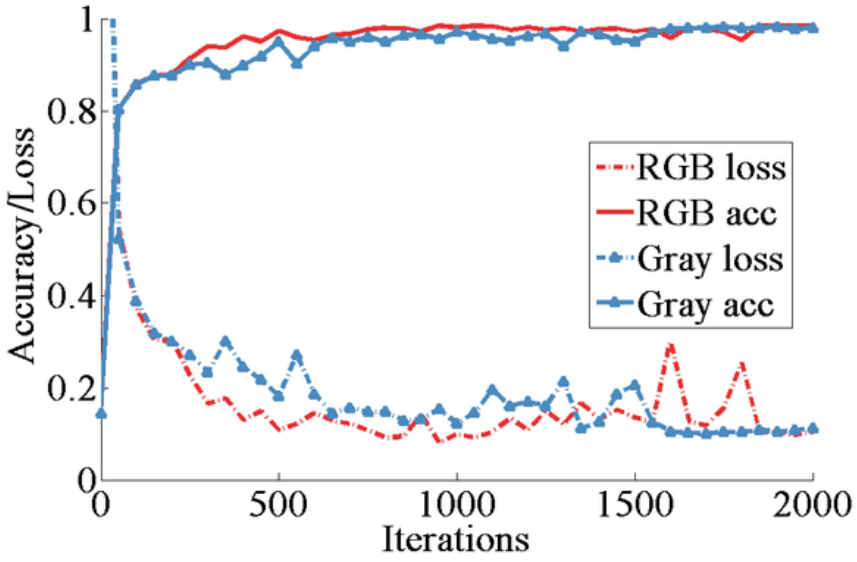

Figure 1: accuracy/loss comparison for RGB and Grayscale image

2.3 Metrics and evaluation

Accuracy metrics were used for all the models evaluation. Display of *confusion matrix* and the *GRAD-CAM* visualization for CNN architectures are also used to better understand the models' behavior.

Evaluation was done in the TEST folder, composed of 3279 images (1601 food and 1678 non food)

2.4 Grad-CAM

The Grad-CAM algorithm was used in

- Pre-trained model (VGG16)
- New CNNs models

While new CNNs models could predict directly from the image input, the pre-trained models had first to pass the image through the base convolutional layers before predicting.

Models

1st approach: simple DNN model.

If CNN are largely used to perform image classification tasks, DNN were the basis for the first models learning to recognize images. For the sake of curiosity, a simple (really simple) DNN is built here.

It is composed of no more than dense layers. Its final architecture can be found in the *Appendix* section.

Parameters

batch size	32
rlrop patience, rlrop ratio factor	50, 0.1
optimizer	Adam
epochs	500

Tuning the parameters

In the first approaches, the training accuracy increased until 0.8, falling suddenly to a steady value of 0.5 until the end of training.

Taking into account the problem of vanishing gradients and local maximum, a possible solution for such behavior could be a variant learning rate that takes into account an "unchanged accuracy value" through a certain period of training.

For that reason, a ReduceLROnPlateau algorithm was used in order to scheduled change the learning rate when reaching a plateau of learning.

This solution solved the problem with a factor change of 0.1 (*new_lr = factor*current_lr)* in the learning rate for each 50 epochs showing no accuracy improvement.

2nd approach: pre-trained VGG16 + training fully-connected layer

"Standing on the shoulder of giants"

Although all the pre-trained models for image classification should have a great accuracy for most part of the classification tasks, the use of vgg16 was prioritized as its training dataset (Imagenet) has food images[7].

For this model, we'll use the data in the 3 channels format (60, 80, 3) as vgg16 was trained with 3 channels. *50 epochs* were used for training this model.

Before training the fully connected layers, we have to extract the features from the VGG16 model simply fitting the TRAIN dataset with VGG16 (without its top predictor layer).

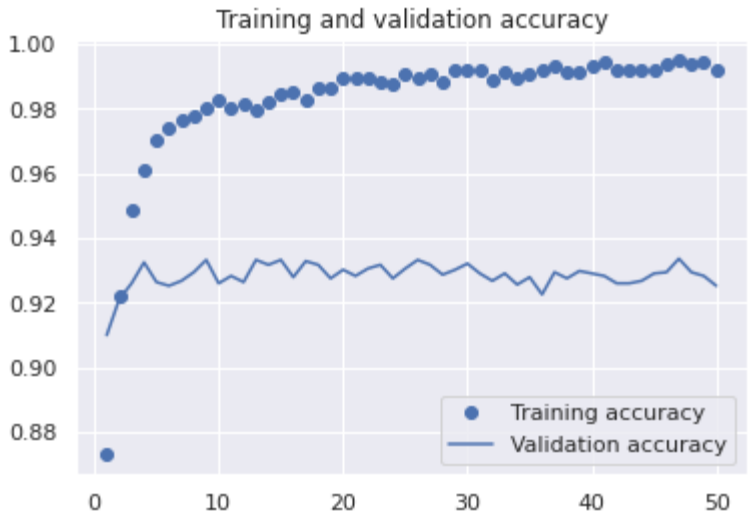

Figure 2: train and validation accuracy for VGG16 + fully connected layers

3rd approach : VGG16 + SVM on top

For this approach, we used a *Stochastic Gradient Descent Classifier* with *hinge_loss* and a *l2* regularizer. Its final architecture can be found in the *Appendix* section.

4th approach: new CNN architecture:

The new CNN architecture was built inspired by the *StridedNet* model, using the *BatchNormalization* layer for stabilizing the training. Its final architecture can be found in the *Appendix* section.

Parameters	
batch size	256
rlrop patience, rlrop ratio factor	5, 0.1
optimizer	Adam
epochs	25
data augmentation	None

Results

Confusion matrix:

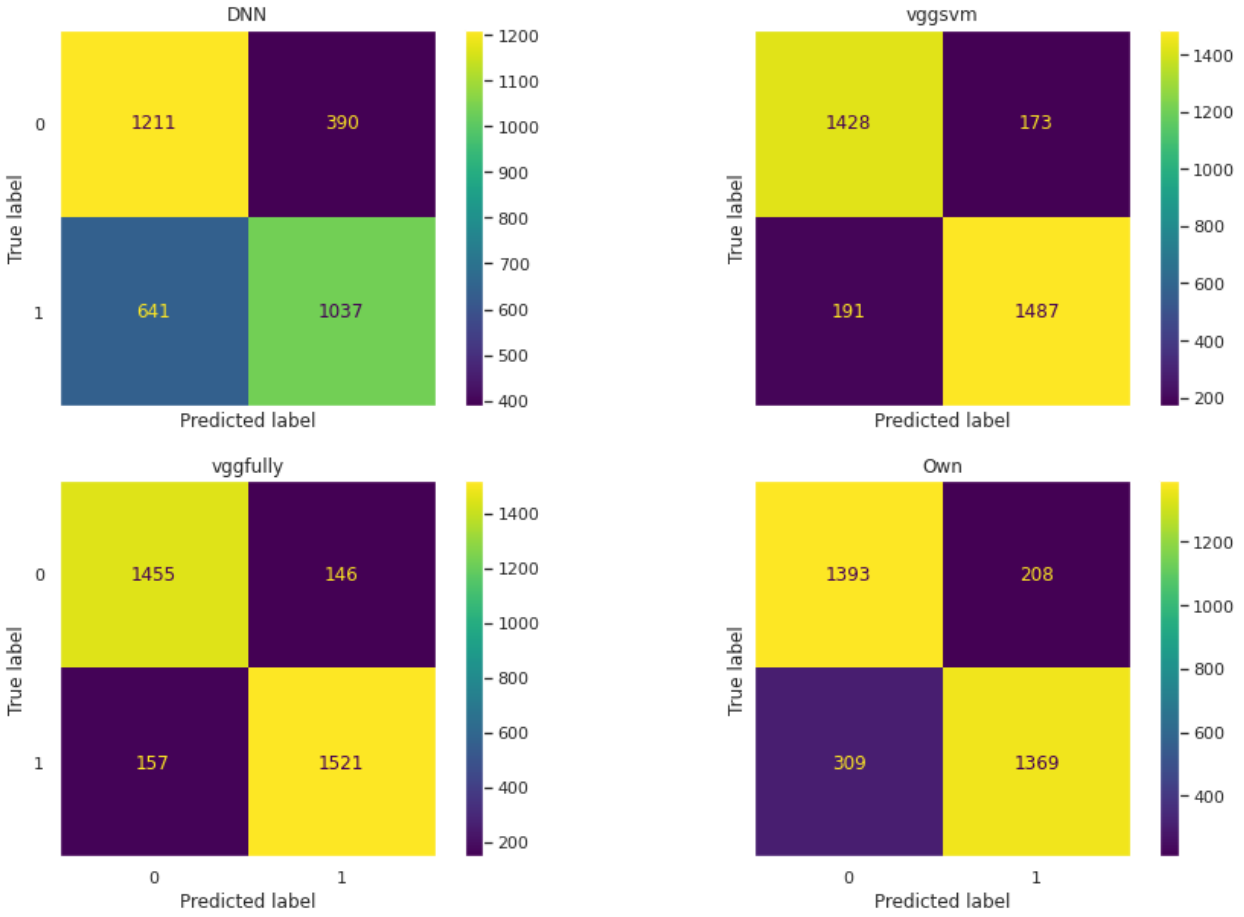

Figure 4 : comparison between all confusion matrix

We can observe in decreased order of importance the values of TP, TN, FP and FN:

Model	TP	TN
VGG + fully	1521	1455
VGG + SVM	1487	1428
CNN	1369	1393
DNN	1037	1211

Model	FP	FN
DNN	390	641
CNN	208	309
VGG + SVM	173	191
VGG + fully	146	157

Classification Report:

DNN

	precision	recall	f1-score	support
0.0	0.65	0.76	0.70	1601
1.0	0.73	0.62	0.67	1678
accuracy			0.69	3279
macro avg	0.69	0.69	0.68	3279
weighted avg	0.69	0.69	0.68	3279

VGG + fully

	precision	recall	f1-score	support
0.0	0.90	0.91	0.91	1601
1.0	0.91	0.91	0.91	1678
accuracy			0.91	3279
macro avg	0.91	0.91	0.91	3279
weighted avg	0.91	0.91	0.91	3279

VGG + SVM

	precision	recall	f1-score	support
0.0	0.88	0.89	0.89	1601
1.0	0.90	0.89	0.89	1678
accuracy			0.89	3279
macro avg	0.89	0.89	0.89	3279
weighted avg	0.89	0.89	0.89	3279

new CNN

	precision	recall	f1-score	support
0.0	0.82	0.87	0.84	1601
1.0	0.87	0.82	0.84	1678
accuracy			0.84	3279
macro avg	0.84	0.84	0.84	3279
weighted avg	0.84	0.84	0.84	3279

It's evident the greater performance of the pre-trained models compared with the DNN and new CNN architecture. Nevertheless, we point out the simplicity of the CNN training model, with only 9 layers and ***25*** epochs training.

Grad-CAM visualization for CNN models

For the VGG + fully and new CNN architecture we could visualize the gradient of the activation map of certains convolutional layers:

VGG + fully connected layers:

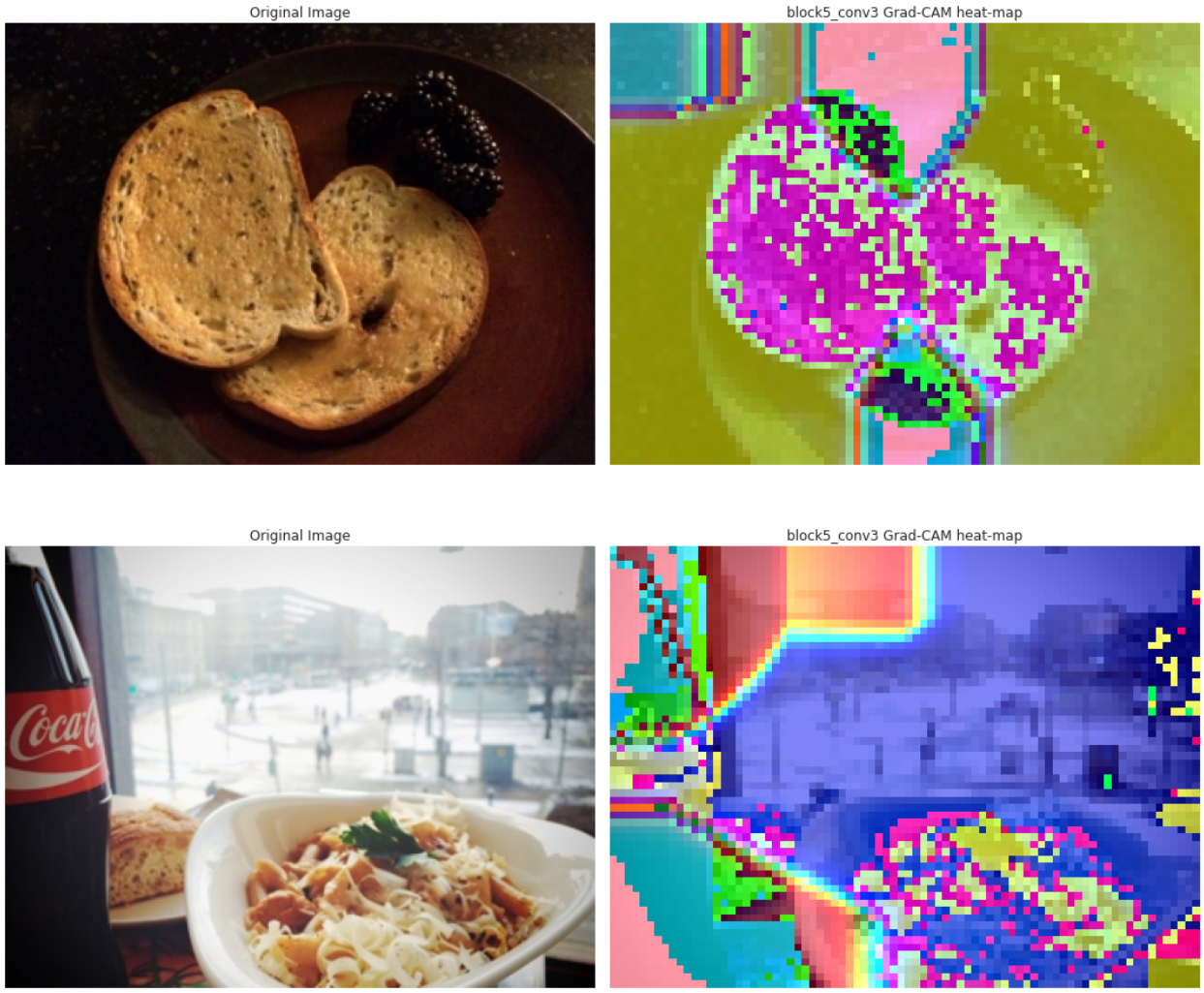

FIgure 5: grad-cam visualization for VGG's *block5_conv3* layer for 3 test images. Prediction ([value]-label): [1]-food, [1]-food and [0]-not_food, from top to bottom.

Looking at all the convolutional layers:

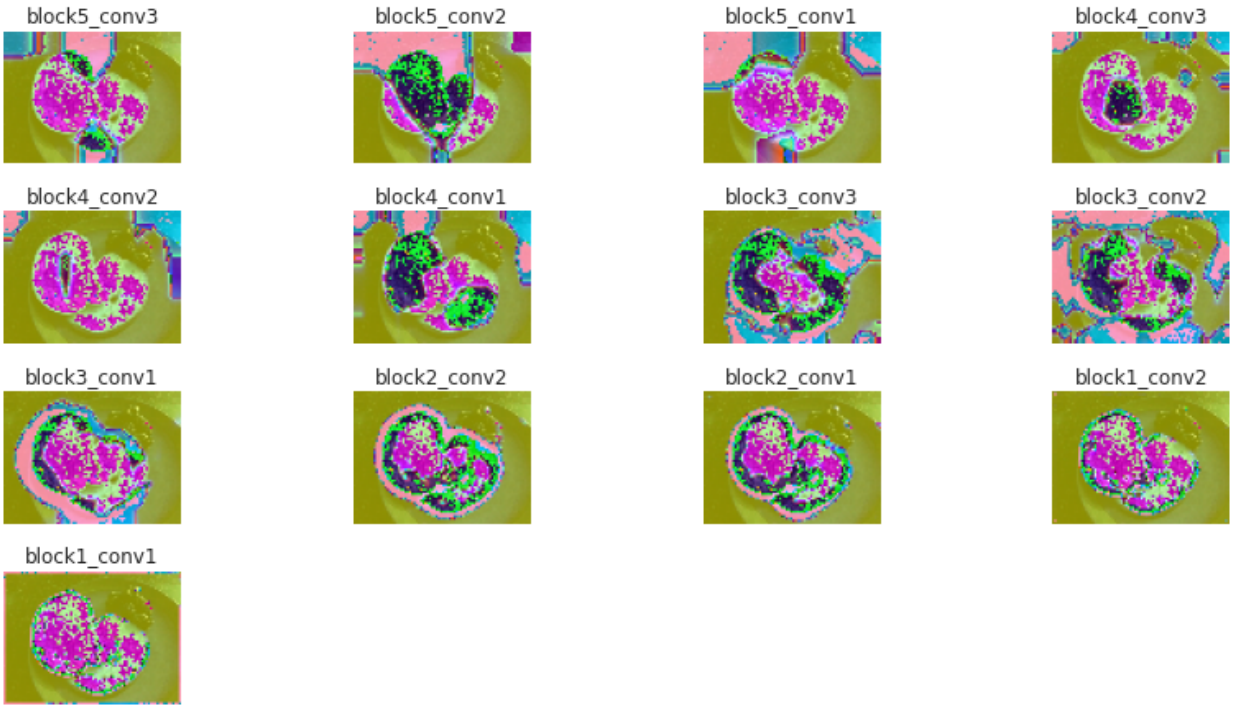

Figure 6: grad-cam for all the VGG's convolutional layers for 1 test image. Prediction ([value]-label): [1]-food

New CNN:

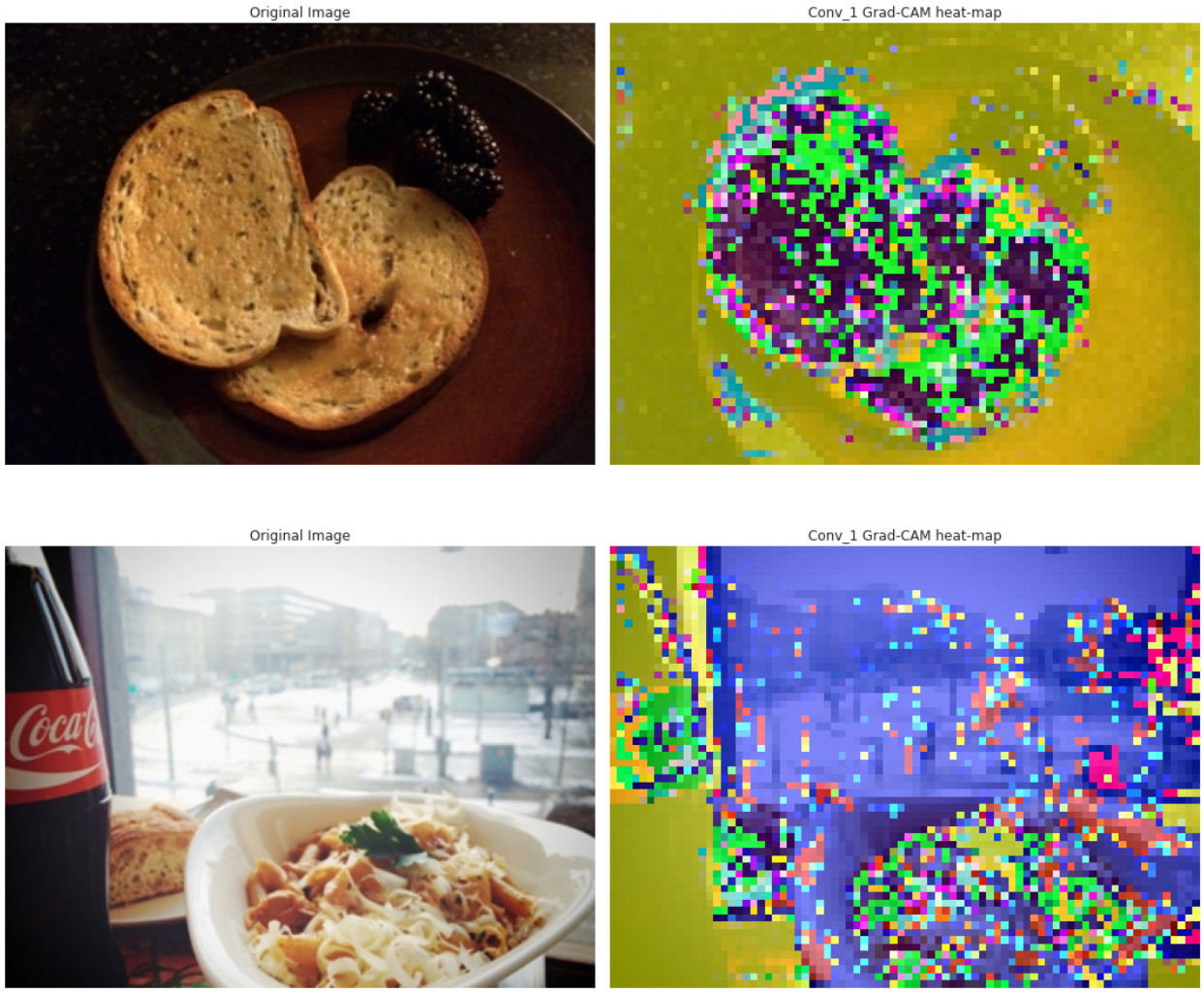

FIgure 7: grad-cam visualization for new CNN's conv2d_17 layer for 3 test images. Prediction ([value]-label): [0.3465631]-not_food, [0.9901761]-food and [0.10886151]-not_food, from top to bottom.

Looking at all the convolutional layers:

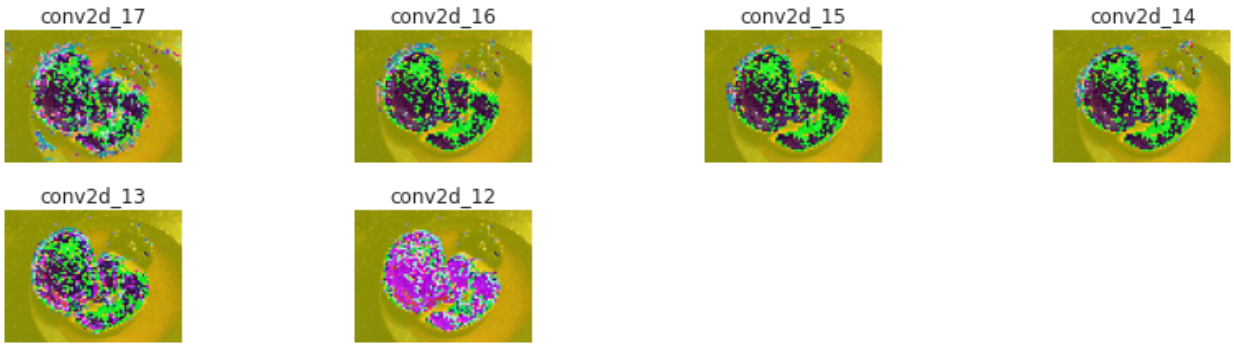

Figure 8: grad-cam for all the new CNN's convolutional layers for 1 test image. Prediction ([value]-label): [0.9901761]-food

Interestingly, we can see that the first conv layer conv2d_17 has a more defined gradient output (a more homogenous gradient of color) whereas the subsequents layers seem to learn other patterns beyond the food shape (we can see some activation importance coming from the plate as well)

Comparing the 2 CNN models:

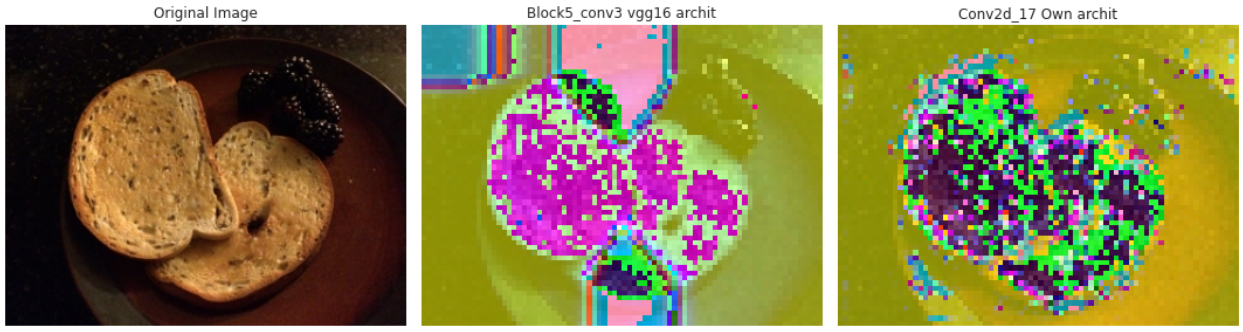

Figure 9 : grad-cam comparison for the 2 CNN models. Prediction (label): *food* for both.

We can observe that where the pretrained model VGG has a more defined area of observance in the food element (the toast), the new CNN architecture seems to give importance to the round shapes of the plate, an eventual sign of the element learned by the model.

Additional Models

Iterative process with different CNN architectures

Due to time and computational process constraints, we will build several new CNN models with only 20 epochs, changing some parameters like:

- kernel_initializer ("glorot_uniform" or "he_normal")
- kernel_regularizer (l2(0.0001 or None)
- kernel_size **
- dropouts (3 layers or 1 layer or None)
- data augmentation. (True or None)

** the kernel_size:

Borrowing the concept of human learning from baby to adult when deciphering an image and adapting for this task, I will try some different patterns through the layers of the model, establishing 3 parts of "learning":

1. baby: where we first have a broad picture of an image without any categorization of shapes and colors (bigger kernel sizes e.g. 5 or 7)
2. child: where we can define and characterize lines, contours and colors (small kernel sizes e.g. 3, or 1)

3. adult: where we can generalize without looking to specific parts but rather to the whole context and are able to infer concepts, ideas and categorizations (bigger kernel sizes e.g. 5 or 7)

Below you can find the trained models and their parameters.

Model	Dropout quantity	Dropout values	Kernel_init	kernel_reg (l2)	kernel_size	data aug	train acc	val acc	epochs
3	1	0.25	glorot	0.0005	3	True	0.85	0.87	20
4	1	0.25	he_uni	0.0001	3	True	0.85	0.84	20
5	1	0.25	glorot	None	3	True	0.86	0.83	20
6	0	-	glorot	0.001	3	True	0.86	0.79	20
7	0	-	glorot	0.0001	3	None	0.99	0.84	20
8	0	-	glorot	0.0001	7,5,3,1,3,7	None	0.99	0.86	20
9	0	-	glorot	0.0004	7,5,3,1,3,7	True	0.85	0.83	20
10	0	-	glorot	0.0004	5,5,3,1,3,5	None	0.99	0.86	20
11	0	-	glorot	0.0001	5,5,3,1,3,5	True	0.85	0.81	20

Table 1: CNN models and their parameters

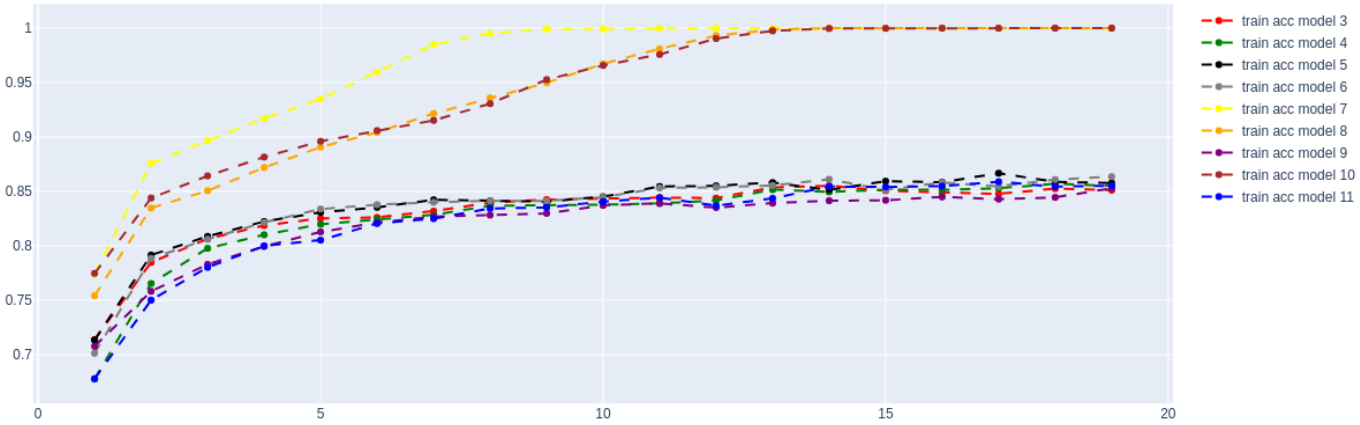

Figure 10 : training accuracy for 8 CNN models

From these first round of attempts, a final CNN version (called *own2*) will be trained and inspired by one or multiple models' parameters, adding more layers, increasing training iteration to 50 epochs and using data augmentation.

Although the choosing of the model's parameters doesn't follow any strict criterion, we will give priority to the use of data augmentation and to not overfitting.

Furthermore, as the models were simple and trained with only 20 epochs, we don't think that they would have a great performance difference between them when training with more epochs in a higher complex architecture.

Nevertheless, respecting the above criterion, models 5 and 11 are chosen and a blend of their parameters will be made.

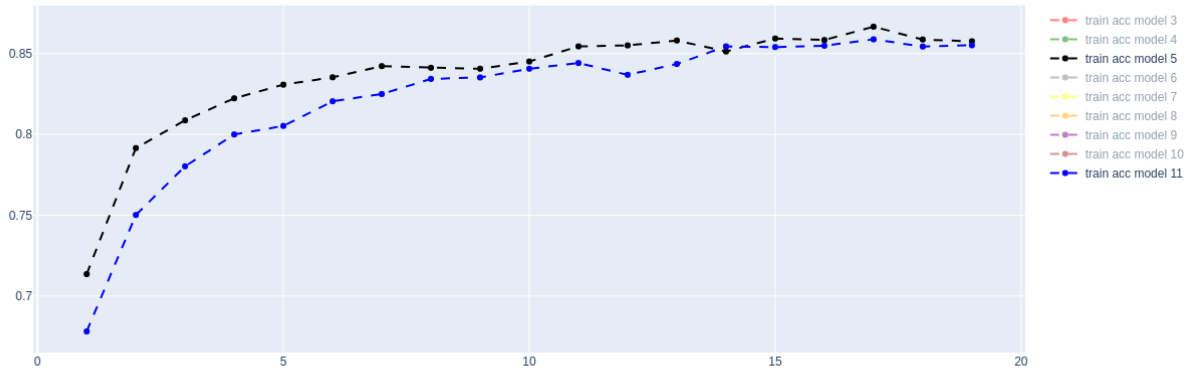

Figure 11 : training accuracy for chosen models 5 and 11.

The new parameters for the model:

- kernel size: will follow the same pattern as the model5 (also the same as VGG16 one (3 for all layers)).
- dropouts: the same as model11 (None)
- kernel reg : None
- data aug: True

We added now 8 more layers (4 conv2d and 4 batch_normalization) and boosted up the final dense layer's node (from 32 to 128). Its final architecture can be found in the *Appendix* section.

Results

Final training and validation accuracy : *0.89, 0.75* respectively

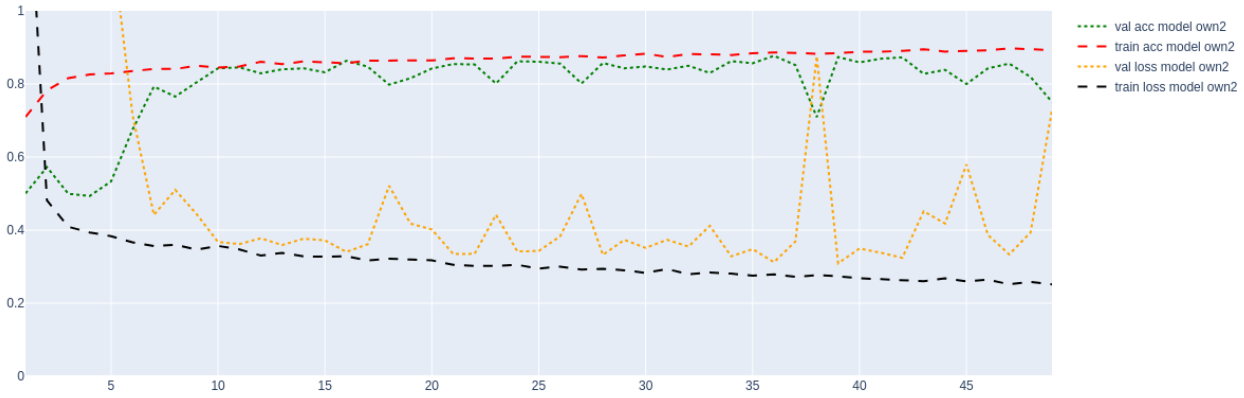

Figure 12: training and validation accuracy model *own2*.

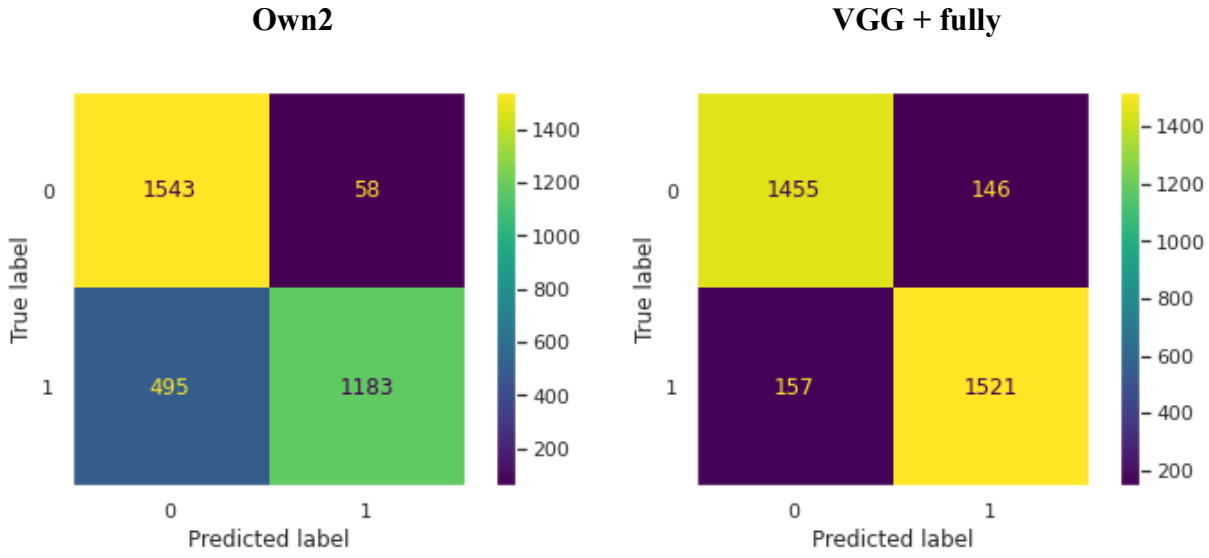

Figure 13 : comparison of confusion matrix for CNN *own2* and *VGG + fully*

While the VGG + fully had a better training and validation score, its interesting to observe the comparison above: the *own2* detected better the *non presence* of food in images, with a TN value greater than the *VGG + fully,* but was certainly worse for detecting the presence of food, with a FN almost 2,5 times greater than *VGG + fully.* The model *own2* has a more unbalanced (or irregular) behavior comparing the TP, TN, FP and FN with *VGG + fully*.

Classification Report

	precision	recall	f1-score	support
0.0	0.76	0.96	0.85	1601
1.0	0.95	0.71	0.81	1678
accuracy			0.83	3279
macro avg	0.86	0.83	0.83	3279
weighted avg	0.86	0.83	0.83	3279

Grad-CAM visualization

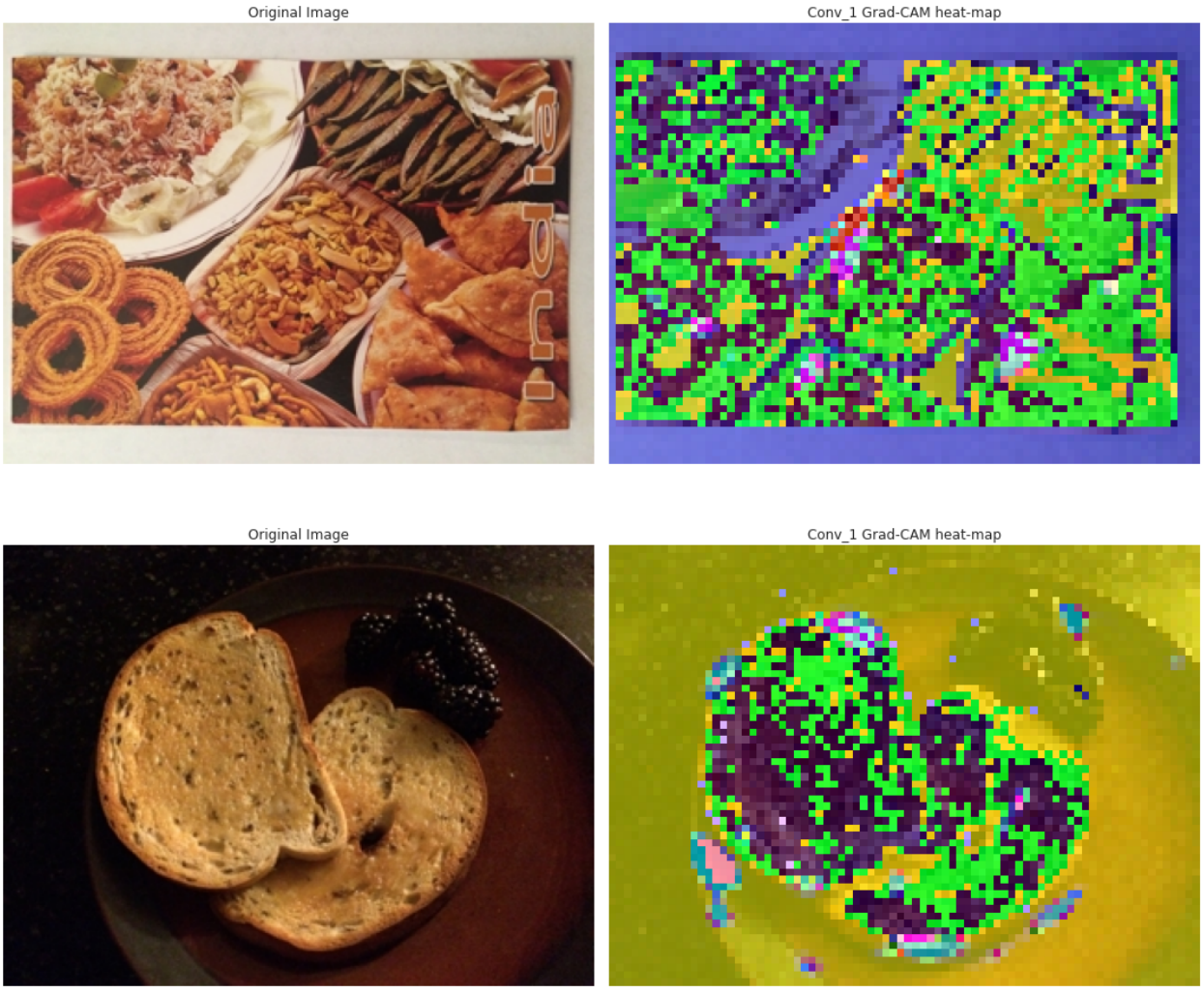

FIgure 14: grad-cam visualization for CNN *own2's* conv2d_41 layer for 3 test images. Prediction ([value]-label): [0.6117783]]-food, [0.9781345]-food and [0.00065145]-not_food, from top to bottom.

We can notice a similar behavior compared with the first new CNN architecture for the above visualizations although the different predictions.

Comparing all 3 CNN models

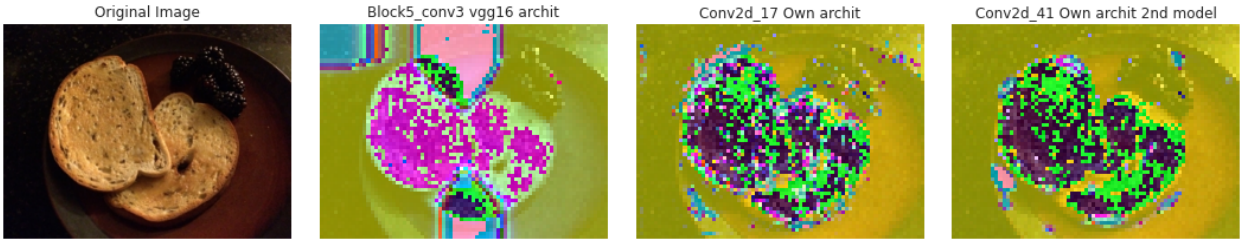

Figure 15 : grad-cam for *VGG16 + fully, own, own2* from left to right

Again, we can notice that the new CNN models (*own and own2)* give more attention to the plate despite the more sparse activation map when compared with *VGG + fully*

Bootstrap Sampling with a Decision Tree Classifier

What if we could drastically reduce the input variables and still have a reasonable model to predict?

While DNN are so commonly used when the task is to detect and classify images, some other approaches exist and sometimes are worth checking!

For the sake of curiosity, we tried a bootstrap sampling with 5 samples, only 150 input variables (PCA) and half the training data!

As we are going to train a Decision Tree Classifier, we'll be using the *data_stack* data, as the image is unrolled in one dimension.

Reducing the dimension using PCA.

To check how much information we conserve using 150 variables (a drastic change of 97% of leave out), we can plot the *cumulative explained variance plot.*

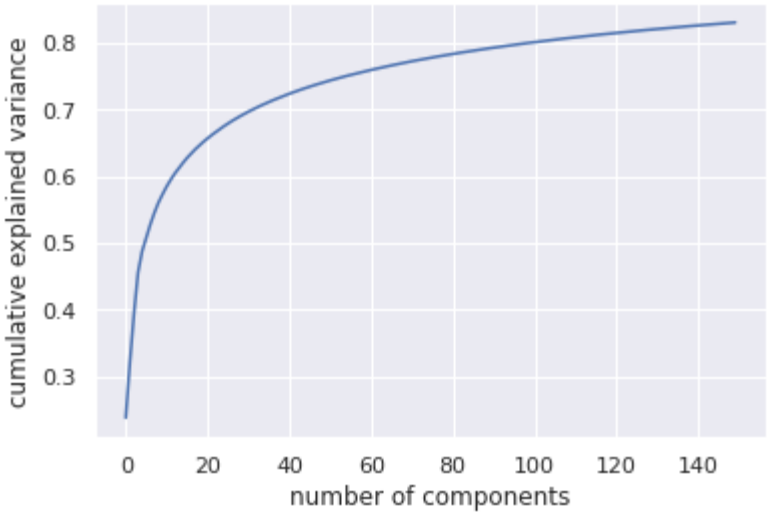

Figure 16: cumulative explained variance *vs* number of components when using PCA

From the above figure, we can observe tha reducing this reduction of dimensionality conserves around 83% of the original data information.

Results

Time consumed	149.351 s
Mean accuracy	0.751
accuracy sample 1	0.741
accuracy sample 2	0.739
accuracy sample 3	0.745
accuracy sample 4	0.764
accuracy sample 5	0.762

Table 2: bootstrap sampling accuracies for 5 samples, half training dataset and 3% of original data

Conclusion and Discussion

With the main objective of detecting the presence of food in images and testing different types of models, we tested 4 different algorithms: 1) a simple DNN; 2) VGG + fully connected layers; 3) VGG + SVM and; 4) new CNN architecture. The average accuracy for training and testing the pre-trained models were practically identical and had the highest value (*0.98 and 0.89* respectively) followed by the new CNN architecture (*1 and 0.84* respectively) and the DNN model (*0.92 and 0.69* respectively)

Taking into consideration the simplicity of the new CNN architecture (only 9 layers and 20 training epochs) we may conclude that this model has a lot of room to improve, the reason why some new attempts were further performed.

Inspired by the above results, new CNN models were trained in an iterative process to finally train a final version with a higher complex architecture (*own2*).

Although increasing the model complexity by adding 8 more layers and augmenting the data, the final training and validation accuracies reached the values of *0.89 and 0.75* respectively with 50 epochs.

This work gave a clear insight of the multiple solutions for detecting food in images and further works include:

1. for new CNN
    a. longer training
    b. change the batch size
    c. change *BatchNormalization* (or add) *MaxPooling* layers to extract most important features
2. for pre-trained *VGG + fully*
    a. add more layers and increase number of nodes
    b. longer training

Références

[1] Joutou, T., Yanai, K.: A food image recognition system with multiple kernel learning. In IEEE International Conference in Image Processing, pp. 285–288 (2009)

[2] Farinella, G. M., Allegra, D., Stanco, F., & Battiato, S. (2015, September). On the exploitation of one class classification to distinguish food vs non food images. In International Conference on Image Analysis and Processing (pp. 375 -383). Springer, Cham

[3] Farinella, G. M., Allegra, D., & Stanco, F. (2014, September). A benchmark dataset to study the representation of food images. In European Conference on Computer Vision (pp. 584 - 599). Springer, Cham

[4] Papathanail, Ioannis; Lu, Ya; Vasiloglou, Maria; Stathopoulou, Thomai; Ghosh, Arindam; Faeh, David; Mougiakakou, Stavroula (March 2021). FOOD RECOGNITION IN ASSESSING THE MEDITERRANEAN DIET: A HIERARCHICAL APPROACH (Unpublished). In: 14th International Conference on Advanced Technologies & Treatments for Diabetes.

[5] Lang, Yue & Hou, Chunping & Yang, Yang & Huang, Danyang & He, Yuan. (2017). Convolutional neural network for human micro-Doppler classification.

[6] Kanan C, Cottrell GW (2012) Color-to-Grayscale: Does the Method Matter in Image Recognition?. PLOS ONE 7(1): e29740. https://doi.org/10.1371/journal.pone.0029740 Color-to-Grayscale: Does the

[7] Kiourt, C., Pavlidis, G. and Markantonatou, S., (2020), Deep learning approaches in food recognition, MACHINE LEARNING PARADIGMS - Advances in Theory and Applications of Deep Learning, Springer

Appendix

Architecture DNN

Layer (type)	Output Shape	Param #
dense_43 (Dense)	(None, 32)	153632
dense_44 (Dense)	(None, 30)	990
dense_45 (Dense)	(None, 64)	1984
dense_46 (Dense)	(None, 64)	4160
dense_47 (Dense)	(None, 64)	4160
dense_48 (Dense)	(None, 1)	65

Total params: 164,991

Trainable params: 164,991

Non-trainable params: 0

Architecture VGG + fully connected layers

Layer (type)	Output Shape	Param #

flatten_9 (Flatten)	(None, 1024)	0
dense_10 (Dense)	(None, 256)	262400
dropout_5 (Dropout)	(None, 256)	0
dense_11 (Dense)	(None, 1)	257

Total params: 262,657

Trainable params: 262,657

Non-trainable params: 0

own CNN architecture

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 60, 80, 16)	160
conv2d_13 (Conv2D)	(None, 60, 80, 16)	2320
batch_normalization_12 (Bat chNormalization)	(None, 60, 80, 16)	64
conv2d_14 (Conv2D)	(None, 60, 80, 32)	4640
batch_normalization_13 (Bat chNormalization)	(None, 60, 80, 32)	128
conv2d_15 (Conv2D)	(None, 60, 80, 32)	9248
batch_normalization_14 (Bat	(None, 60, 80, 32)	128

chNormalization)

conv2d_16 (Conv2D) (None, 60, 80, 128) 36992

batch_normalization_15 (Bat (None, 60, 80, 128) 512

chNormalization)

conv2d_17 (Conv2D) (None, 60, 80, 128) 147584

batch_normalization_16 (Bat (None, 60, 80, 128) 512

chNormalization)

flatten_2 (Flatten) (None, 614400) 0

dense_4 (Dense) (None, 32) 19660832

batch_normalization_17 (Bat (None, 32) 128

chNormalization)

dense_5 (Dense) (None, 1) 33

==========

Total params: 19,863,281

Trainable params: 19,862,545

Non-trainable params: 736

own2 CNN architecture:

Layer (type) Output Shape Param #

==========

conv2d_32 (Conv2D) (None, 60, 80, 16) 160

conv2d_33 (Conv2D) (None, 60, 80, 16) 2320

batch_normalization_32 (Bat (None, 60, 80, 16) 64

chNormalization)

conv2d_34 (Conv2D) (None, 60, 80, 32) 4640

batch_normalization_33 (Bat (None, 60, 80, 32) 128

chNormalization)

conv2d_35 (Conv2D) (None, 60, 80, 32) 9248

batch_normalization_34 (Bat (None, 60, 80, 32) 128

chNormalization)

conv2d_36 (Conv2D) (None, 60, 80, 64) 18496

batch_normalization_35 (Bat (None, 60, 80, 64) 256

chNormalization)

conv2d_37 (Conv2D) (None, 60, 80, 64) 36928

batch_normalization_36 (Bat (None, 60, 80, 64) 256

chNormalization)

conv2d_38 (Conv2D) (None, 60, 80, 128) 73856

batch_normalization_37 (Bat (None, 60, 80, 128) 512

chNormalization)

conv2d_39 (Conv2D) (None, 60, 80, 128) 147584

batch_normalization_38 (Bat (None, 60, 80, 128) 512

chNormalization)

conv2d_40 (Conv2D) (None, 60, 80, 128) 147584

batch_normalization_39 (Bat (None, 60, 80, 128) 512

chNormalization)

conv2d_41 (Conv2D) (None, 60, 80, 128) 147584

batch_normalization_40 (Bat (None, 60, 80, 128) 512

chNormalization)

flatten_4 (Flatten) (None, 614400) 0

dense_8 (Dense) (None, 128) 78643328

batch_normalization_41 (Bat (None, 128) 512

chNormalization)

dense_9 (Dense) (None, 1) 129

==========

Total params: 79,235,249

Trainable params: 79,233,553

Non-trainable params: 1,696

_____