



# A Convolutional Neural Network application for a Food Image classification task





## Summary of the project

1. List the group members
2. Introduce the problem you are trying to solve
3. Show an analysis of the available data
4. Describe your solution
5. Comment positive and negative results of your solution (rely on quantitative and qualitative observations)
6. Describe alternative (possibly failed) experiments



## Summary of the project

1. List of the group members
  - Alessandro Maccario (Badge number: 865682)



## 2. Introduce the problem you are trying to solve

- Image Food classification is the process of identifying the type of food contained in an image. It is a challenging problem due to the wide variety of food items that exist and the variations that can occur within each type of food. Deep neural networks have proven to be effective for solving a variety of image classification tasks, including food classification.
- CNN are particularly well-suited for image classification tasks because they can learn features from images and identify patterns and characteristics that are relevant for classification;
- One of the problem encountered in this specific project is that for some categories (sushi and sashimi, for instance) the features to be learned are similar to each other and therefore is not simple, even for a human, to be able to distinguish those two types of food.



sushi



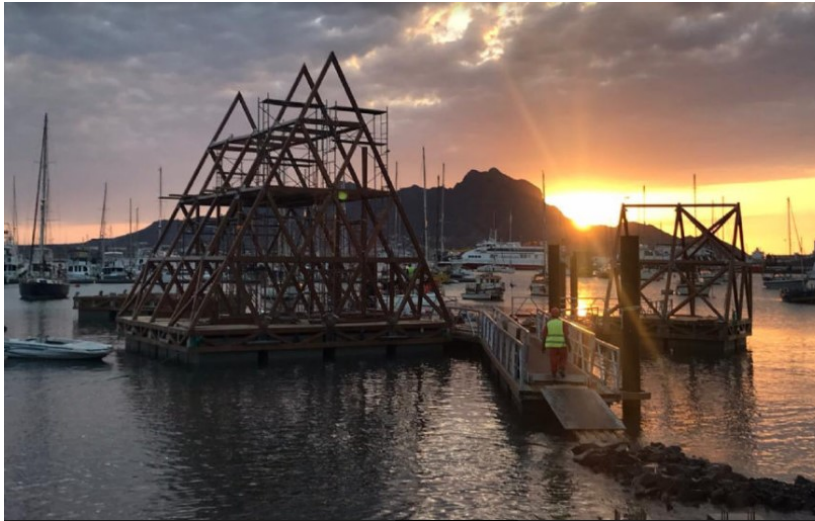
sashimi





## 2. Introduce the problem you are trying to solve

- In order to consider a *rejection class*, a script have been created to download images under *free licenses* (mostly *Creative Commons, attribution-Share-Alike*) from Wikimedia Commons.
- Some sample images are shown below, adding another categories to the one already given by the original dataset:



(1) [https://en.wikipedia.org/wiki/Wikipedia:Ten\\_things\\_you\\_may\\_not\\_know\\_about\\_images\\_on\\_Wikipedia](https://en.wikipedia.org/wiki/Wikipedia:Ten_things_you_may_not_know_about_images_on_Wikipedia)

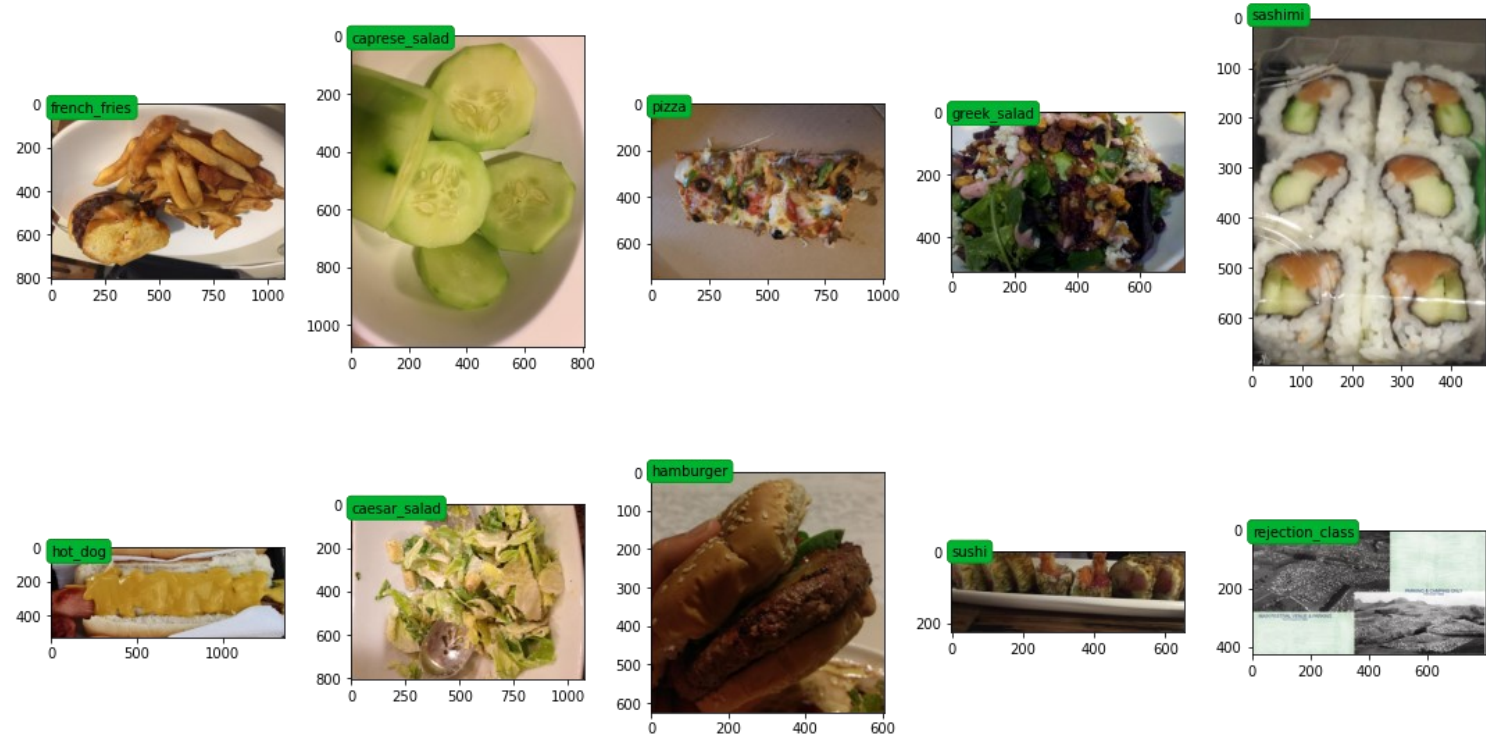
(2) [https://en.wikipedia.org/wiki/Wikipedia:File\\_copyright\\_tags#For\\_image\\_creators](https://en.wikipedia.org/wiki/Wikipedia:File_copyright_tags#For_image_creators)



### 3. Show an analysis of the available data

- In this picture it is possible to see different food classes for the problem at hand. The pictures available are also in different shapes, for instance:
  - french\_fries: (756, 1008, 3)
  - caprese\_salad: (1078, 808, 3)
  - pizza: (782, 1056, 3)
  - greek\_salad: (159, 212, 3)
  - sashimi: (389, 422, 3)
  - hot\_dog: (1650, 1081, 3)
  - caesar\_salad: (808, 1078, 3)
  - hamburger: (549, 502, 3)
  - sushi: (808, 1078, 3)
  - rejection\_class: (426, 800, 3)
- Before feeding them into the models, the preprocessing procedure involves a resizing of the images.

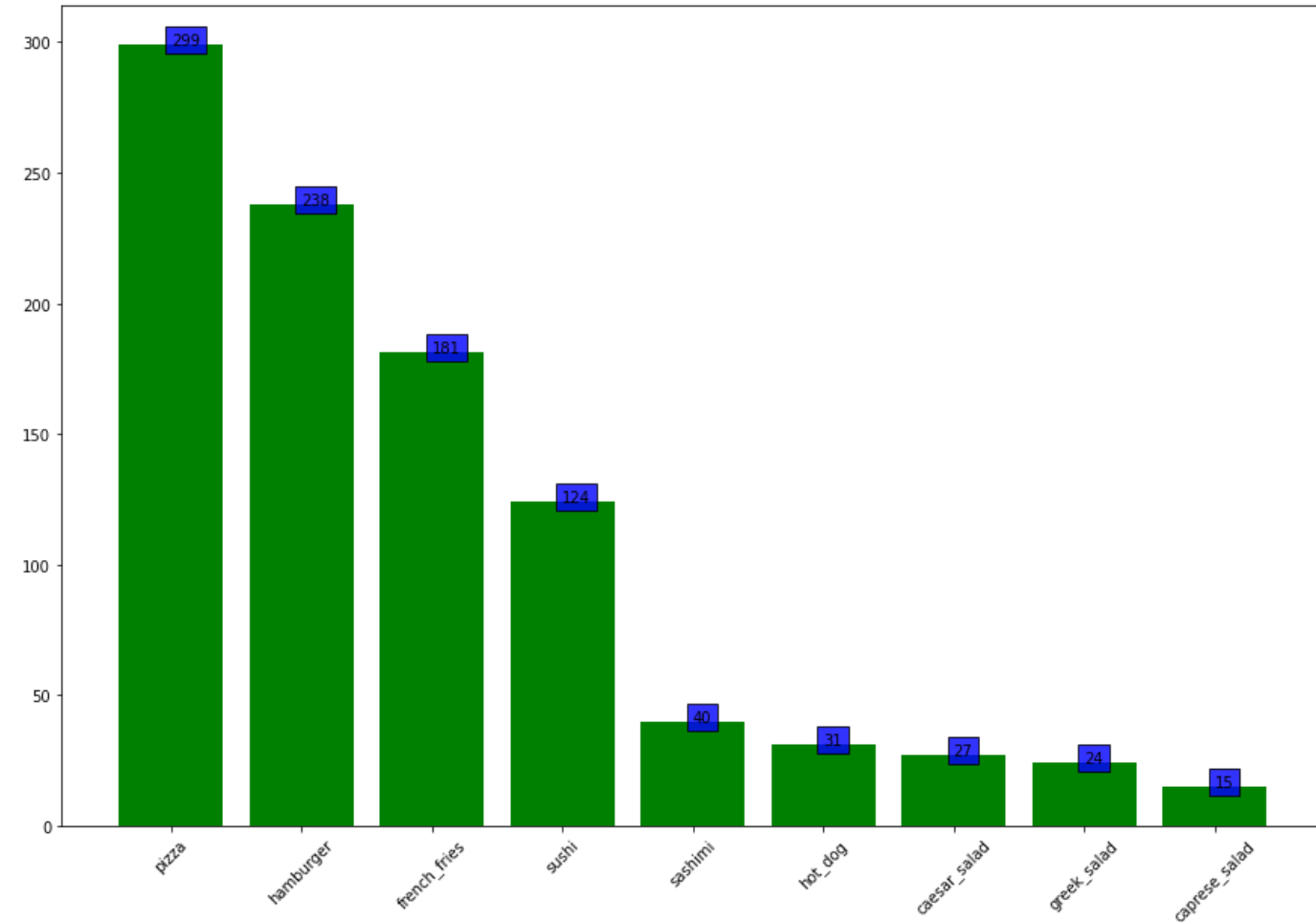
Random images from each food class and rejection class





### 3. Show an analysis of the available data

- The *Example Food Images* data set is divided in 9 different categories, such as Caesar salad, caprese salad, French fries, Greek salad, Hamburger, hot dog, pizza, sashimi, and sushi. It contains 978 photographs of food and the size of the data set is about 77 MB.
- The problem itself is an *unbalanced problem*, that is the number of examples in different classes is not equal. The model in this case can have difficulties to learn the classes with fewer data points. In order to overcome this problem a Data Augmentation technique have been performed, obtaining 200 images for each category on the training set.
- The training set contains 2003 images, the validation 242 images and the test set 193 images.





### 3. Show an analysis of the available data

- In order to overcome the previous challenge, a **data augmentation** procedure has been applied in order to create slightly different copies of the data right after separating the data in training, testing and validation folders. The training folder has been augmented reaching 200 images for each classes, before feeding the data into the model. The augmentation procedure involved horizontal and vertical flipping, brightness and hue modification.



Original fries  
image



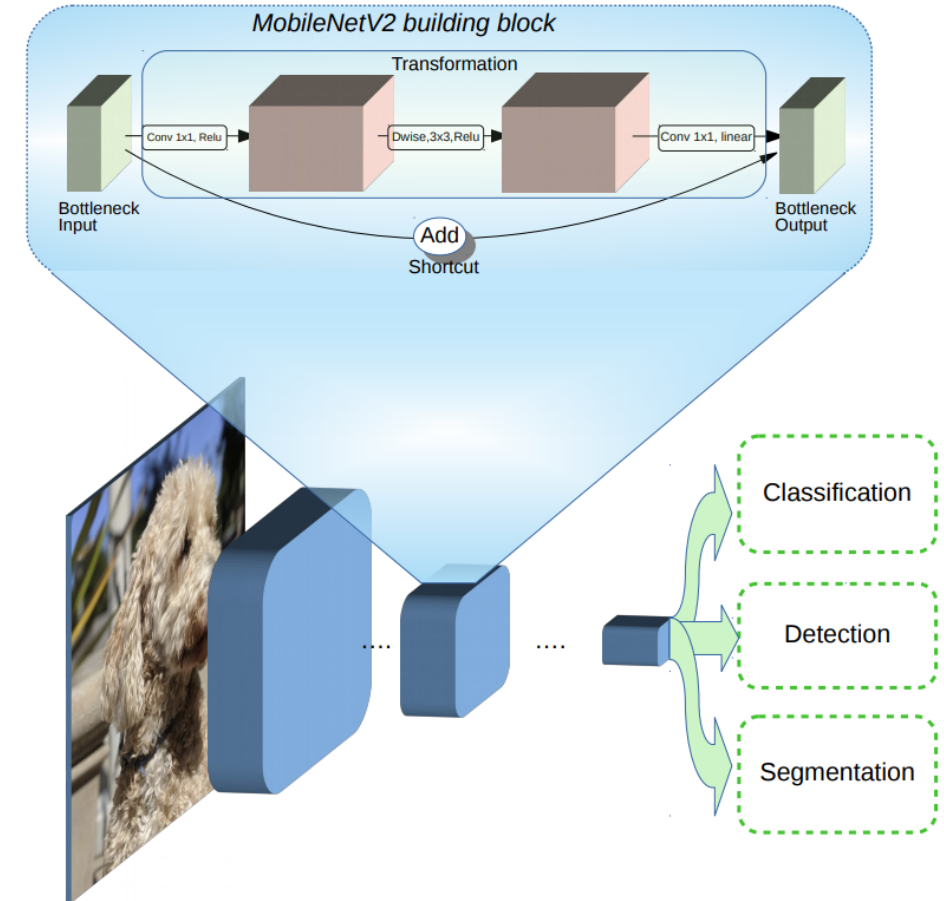
Horizontal flip  
of fries image





## 4. Describe your solution

- The solution applied involved the usage of the *MobileNetV2* (1) architecture pretrained on the *Imagenet dataset* (2) in order to use the concept of *Transfer Learning*, to apply the features learned by the model on another bigger dataset, on the one available in this project.
- *MobileNetV2* is a convolutional neural network (CNN), lightweight model that is well-suited for mobile and embedded applications, and can run efficiently on devices with limited computational resources.



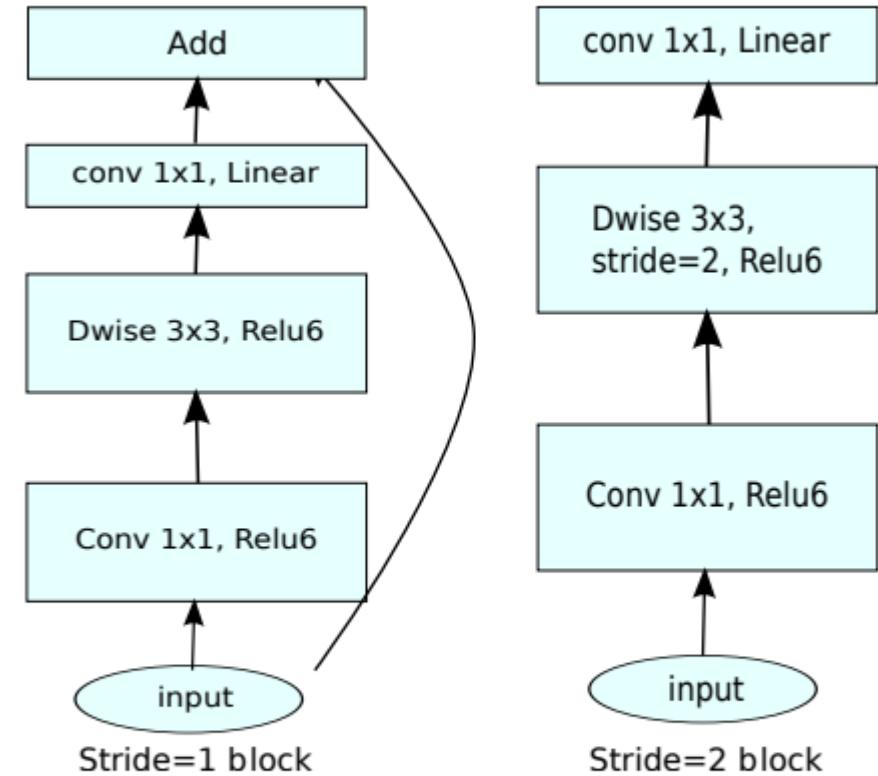
## Architecture

(1) <https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html>



#### 4. Describe your solution

- The model involved a new layer called *inverted residual with linear bottleneck*
- The *inverted* part of the name refers to the fact that the input and output connections are swapped compared to a traditional residual block. If in a traditional residual block, the input is added to the output of the block, in an inverted residual block the output of the block is added to the input.
- The *linear bottleneck* refers to the use of a 1x1 convolutional layer, also known as a *pointwise convolution*, as a bottleneck to reduce the number of channels in the output of the block.



(d) Mobilenet V2  
Architecture

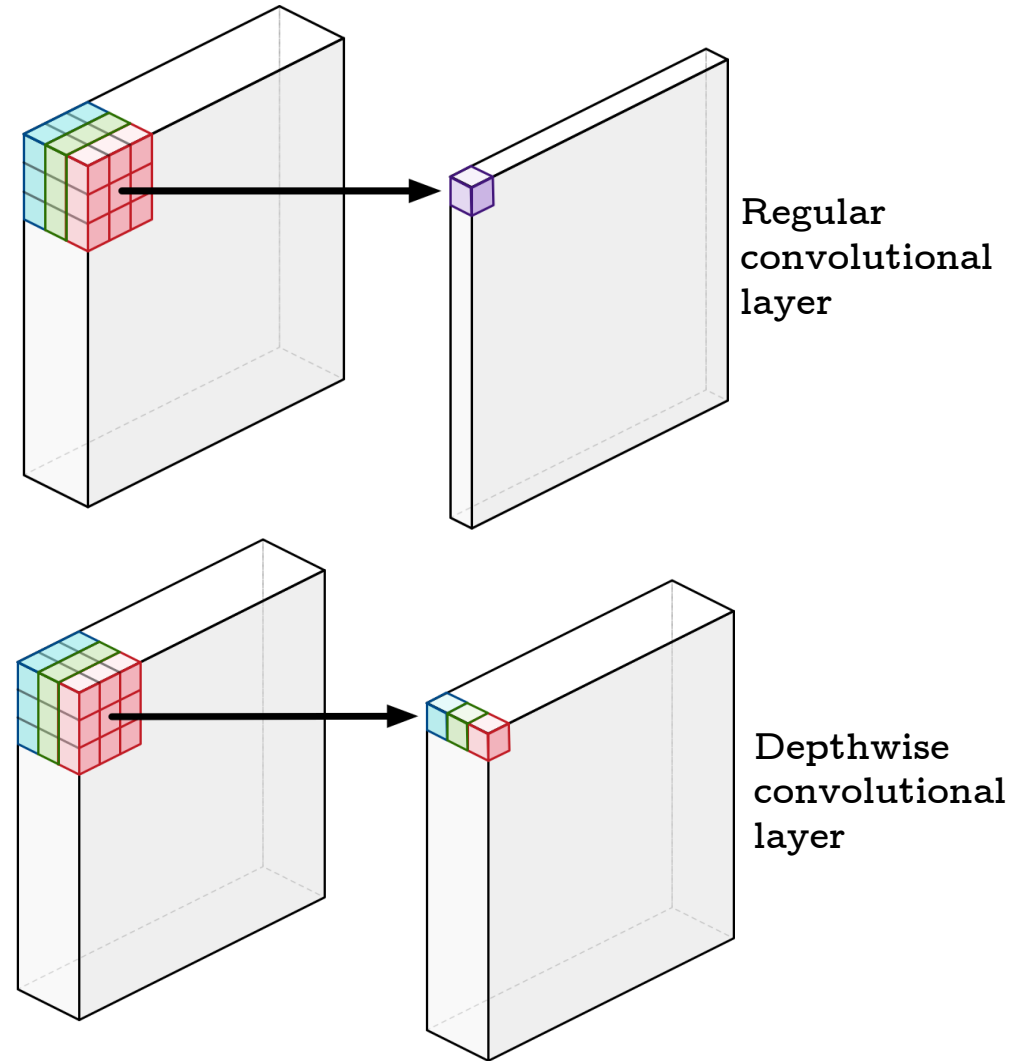
(1) <https://arxiv.org/pdf/1801.04381v4.pdf>

(2) <https://www.image-net.org/>



#### 4. Describe your solution

- The most important idea about MobileNets is that it uses *depthwise separable convolutions* to build light-weight deep neural networks. A regular convolutional layer applies a convolution kernel (or so-called filter) to all of the channels of the input image, sliding this kernel across the image and at each step and performing a weighted sum of the input pixels covered by the kernel across all input channels. The convolution writes a new output pixel with only a single channel.
- The MobileNets architecture uses this standard convolution, but only as the very first layer. All other layers apply a *depthwise separable convolution* instead.
- A main advantage is that with a depthwise separable convolution these two operations are done as separate steps. Both of the methods filter the data and make new features, but where a regular convolution has to do much more computational work to get to the final results and needs to learn more weights, the *depthwise separable convolution is going to be much faster*.



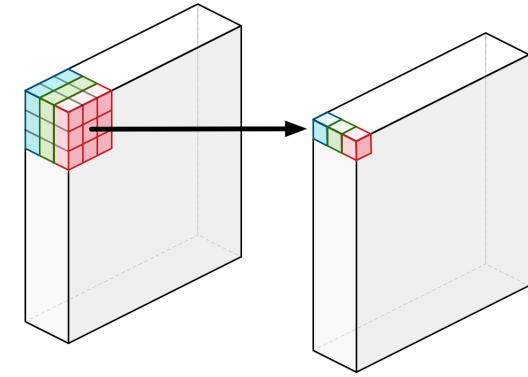
(1) <https://machinethink.net/blog/googles-mobile-net-architecture-on-iphone/>

(2) <https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html>

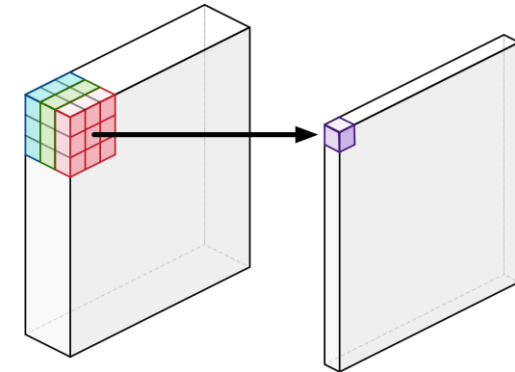


#### 4. Describe your solution

- This *depthwise separable convolution* is combination of two different operations:
  - *depthwise convolution*: the difference compare to a regular convolution is that it does not combine the input channels but it performs *convolution* on each channel separately. For an image with 3 channels, a depthwise convolution creates an output image that also has 3 channels. Each channel gets its own set of weights. The purpose of the depthwise convolution is to *filter* the input channels.
  - *pointwise convolution*: this is the same as a regular convolution but with a  $1 \times 1$  kernel. In other words, this simply adds up all the channels (as a weighted sum). As with a regular convolution, we usually stack together many of these pointwise kernels to create an output image with many channels. The purpose of this pointwise convolution is to *combine* the output channels of the depthwise convolution to create new features.



Depthwise  
convolution



Pointwise  
convolution





\*In metadata, a **synonym ring** or **synset**, is a group of data elements that are considered semantically equivalent for the purposes of information retrieval.

<https://en.wikipedia.org/wiki/Synset>

## 4. Describe your solution

- MobileNetV2 have been trained using the weights learned from the ImageNet dataset that contained 1000 classes.
- ImageNet is defined as a largescale ontology of images built upon the backbone of the WordNet structure. It aims to populate the majority of the 80.000 synsets\* of WordNet with an average of 500 - 1000 clean and full resolution images. This will result in tens of millions of annotated images organized by the semantic hierarchy of WordNet.
- The last layer has been freezed and connected to another Dense layer with 512 neurons and then to the final Dense layer with the softmax activation function, in order to obtain the class probability distribution necessary to understand the prediction of the model.

IMAGENET



Figure 1: A snapshot of two root-to-leaf branches of ImageNet: the **top** row is from the mammal subtree; the **bottom** row is from the vehicle subtree. For each synset, 9 randomly sampled images are presented.

(1) <https://machinethink.net/blog/googles-mobile-net-architecture-on-iphone/>

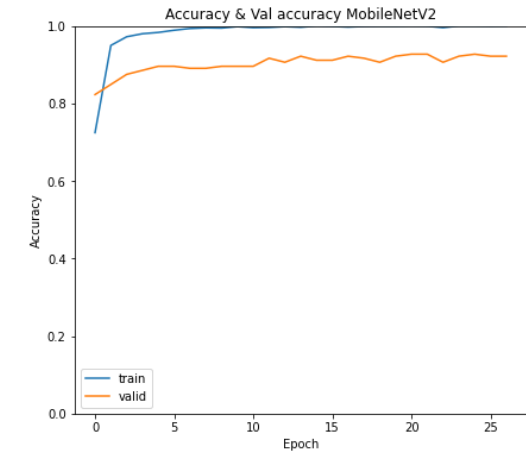
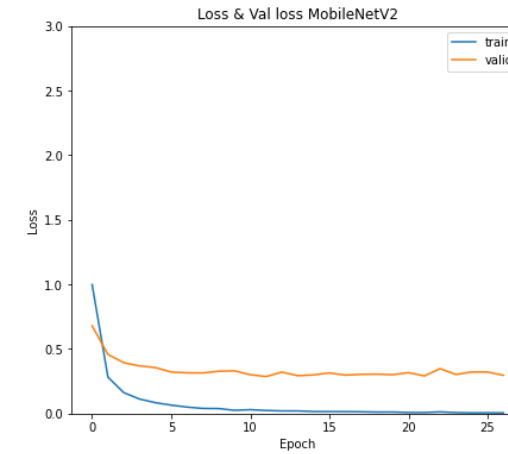
(2) [https://www.image-net.org/static\\_files/papers/ImageNet\\_2010.pdf](https://www.image-net.org/static_files/papers/ImageNet_2010.pdf)



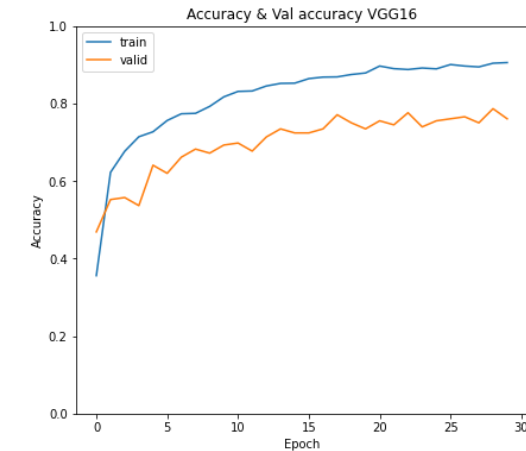
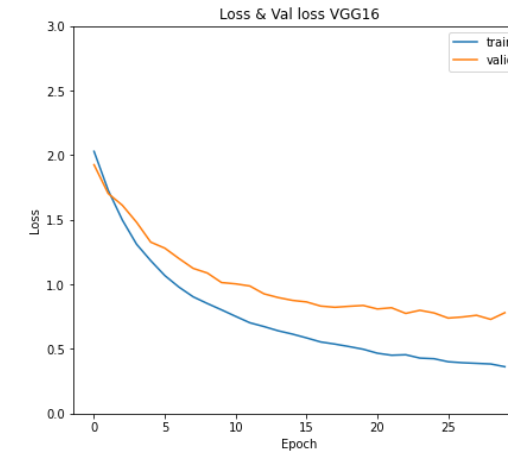
## 4. Describe your solution

- Different other models have been trained in this project: custom models in order to try out how one can create them from scratch. Performances have increased from the first model (around 40%) to more than 60% in the last model before applying the transfer learning techniques.
- Other models have also been trained (VGG16, ResNet50), but from the *accuracy metric* point of view, only MobileNetV2 has been able to outperform the other models in only 30 epochs reaching over 92% accuracy both for the accuracy on the training set and validation set.
- The value of the *recall* metric using the MobileNetV2 model is also the highest value, over 90%: that means that the model is able to correctly classify, for each class, almost all the images that are fed into it.

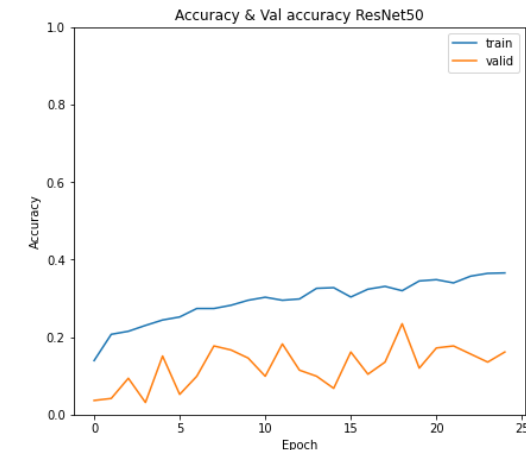
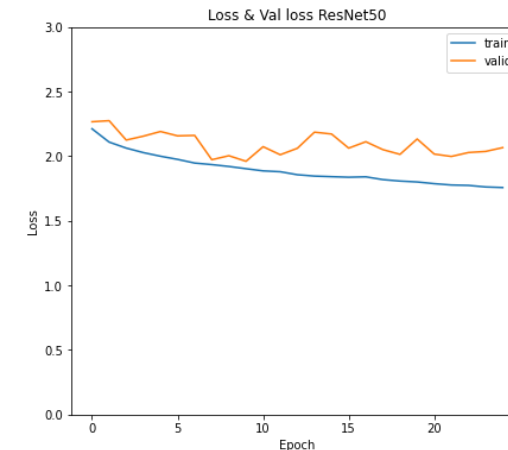
MobileNetV2



VGG16



ResNet50





#### 4. Describe your solution

- More closely, the numerical results are the following (using the hyperparameters shown):

	Performance	MobileNetV2	VGG16	ResNet50
★	minAccuracy	72,47%	35,59%	13,93%
	minValAccuracy	82,29%	46,88%	3,13%
★	maxAccuracy	100,00%	90,54%	36,54%
	maxValAccuracy	92,71%	78,65%	23,44%

★ With min/maxAccuracy is meant w.r.t. the training set while fitting the model

#### Hyperparameters

batch\_size = 16  
loss = Categorical crossentropy  
optimizer = Adam  
learning\_rate = 0.0001

- As it is clear, MobileNetV2 outperforms the other two transfer model in only 30 epochs.



## 5. Comment positive and negative results of your solution (rely on quantitative and qualitative observations)

### Positive results (+)

- Accuracy results obtained by MobileNetV2 and recall values are over 90%, as for the results on the test set;
- MobileNetV2 is faster than other more complex models as VGG16 and ResNet50
- The solution presented consider also a rejection class: the model is able to return a class that is not food; furthermore, another option that the model has is, in case the probability computed is lower than 50%, it can reply that is not able to provide a clear answer to the image feeded.

### Negative results (-)

- The amount of original images, less than a 1000, required Data Augmentation, that could be the reason why it performs very well, but it seems not able to reach an accuracy greater than 92% on the validation set.
- Some images are difficult to recognize, also for a human being: for instance, in some images in the sushi folder, they contain both sushi and sashimi but the images itself are labeled as sushi. This can lead the model to perform not as its maximum capacity.








## 5. Comment positive and negative results of your solution (rely on quantitative and qualitative observations)

### Negative results (-)

- One last negative result comes from the deployment notebook. Even though for most of the images the model run well, for some of them is incapable to give a correct prediction or, at least, to state that is not able to recognize a certain output. Instead, it seems confident in giving a prediction amongst the food categories, like in the example shown here.



img\_loading


Clear Submit

output

french\_fries

french_fries	99%
hot_dog	0%
hamburger	0%

Flag Interpret



img\_loading

Clear Submit

output

sashimi

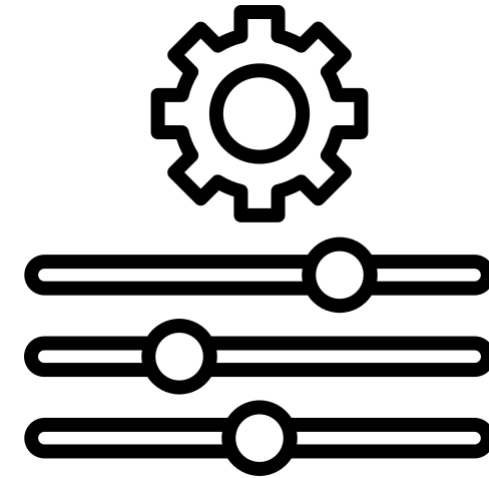
sashimi	100%
sushi	0%
pizza	0%

Flag Interpret



## 6. Describe alternative (possibly failed) experiments

- One attempt was made in order to apply GridSearchCrossValidation on the deeper Model V04, with the goal of finding out which are the best permutations of parameters that perform better on this data.
- Even though the results showed the usage of a *learning\_rate* = 0.001 and an *activation\_function* = LeakyReLU, multiple experiments with this hyperparameters have shown that the best set of them is the one that involves a *ReLU* activation function and a learning rate of 0.0001.
- This is the reason why this hyperparameters have been chosen for the Transfer Learning with the MobileNetV2 architecture.

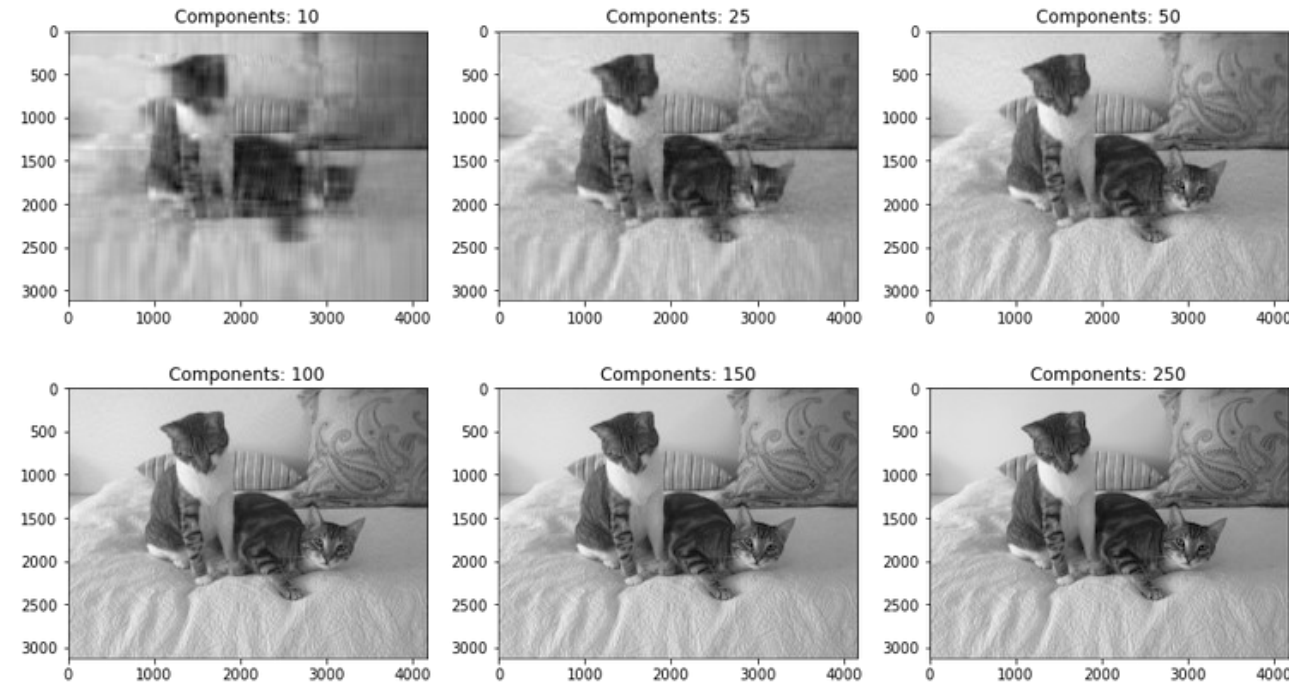




## 6. Describe alternative (possibly failed) experiments

- An alternative experiment involved the usage of the Principal Component Analysis in conjunction with Transfer Learning using MobileNetV2 architecture.
- The idea behind that experiment was to reduce the number of parameters involved in the model (considering only the images, after resizing them by 224x224, the number of parameter in total has risen to 50.176) that, in total, the model has to learn in order to predict the new image.
- The experiment succeeded: in fact, during the training phase, the fitting speed was much higher than the traditional methods. Nevertheless, the final accuracy was around 10% lower than when using the full amount of parameters with MobileNetV2 architecture and Transfer Learning (around 80%)

Original image  
Size: (3120, 4160,3)



With PCA we only need to use 54 of the original 3120 components to explain 95% of the variance in the image.





THANK YOU FOR  
YOUR ATTENTION!







## REFERENCES

### DATASET:

- <https://de.mathworks.com/help/deeplearning/ug/data-sets-for-deep-learning.html>

### PAPERS:

- [https://link.springer.com/content/pdf/10.1007/978-3-319-23222-5\\_41.pdf](https://link.springer.com/content/pdf/10.1007/978-3-319-23222-5_41.pdf)
- <http://madima.org/wp-content/uploads/2017/11/12-A0-ICIAP-foodCNN-poster.pdf>
- [https://boa.unimib.it/bitstream/10281/206401/4/CNN-based%20features\\_post-print.pdf](https://boa.unimib.it/bitstream/10281/206401/4/CNN-based%20features_post-print.pdf)
- <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8228338>
- <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8550005>
- <https://arxiv.org/pdf/1612.00983.pdf>
- <https://www.irjet.net/archives/V9/i3/IRJET-V9I3368.pdf>
- <https://www.irjet.net/archives/V9/i1/IRJET-V9I131.pdf>
- [https://image-net.org/static\\_files/papers/imagenet\\_cvpr09.pdf](https://image-net.org/static_files/papers/imagenet_cvpr09.pdf)
- <https://medium.com/analytics-vidhya/indian-food-image-classification-using-transfer-learning-b8878187ddd1>
- <https://wordnet.princeton.edu/>
- [https://github.com/abhijeet3922/Food-Classification-Task-Transfer-learning/blob/master/Food\\_classification\\_transfer\\_learning\\_finetuningVGG16\\_30072019.ipynb](https://github.com/abhijeet3922/Food-Classification-Task-Transfer-learning/blob/master/Food_classification_transfer_learning_finetuningVGG16_30072019.ipynb)
- <https://www.image-net.org/>
- <https://towardsdatascience.com/image-compression-using-principal-component-analysis-pca-253f26740a9f>
- <https://www.kaggle.com/code/poigal/transfer-learning-feature-extraction-and-pca>
- <https://github.com/Murgio/Food-Recipe-CNN>
- <https://www.dataknowsall.com/imagepca.html>
- <https://www.kaggle.com/code/themlphdstudent/food-classification-using-inceptionv3>