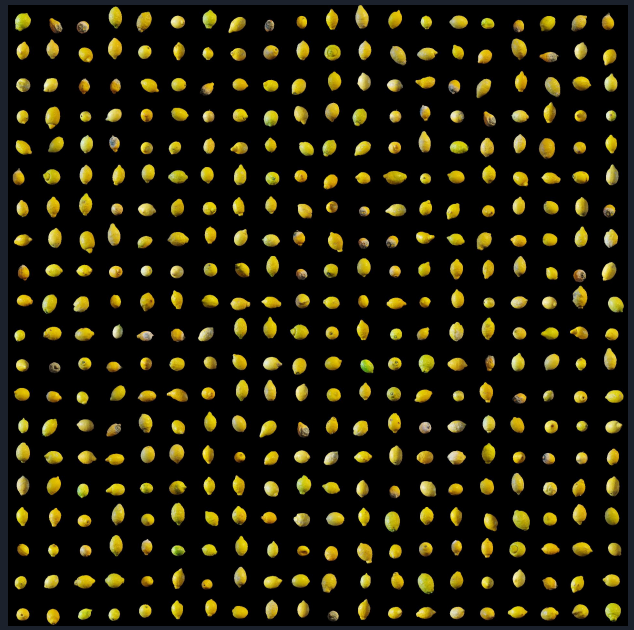
Lemons quality control project

Task Description

Product quality control is an exploration carried out to determine if they adequately comply with the expected characteristics to reach a final consumer.
In industry scenarios, this task can be performed using several different techniques. In this project we decided to approach this problem using computer vision algorithms.

Dataset

- We decided to rely on the Lemons quality control dataset, which is composed of 2690 images representing lemons, annotated with all the specific types of defects that the product can have.
- This dataset has annotations both for object detection and for semantic segmentation.
- Thinking about our use case we decided to use a semantic segmentation approach, believing that a semantic map of the defects of the lemon could be more informative and more flexible with respect to bounding boxes.



Classes

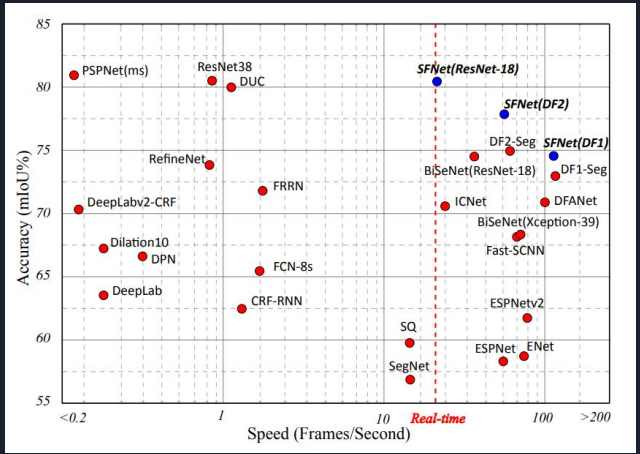
1. **Illness**: disease affecting the lemon.
2. **Gangrene**: localized death and decomposition of body of the lemon.
3. **Mould**: green and white fungus growing on the surface of the lemon.
4. **Blemish**: a small mark or flaw on top of the lemon.
5. **Dark style remains:** After pollination, the remains 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.
6. **Pedicel:** structure connecting a single flower to its inflorescence

Choice of the model architecture

Thinking about a possible scenario in which the product quality control system have to perform its predictions, for example analyzing lemons that are passing on a conveyor belt, we searched for semantic segmentation architectures with real-time performances.

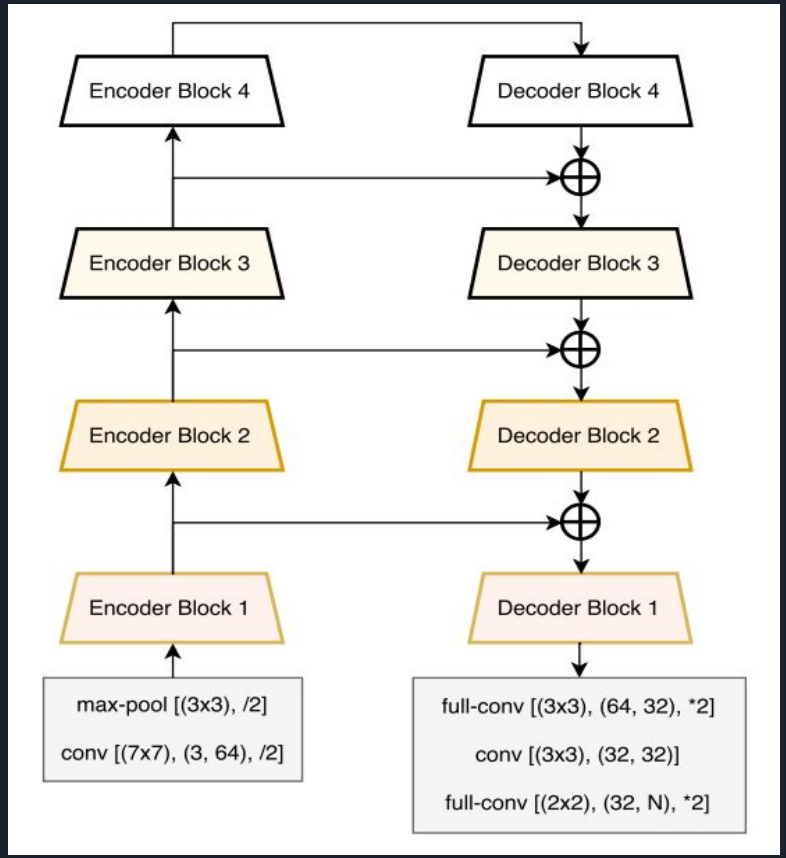
Consulting the state-of-the-art in this field we decided to implement the following models:

- LinkNet with ResNet-18 backbone (BASELINE)
- SFNet with ResNet-18 backbone



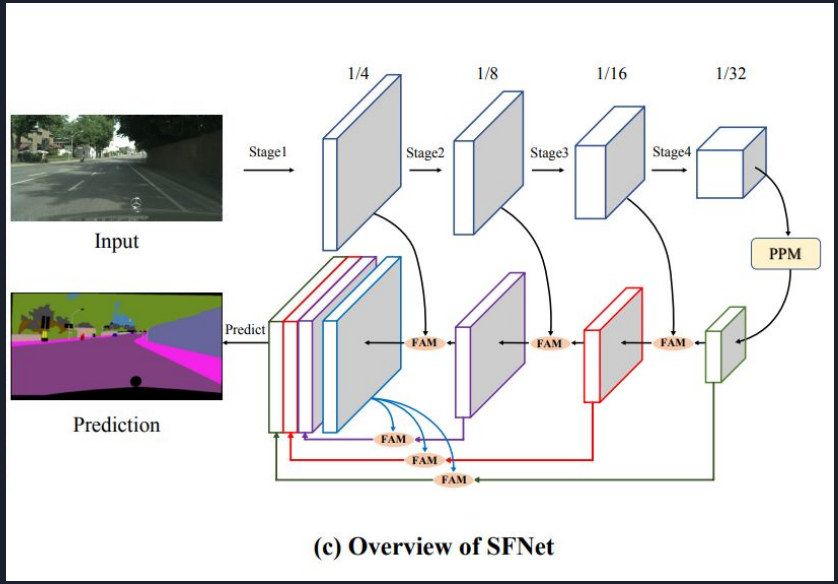
LinkNet model

- Efficient Encoder-Decoder architecture developed in 2017.
- Archived 76.4% of mIoU on Cityscapes benchmark
- The Encoder captures the context and extract features, while the decoder is used to recover spatial localization. Skip connections are used to transfer informations from Encoder and Decoder blocks, increasing accuracy
- Implemented with ResNet-18 (pretrained on ImageNet) blocks in the Encoder.



SFNet model

- More complex Encoder-Decoder architecture developed in 2021
- Archived 80.4% of mIoU on Cityscapes benchmark
- The Encoder is composed of four residual stages of the ResNet-18 and of a Pyramid Pooling Module (PPM).
- The Decoder instead can be seen as a Feature Pyramid network (FPN) equipped with several Flow Alignment Module (FAM).



Experimentation

- Pytorch framework
- Google Colab Platform for training the models with GPU

- Data augmentation:
  - horizontal and vertical flips
  - random rotation
  - cropping with random size

- Cross-Entropy loss function
- Class weights for dealing with class imbalance
- LR Scheduler to module the LR parameter
- Early stopping procedure to avoid overfitting

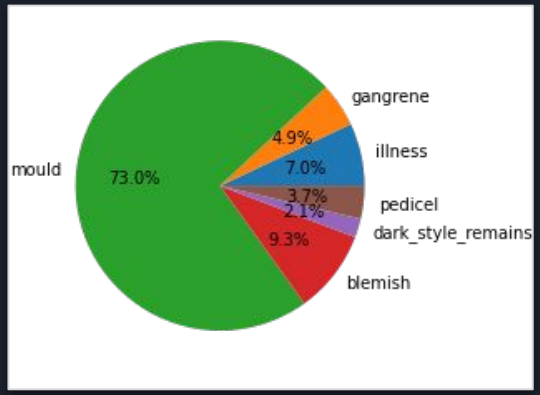
- Hyperparameters tuning:
  - Optimizer (Adam and SGD)
  - Batch size
  - Learning Rate

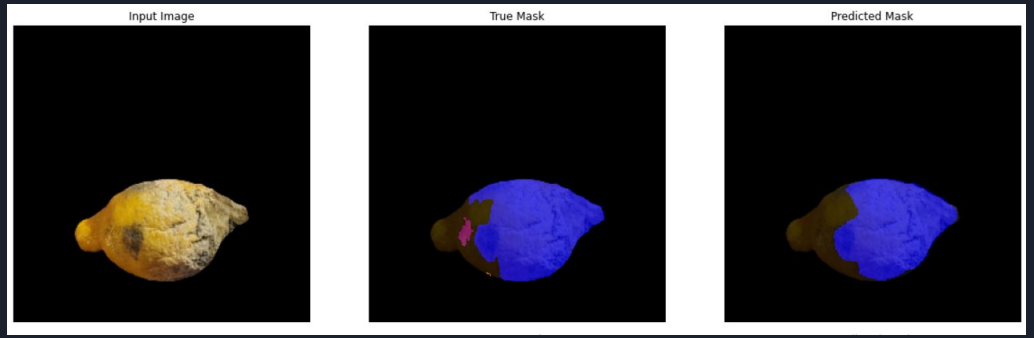
Results

Model	Mean IoU
LinkNet	13.0%
LinkNet + class weights	16.9%
SFNet	34.8%
SFNet + class weights	39,1%

Error Analysis

- The model has performances that are strictly related to the composition of the classes within the training set.
- 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 0.2 and 0.5.

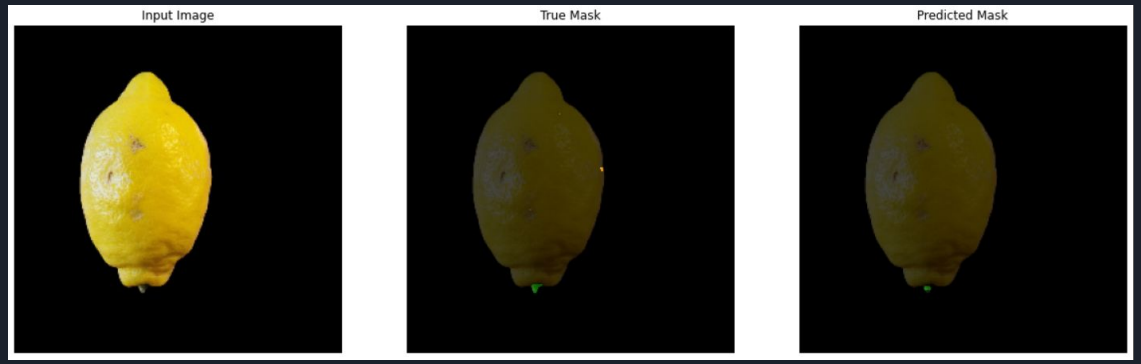




Error analysis

- 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



Conclusions

- 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 these architectures achieved on other benchmarks.
- 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.

Future works

In the end, possible extensions of this work could be:

- ➢ Performing a more accurate hyperparameters tuning and trying more complex training procedures.
- ➢ Introducing and testing more powerful methods for better handling the problem of class imbalance.
- ➢ 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.