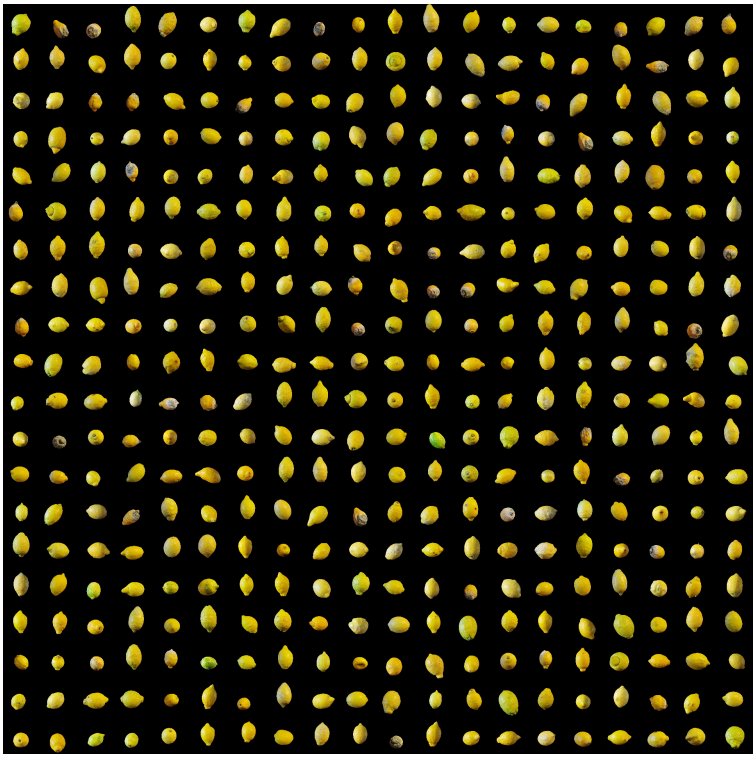
Product quality control Project

Riccardo Fava - riccardo.fava6@studio.unibo.it

Project work - Machine Learning for Computer Vision

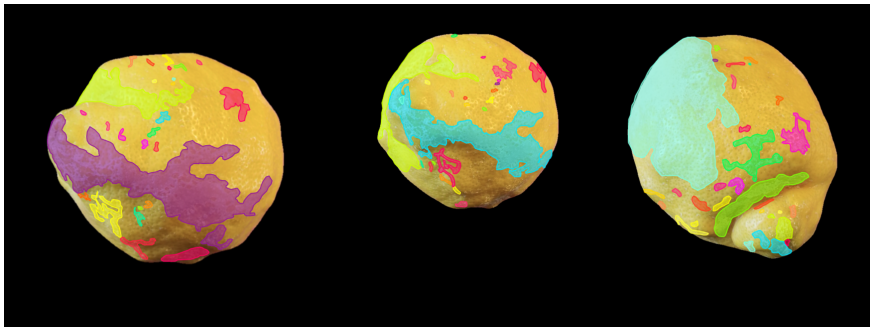


1 Abstract

In this work, we have tried to address the product quality control problem, by implementing a neural network for performing the semantic segmentation of images containing lemons. In order to have performances near to real-time, for this particular task, we decided to use the SFNet model on top of a lightweight ResNet-18 backbone. We experimented with different configurations of this architecture, applying simple modifications to the model, in order to increase the performance. In the end, we've been able to archive around 39% of mIoU with our best configuration.

2 Task Description

Our task consists in performing the semantic segmentation of images depicting lemons and applying quality control through the inspection of the products, in order to find possible defects. In particular, the model has to map an RGB image (H×W×3) to a semantic map (H×W×C) with the same spatial resolution H×W, where C is the the number of predefined semantic categories.



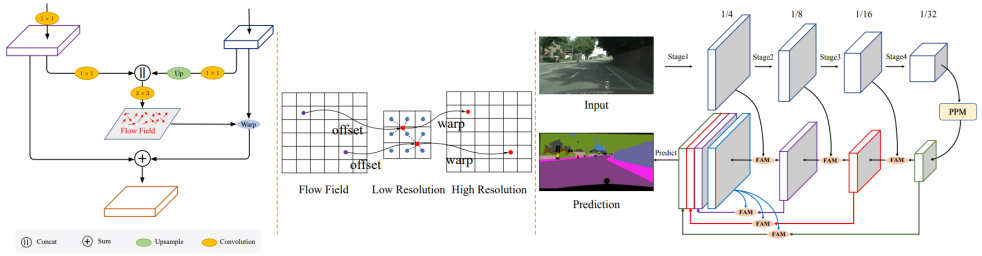
We decided to rely on the Lemons quality control dataset[1], which is composed of 2690 images representing lemons, annotated with all the specific types of defects that the product can have. In particular, filtering the classes based on the ones that are related to our use case, we ended up with the following six categories to identify within our images:

- **Illness**: disease affecting the lemon.

- **Gangrene**: localized death and decomposition of body of the lemon.
- **Mould**: green and white fungus growing on the surface of the lemon.
- **Blemish**: a small mark or flaw on top of the lemon.
- **Dark_style_remains**: After pollination, the remains of style are preserved in the fruit. A dark area around the remain of style indicates an unhealthy fruit. This place is the region from which the fruit starts rotting or catches mould.
- **Pedicel**: structure connecting a single flower to its inflorescence

3 Model Used

With the purpose of implementing a model that has real-time performances, we decided to use the SFNet[2] with a ResNet-18[3] backbone, an architecture that has been designed for accurate and fast scene parsing. The main peculiarity and innovation of this network is the Flow Alignment Module (FAM) to learn Semantic Flow between feature maps of adjacent levels, and broadcast high-level features to high-resolution features effectively and efficiently. Basically, the transformed high-resolution feature map and low-resolution feature map are combined to generate the semantic flow field, which is utilized to warp the low-resolution feature map to a high-resolution feature map. During the warping procedure, the value of the high-resolution feature map is the bilinear interpolation of the neighboring pixels in a low-resolution feature map, where the neighborhoods are defined according to the learned semantic flow field.
More in general instead, the network that we developed contains a bottom-up pathway as Encoder and a top-down pathway as Decoder. The Encoder is composed of four residual stages of the ResNet-18 and of a Pyramid Pooling Module (PPM)[4]. The Decoder instead can be seen as a Feature Pyramid network (FPN)[5] equipped with several FAMs.



(a) Flow Alignment Module **(b) Warp Procedure** **(c) Overview of SFNet**

4 Experiments

We decided to rely on the Pytorch framework for the implementation of the architecture and on the Google Colaboratory platform for training the model. All the experiments were carried out using more or less the same settings: a batch size of 16, the Cross-Entropy as loss function, the Pytorch StepLR as learning rate scheduler, and an early stopping procedure based on the loss of the test set. Data augmentation contains horizontal and vertical flips, random rotation, and cropping with random size. For what concerns the optimizer we tried both Adam and stochastic gradient descent (SGD) with different values of the learning rate. Together with the combination of the previously described hyperparameters, we decided also to perform some experiments applying a few changes to the model. In particular we tried to execute a sort of ablation study, substituting the PPM module with a simple 1x1 convolution and/or removing the use of the FAM modules (used before the last concatenation) in favor to a simple bilinear upsampling operation. All the combinations of these options have been explored during the hyperparameters tuning phase, executed based on the validation set. For quantitative evaluation, it has been used the mean of class-wise intersection-over-union (mIoU) metric. In the end, we obtained the best results using the Adam optimizer with a learning rate of 0.0001.

5 Results

Training our neural network for about 40 epochs (or less, in the case of early stopping) on the best configuration of the hyperparameters that we found during the tuning phase, we obtained a mIoU of 0.39 .
Considering how well this specific model performed on other benchmarks, the results are not very satisfying. Regarding instead the speed performances, the model reached results that are around 17 FPS measured on a Google Colab GPU (Tesla T4).

6 Error Analysis

We displayed the model's predictions in order to more thoroughly assess the mistakes it is producing. The first consideration that we can do is that the model has performances that are strictly related to the composition of the classes within the training set. Analyzing in fact the per-class Intersection over Union metric, we can notice a great discrepancy between the value of the most popular class (*mould* with 0.81) and the value of the less popular (*dark_style_remains* with 0.01). All the other classes have instead a value of the IoU that is between 0.3 and 0.5.
Another consideration that we can do is that, by looking at the images, it is difficult to find the difference between some classes that result to be very similar, even by a human observer. Most of the mistakes that our model performs are effectively executed confusing similar classes like for example *mould* and *blemish*.
In the end, by observing some images with the ground truth mask, we noticed that the majority of the defects on lemons are very tiny; in this kind of situation our model has a lot of difficulties in detecting the imperfections.

7 Conclusions and future works

As said before, the results in terms of mIoU turned out to be not so exciting if we make a comparison with the performances that the developers of this architecture achieved on other datasets. However, analyzing the results, we have to keep in consideration the dimension of the dataset that we used and the relative difficulty of the task. In particular, as we observed in the previous section, the performance of the model on some classes is worse mainly due to the limited number of examples present in the

training set. Increasing the dimension of the dataset would definitely allow us to increment the results that we obtained. In fact, if we analyze the performance that the model obtained on the most populous class, we can consider ourselves relatively satisfied.
In the end, possible extensions of this work could be:

- Performing a more accurate hyperparameters tuning and trying more complex training procedures.
- Trying with a broader spectrum of architectures and with variants of the ones we used, like for example experimenting with other backbones.
- Extending the dimension of the dataset or pre-training the model on other sets of data collected for a similar task.
- Introducing and testing methods for better handling the problem of class imbalance.

References

[1] Lemons quality control dataset
https://github.com/softwaremill/lemon-dataset

[2] Semantic Flow for Fast and Accurate Scene Parsing (2021)
Xiangtai Li, Ansheng You, Zhen Zhu, Houlong Zhao, Maoke Yang, Kuiyuan Yang, Shaohua Tan and Yunhai Tong.

[3] Deep residual learning for image recognition. (2016)
He, K., Zhang, X., Ren, S., Sun, J.

[4] Pyramid scene parsing network. (2017)
Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.

[5] Feature Pyramid networks for object detection. (2017)
Lin, T.Y., Doll´ar, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.