91250 - Deep Learning

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Final Project

Food Recognition

AUTHORS:

Gonzalo Joaquín Davidov (1900092260)

Facundo David Farall (1900092175)

Rafael Nicolás Trozzo (1900092496)

TEACHER:

Andrea Asperti

BOLOGNA

May 2021

Contents

1 Introduction **3**

2 Problem Definition **3**

3 Proposed Model **3**

4 Problem Approach **5**

4.1 Variables of Interest 5

4.1.1 Backbone 5

4.1.2 Activation 5

4.1.3 Loss Function 6

4.1.4 Class Weights 6

4.2 Evaluation Metrics 7

4.2.1 IoU 7

4.2.2 Only True IoU 7

4.3 Resources Management 8

5 Results Analysis **8**

5.1 Training 8

5.2 Compilation Conditions 10

5.3 Threshold Selection 10

5.4 Background Inclusion 11

5.5 Backbones 13

5.6 Channel Correction 17

5.7 Results Summary 18

6 Conclusions **18**

7 Bibliography **19**

1 Introduction

The aim of this project is to train a DNN using image segmentation in order to be able to identify and classify different food categories. Different backbones for a U-Net architecture will be shown and evaluated with variants on the activation function, loss function, metrics and categories to find and justify which is the more adecuate for this problem.

2 Problem Definition

The problem to solve is the one defined in the AICrowd Food Recognition Challenge, the goal is to train a model which can look at images of food items and detect the individual food items present in them, producing an output similar to the one depicted in Figure 2.1.

FIGURE 2.1: Food Recognition Example

This is a segmentation problem where each pixel can belong to one or more categories, therefore properly annotated data is needed. The dataset provided by the challenge consists of 24120 training images and 1269 validation images, each of them with their corresponding annotations in the MS-COCO format. In order to make the training more feasible given the limited amount of resources, in this project not all the categories are used, but just the 16 most annotated.

3 Proposed Model

The chosen architecture for the problem resolution is U-Net [3] because it is more successful than conventional models, in terms of architecture and in terms of pixel-based image

segmentation formed from convolutional neural network layers. Another really important aspect about U-Net is that it also works really well even with small datasets.

As it can be seen in Figure 3.1, the architecture is called U-Net because of its shape similar to the letter U. The left side is an encoder path and the right side, a decoder path. The contracting path is a typical CNN that consists of repeated application of convolutions, each followed by a ReLu and a max pooling operation. During the encoding, the spatial information is reduced while feature information is increased. The decoding pathway combines the feature and spatial information through a sequence of up-convolutions and concatenations with high-resolution features from the encoding path.

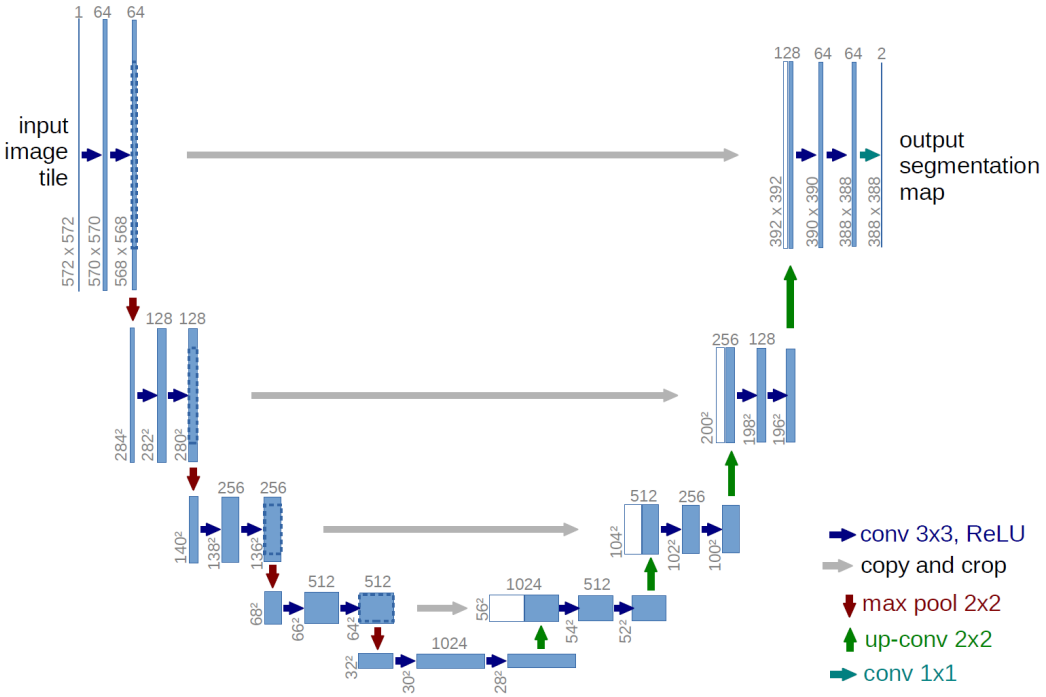


FIGURE 3.1: U-Net Architecture

Training this net is a really expensive process, that is why, it has been decided that the best option is to use a pre-trained model trained on ImageNet to replace the encoder path and use it's weights. The models in Table 3.1 are to be tested to see which one performs better on a food recognition task.

Backbones
VGG16 [4]
VGG19 [4]
ResNet50 [1]
EfficientNetB5 [5]

TABLE 3.1: Pre-trained Models

4 Problem Approach

In search of the best possible solution for this problem, a study of the most relevant variables was conducted in order to adequately select those within our capability of handling, and adjust them to obtain a better performance of the model. Such variables and a short explanation of them are presented in the following section.

With the purpose of properly evaluating the impact of the variations introduced in said variables, a set of two objective metrics was introduced, to complement the subjective visual inspection of the predictions. The fundamentals of this metrics, and the reasons behind them will be discussed in section 4.2.

4.1 Variables of Interest

4.1.1 Backbone

It is of out-most importance for the success of the model, a proper extraction of the input-image features, and that is where the encoder does its job. The selection of the encoder's architecture will benefit some features' detection over others, and this will have an impact later on the segmentation. Which is the best architecture for our dataset is not known beforehand. Therefore, partly based on recommendations found on various articles on this matter, and an arbitrary decision, VGG19 was selected as the first backbone for the first working model, and then compared with the rest.

In addition, the models developed for this project will be evaluated against those provided by the library Segmentation Models [6], according to our metrics. The reason behind this is to compare the solutions proposed to some previously developed and studied solutions for this tasks. The library offers the same U-Net architecture used in this project, as well as others that will not be considered. Furthermore, if offers a set of backbones to choose from, including all the ones mentioned in this report (Table 3.1)

4.1.2 Activation

Specifically the activation of the last layer, the one in charge of classifying the category each pixel belongs to. To begin with, the output should be a probability, consequently *softmax* and *sigmoid* arise as principal candidates. However, both appear in principle to be suitable for different kind of problems. While *softmax* presents itself as a better choice for a multi-class type of problem, where each pixel belongs to one and only one of a set of classes to classify, *sigmoid* offers the possibility to classify each class independently, therefore allowing a pixel to belong to more than one class (i.e. a multi-label type of problem).

Upon inspection of the dataset provided, it is evident that overlapping of the masks is

present, favouring in principle an output with *sigmoid*.

4.1.3 Loss Function

Typical loss functions applied in classification and segmentation problems such as this one are the Jaccard Loss, Dice Loss and Focal Loss [2]. Each of them focuses on a different aspect of the prediction obtained, and for this project each of them, or a combination, will be used as loss function for training the model.

Jaccard Loss: It is based on the intersection over union coefficient explained later in section 4.2.1. This metric yields a result between 0 and 1, where 1 is optimal. The Jaccard Loss is thus defined as:

$$JaccardLoss = 1 - \frac{Area\ of\ overlap}{Area\ of\ union} \tag{4.1}$$

Sørensen–Dice Loss: Based on the Sørensen–Dice coefficient, which also provides a measure of how similar two samples are. It is defined as the quotient between 2 times the common elements between samples, over the total amount of elements including both samples. And so the loss is computed as:

$$DiceLoss = 1 - \frac{2 \cdot |X \cap Y|}{|X| + |Y|} \tag{4.2}$$

The Sørensen–Dice coefficient differs from the IoU index in the sense that the latter only accounts for true positives.

Focal Loss: Variation of the classic Binary Cross Entropy loss that weights the comparisons and is therefore particularly useful in image segmentation, where most of the pixels are generally background, and those of interest are small portions of the whole image.

4.1.4 Class Weights

Another factor worth considering is whether or not the classes are weighted somehow, giving more importance to the learning of some categories over others. For the current study, some models where trained without weighting classes (all have the same importance), and others with. In the latter, the weights were assigned inversely proportional to the amount of pixels over the total that each class has. For example, if a class has 56 pixels over a total of

100, its weight is going to be proportional to 100/56. We use the word "proportional" here because after this inversion of the pixel factions of each class, all the weights are normalised.

4.2 Evaluation Metrics

To be able to compare and decide over the different backbones, activation functions and other variable choices, the evaluation metrics used are Intersection over Union (IoU) and Only True IoU: a modified version developed specifically for this project, to better and objectively represent what humans consider a "good" or "bad" detection.

4.2.1 IoU

The IoU metric is the average of the IoU of the whole validation set over every channel, except the background (when used), since is not in our interest. It is an evaluation metric used to measure the accuracy of the segmentation model on a particular dataset. As the name says, it can be calculated by:

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union} \tag{4.3}$$

Where the area of overlap is the intersection between the predicted mask and the ground-truth mask and the area of Union is the composition of both the predicted mask and the ground-truth mask.

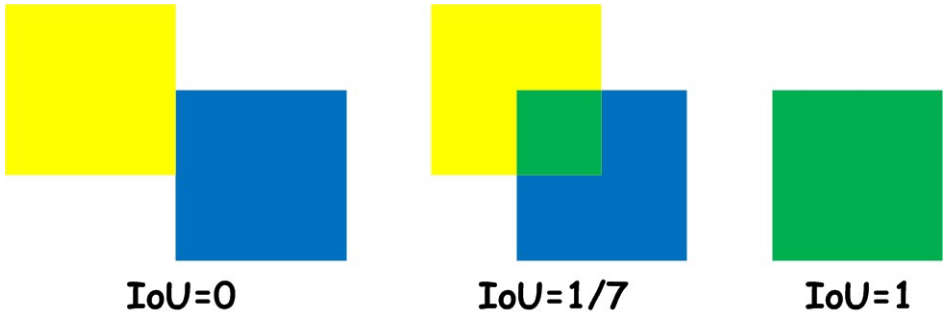


FIGURE 4.1: IoU example

4.2.2 Only True IoU

Even though the IoU metric is highly used for segmentation problems and with success, in the particular case of the problem at hand, the dataset includes many empty channels. In other words, most of the images only have pixels belonging to one or two of the 16 categories studied. This results in the fact that 0 pixels detected when the mask is empty, is highly rewarded with a score of 1, rising considerably the average score, given that this is the case for most of the masks.

Even though is true that the model should be positively rewarded when it does not show a detection where there was not supposed to be one, the results are visually better when the detection of an object that should be detected is accurate.

To this end, the Only True IoU averages only the masks in which the Union is different than 0, i.e. when there is a prediction for that category and/or the ground-truth is non-zero (in this case, the "or" is not exclusive)

4.3 Resources Management

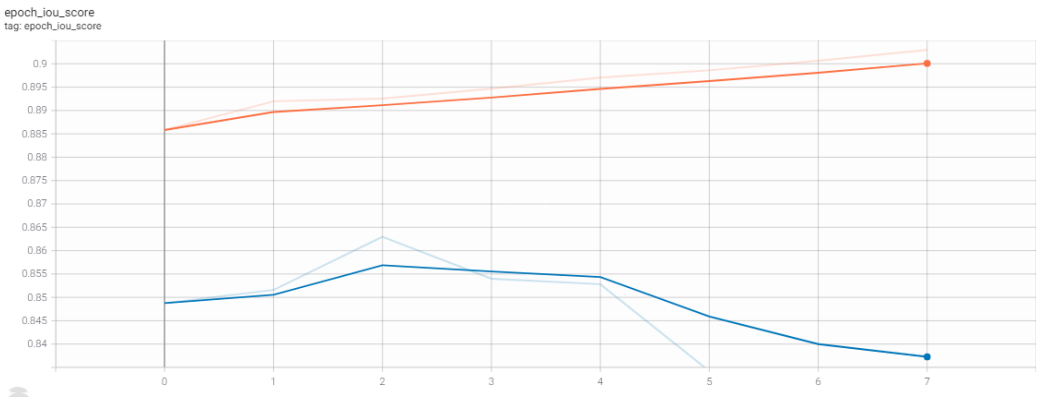
Training is a highly demanding task in terms of both computing and memory usage, so it is important to use properly the available resources. For that purpose, Google Colaboratory is used since it provides GPU, which speeds up the training some orders of magnitude. Since the dataset is large, it is not possible to load it completely in RAM , so the chosen approach is to load all the dataset in Colab's disk, from where files can be quickly read. To sort the RAM issue, Python generators are used for training, enabling data to be read in batches during training, therefore using memory more efficiently.

5 Results Analysis

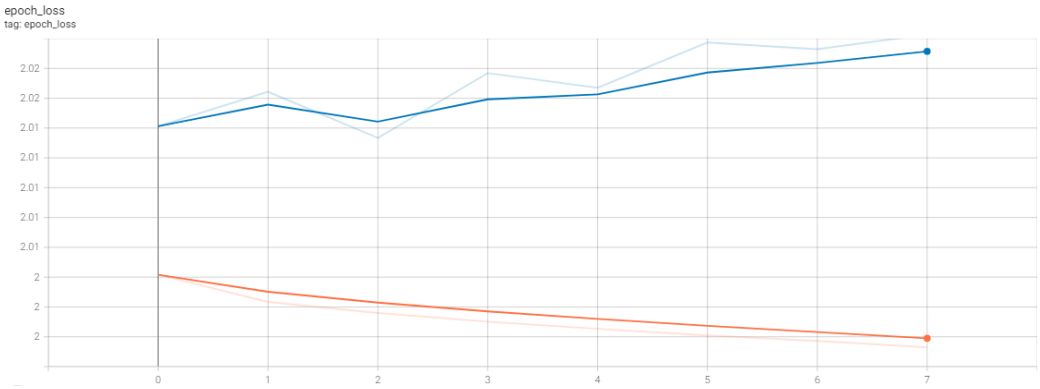
In the following section the results for the variation of each one of the variables of interest is shown, and finally a table is presented to be able to visualize all the results summarised.

5.1 Training

During the training of the different models, logs where generated to be able to use the TensorBoard tool and plot interesting information about the training process and see, for example, where the network stopped learning or how the IoU changed over each epoch. The following Images are just some examples of 2 models, an EfficientNet and a VGG16.

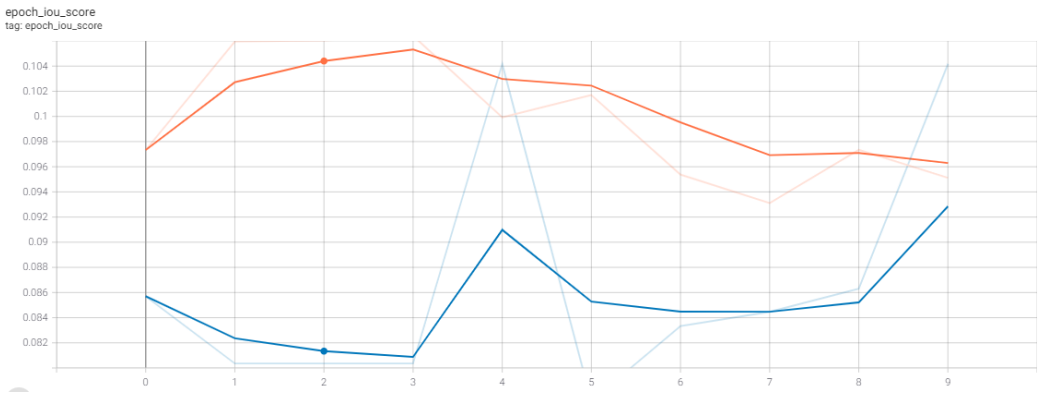


(A) IoU score

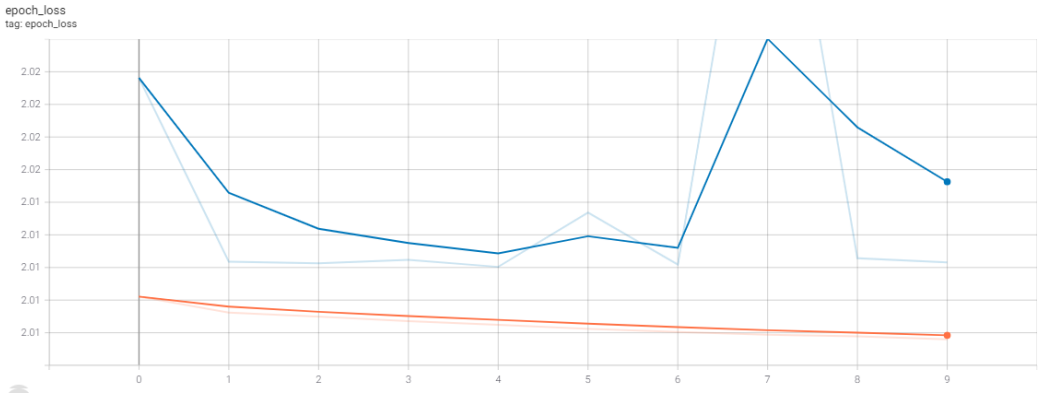


(B) Loss

FIGURE 5.1: TensorBoard graphs VGG16



(A) IoU score



(B) Loss

FIGURE 5.2: TensorBoard graphs EfficientNet

The orange trace represents the train set while the blue one, the validation set.

5.2 Compilation Conditions

Output Layer Activation: Even though in Section 4.1.2 it was said that the most suitable activation function for this task is *sigmoid*, considerably better results were obtained when using *softmax*, so it is the one used, at the expense of not permitting the detection of overlapped items. In short, a better overall result is preferred over a small number of overlapping cases.

Loss Function: As a first approach, each of the loss functions described in 4.1.3 were considered separately. However, in every case, the results of the predictions were lacking on some way or another. For instance, the Jaccard Loss was good at optimising the IoU metric (which is no other than our main metric of interest), but the predictions were visibly inaccurate, as the network tended to classify most of the background, or any of the pixels that could not clearly classify with another label, as the class that was present the most (in our case, water). In the end, the best results where achieved by a balanced combination of the three of them.

Variation of Pretrained Weights: The variation of the backbone's pretrained weights while doing the transfer learning was tested but the results were far from what was expected, probably because the dataset is not big enough. The weights already trained on a dataset as big as ImageNet proved to be better for feature extraction and improve the quality of the recognition.

Class Weights: Applying class weights in the way described in section 4.1.4 did not prove to have a substantial effect the capabilities of the model. Perhaps another method of weighting could have brought up better results, however that is left for further studies.

5.3 Threshold Selection

For each network trained, the validation is run for different thresholds to see which one is better for each case, having as a result a chart like the one in Figure 5.3. In the case depicted, the best threshold is 0.95 so it is the one used for the metrics calculations.

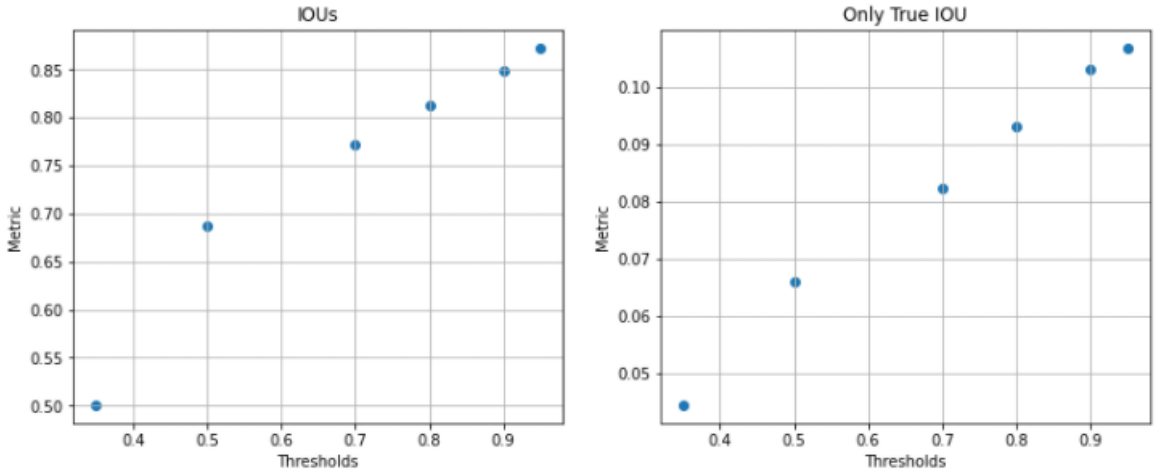


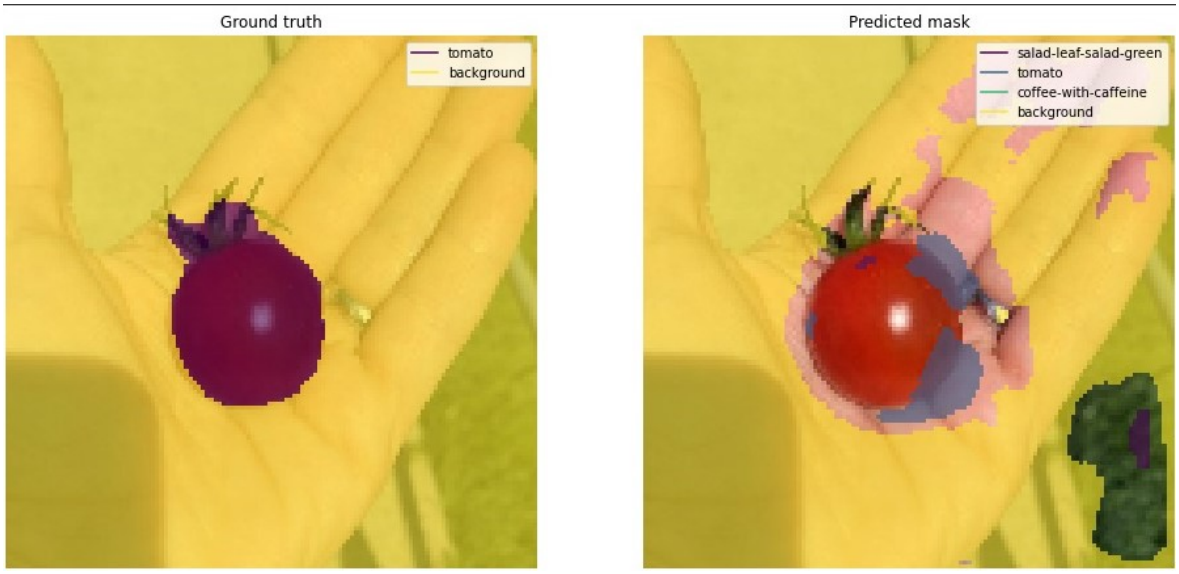
FIGURE 5.3: Metrics as a function of threshold

5.4 Background Inclusion

The inclusion of the background as an extra class was evaluated, and for this purpose it is very important to understand both metrics presented in Section 4.2.1 because they might lead to evaluation mistakes if not correctly interpreted.

The comparison is made between two networks with VGG19 pretrained backbones, *softmax* activation in the output layer, applying class weights and using all three Dice, Jaccard and Focal losses, the only difference is whether the background is included or not.

In Figure 5.4 both predictions are depicted. For the case with background it can be seen that the latter is detected almost completely, but the tomato, the item we actually care about, is barely detected. On the other hand, when the background is not taken into account, the tomato is detected accurately. The detection of the grass in the background as salad should be ignored, it is not relevant for what is being intended to show.



(A) With background.

(B) Without background.

FIGURE 5.4: Network Predicted Output

The situation showed before happens consistently along the whole validation set, with the background being predicted accurately but not the other items, leading to the metric values shown in Table 5.1.

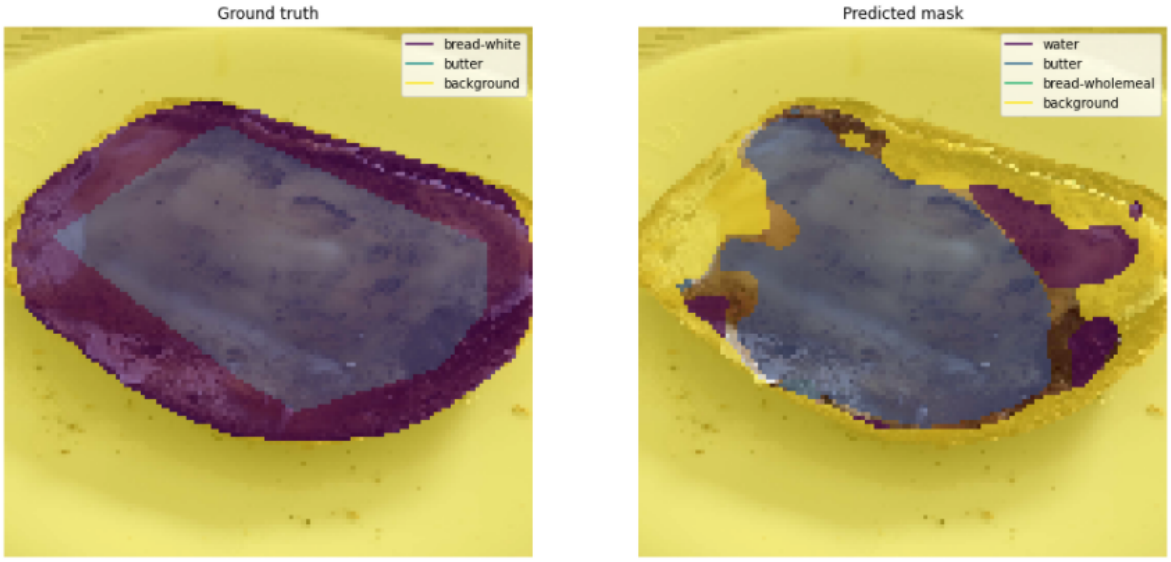
From the metrics it can be seen that both the IoU and the Only True IoU improve when removing the background, being the latter a more substantial change in relative terms. This is due to the fact that said improvement affects mostly the actual predictions of the food, a characteristic that is highlighted in the Only True IoU metric. On the other hand, since the background channel is not taken into account in any of the metrics, the classic IoU also improves, but when averaged against all the other empty channels, the improvement is not that significant.

Background	IoU	Only True IoU
Yes	0.797	0.035
No	0.838	0.062

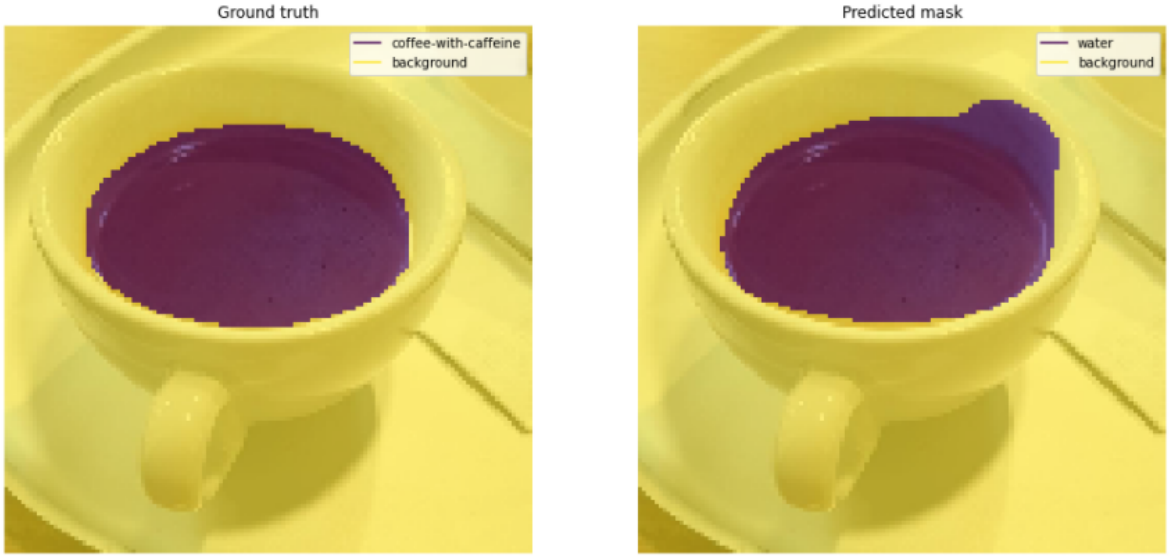
TABLE 5.1: Metric values including background or not

5.5 Backbones

The backbones used are the ones mentioned in Table 3.1. After several tests where the backbone was kept the same (VGG19), the possibility of changing the architecture of the encoder was explored. All the remaining variables were unchanged with respect to the best outcome obtained with VGG19, except for the activation function in the last layer (EfficientNetB5 and ResNet50). It is also worth mentioning that for the case of EfficientNetB5, the model used was authored by the library Segmentation Models [6] as a means of evaluation of the ones developed for this project.



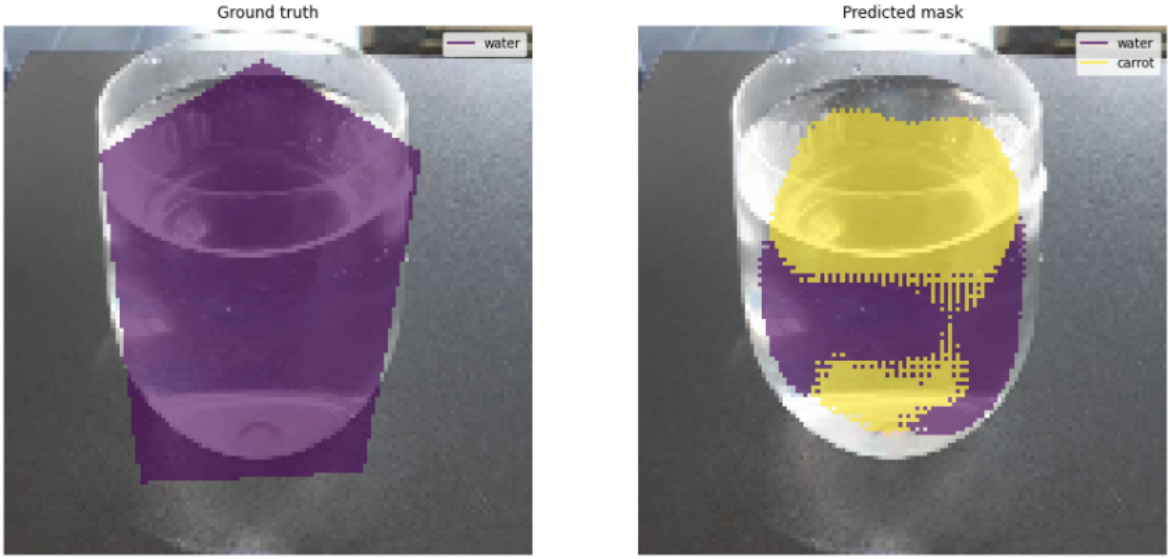
(A) Wrong prediction in complex scenario.



(B) Acceptable segmentation yet wrong classification.

FIGURE 5.5: Results obtained with EfficientNetB5 backbone.

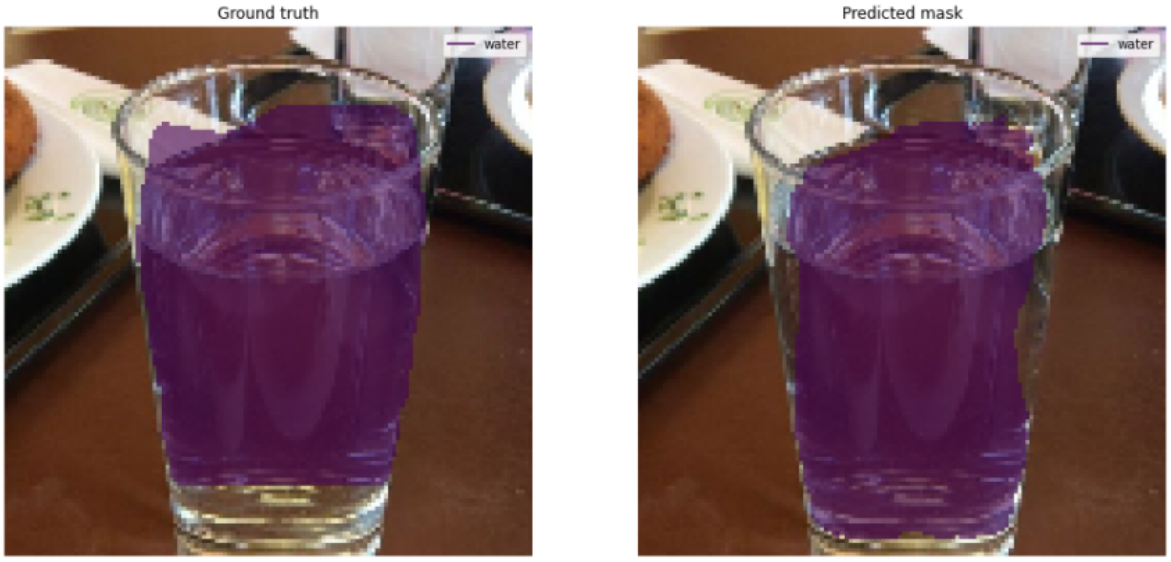
(A) Incorrect classification and incomplete segmentation.



(B) Incorrect prediction. Squared pattern noticeably visible.

FIGURE 5.6: Results obtained with ResNet50 backbone.

(A) Correct classification and segmentation. Prediction even adds a detection of a type of bread that was not in the ground-truth.



(B) Correct segmentation and classification.

FIGURE 5.7: Results obtained with VGG16 backbone.

Backbone	Activation	IoU	Only True IoU
EfficientNetB5	Softmax	0.7905	0.0345
EfficientNetB5	Sigmoid	0.8005	0.0529
ResNet50	Softmax	0.8876	0.0191
ResNet50	Sigmoid	0.8096	0.0152
VGG16	Softmax	0.8579	0.0555
VGG19	Softmax	0.8381	0.0616

TABLE 5.2: Metric values for different backbones

As it can be appreciated in Figures 5.5 and 5.6, and more objectively in Table 5.2, the extraction of features via EfficientNetB5 or ResNet50 did not offer improvements over VGG19, even though with a *sigmoid* activation function, EfficientNetB5 came reasonably

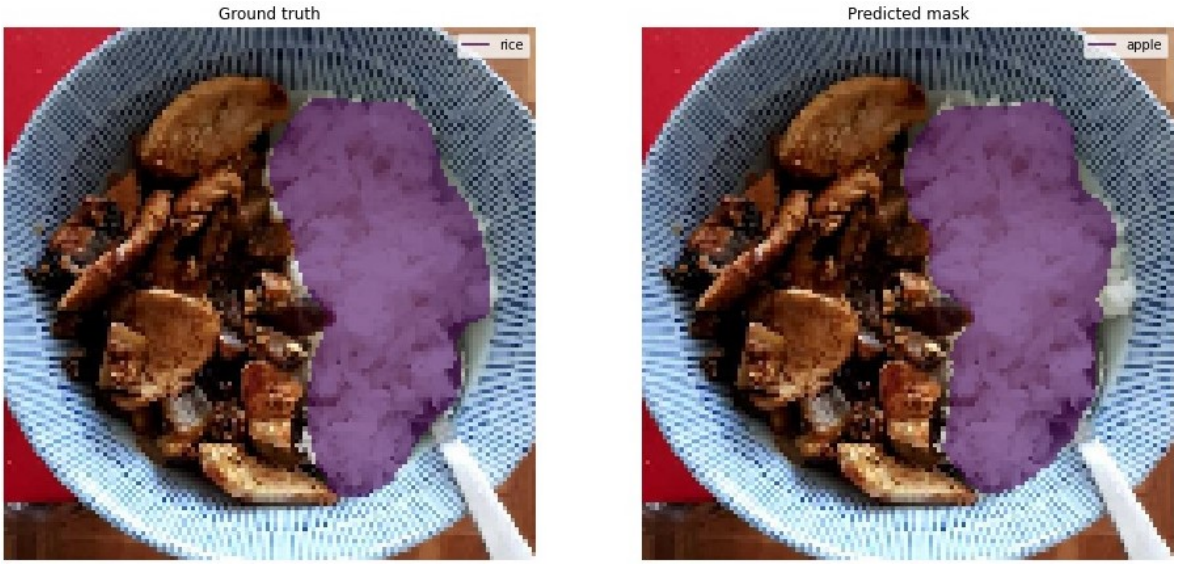
close. It is also worth noting the segmentation areas provided by the ResNet50 model, showing a somewhat squared pattern in the edges, probably due to a different set of features extracted.

Finally, with a more similar architecture, VGG16 offers a comparable performance to that of VGG19.

5.6 Channel Correction

A consistent problem found when evaluating the results was that some categories were consistently swapped, like 'apple' and 'rice', shown in Figure 5.8, also 'bananas' with 'cucumbers', 'carrots' with 'coffee-with-caffeine' and 'egg' with 'mixed-vegetables'. This happened for all the backbones used, the example shown uses a VGG19 backbone with softmax activation, without a background channel and a threshold of 0.95.

(A) Detected rice instead of apple



(B) Detected apple instead of rice

FIGURE 5.8: Channel Confusion

It can be considered cheating, but if the channels are swapped after the prediction, the calculated metrics improve considerably, as shown in Table 5.3, for the case of the same network mentioned above. With this final correction the best metrics are achieved.

Correction	IoU	Only True IoU
Before	0.838	0.062
After	0.873	0.107

TABLE 5.3: Metric values before and after channel correction

5.7 Results Summary

The most important results are shown in Table 5.4, in decreasing order of 'Only True IoU'. As mentioned above, the best performance is obtained when doing the channel correction.

BACKBONE	ACT.	LOSS	CLASS W.	BKG	CH. CORRECT.	SIZE	THRESHOLD	IoU	O. True IoU
VGG19	Sofmax	D-J-F	No	No	Yes	128	0.95	0.873	0.107
VGG19	Sofmax	D-J-F	Yes	No	Yes	128	0.9	0.858	0.097
VGG16	Softmax	D-J-F	Yes	No	Yes	128	0.5	0.877	0.095
VGG19	Sofmax	D-J-F	Yes	No	No	128	0.9	0.838	0.062
VGG19	Softmax	D-J-F	No	No	No	128	0.9	0.825	0.060
VGG16	Softmax	D-J-F	Yes	No	No	128	0.5	0.858	0.055
EfficientNet B5	Sigmoid	D-F	No	Yes	No	128	0.7	0.800	0.053
VGG19	Softmax	D-J-F	No	No	No	256	0.35	0.780	0.042
VGG19	Softmax	D-J-F	Yes	Yes	No	128	0.5	0.797	0.035
EfficientNet B5	Softmax	D-F	No	Yes	No	128	0.5	0.791	0.034
VGG19	Softmax	D-J-F	Yes	Yes	No	128	0.7	0.859	0.024
VGG19	Softmax	D-F	No	Yes	No	128	0.35	0.646	0.019
ResNet50	Softmax	D-J-F	Yes	No	No	128	0.35	0.888	0.019
ResNet50	Sigmoid	D-J-F	Yes	No	No	128	0.35	0.810	0.015

TABLE 5.4: Result of the variation of the network parameters

6 Conclusions

As a final conclusion for this report, the valuation of the results obtained are fairly positive. The original task of providing a Deep Neural Network for image segmentation of food samples was fulfilled to a good extent, and the outcome is a model whose predictions visually make sense, and has the metrics to back it up. However it must be said that such model has room for improvements, as it struggles when the samples increase in complexity (namely a plate of many foods all together, or situations that require a deeper understanding, like a pealed bananas or an apple cut in pieces). For this improvements, a better annotated dataset would also be of help, as some ground-truth errors or inaccuracy tend to confuse the model while training.

7 Bibliography

References

[1] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[2] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV].

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[4] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[5] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].

[6] Pavel Yakubovskiy. *Segmentation Models*. https://github.com/qubvel/segmentation_models. 2019.