Nasri Chiheb

UNIVERSITAT ROVIRA I VIRGILI

AVPR Lab: Lab project

Crack segmentation using deep learning

Introduction:

The purpose of this lab project is to perform a semantic segmentation approach for crack segmentation. The algorithm is implemented with MATLAB using deep learning toolbox library. The presented solution uses a U-Net. The algorithm follows four steps preprocessing, creation of the network, training, and testing. We will conclude by presenting the performances of the algorithm.

Method description:

- **Pre-processing :** In this part we will work on the image to prepare it for the training phase
  - Here pre-processing consists of resizing the input images for two purposes. First the U-Net (and most of the cnn's) needs images of the same size. Second reason is for computation problems, indeed working with high resolution images is resource consuming. Here we work with images of size 64x64x3
  - We also create different datastores to make easier the work with big databases and to assign classes to pixels

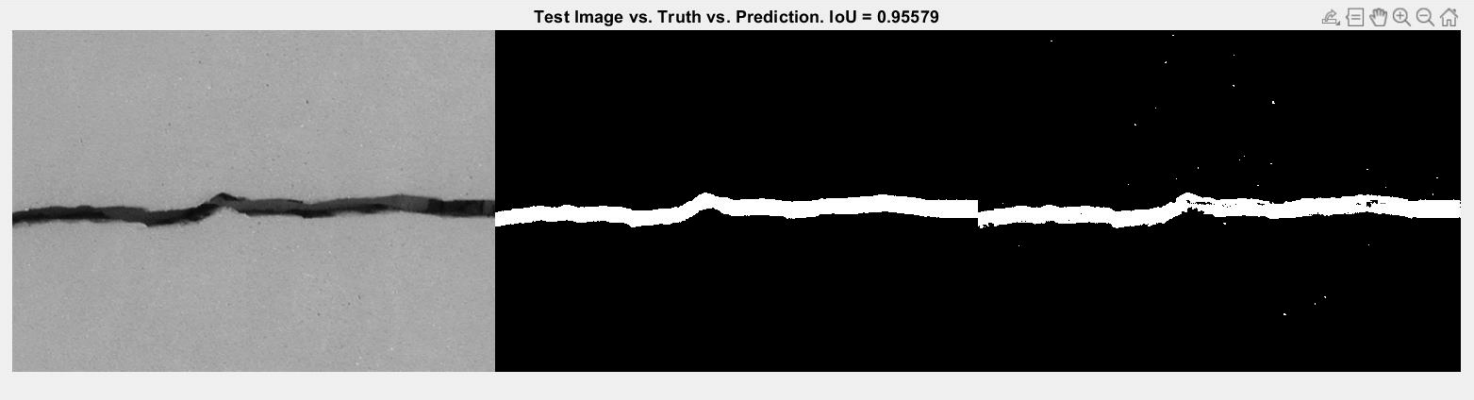
- **Creation of the network :** We will use a U-Net and modify it for the purpose of the task.
  - The creation of network is quite easy as it is already implemented, we only have to set the size of the images, number of classes and the depth of the network. But once again it is resource consuming so we will use default parameters with a depth of 4
  - By visualizing our data, we notice that our class proportion is imbalanced (we have less than 1% of the pixels that represent cracks). This can lead to a problem where the overall accuracy of the model is high, but the crack detection is weak. To deal with this situation we have added weighted classes to the pixel classification layer.

```
classFrequency =

    0.0092
    0.9908
```

```
classWeights =

  108.4923
    1.0093
```

- **Training :** In this part we train the network.
  - Training the network is easy. The most important point of this phase is to choose the right parameters. Indeed, this kind of problems tends often to overfitting due to class proportion. In this way the parameters have chosen regarding this problem. The network uses the stochastic gradient descent with momentum. A constant learning rate and a shuffle in every epoch. It also uses l2 regularization to avoid overfitting. More details about the parameters can be found in the commented code
  - At the end of the training we can save the network for further utilization as it takes time to train.

- **Test :** Here we will use the trained network with the testing data. Note that you can use pre trained networks more details in the commented code
  - First, we load the data, and we create the datastores as we did in first step
  - We perform the testing using the "semanticseg" function furnished with MATLAB.



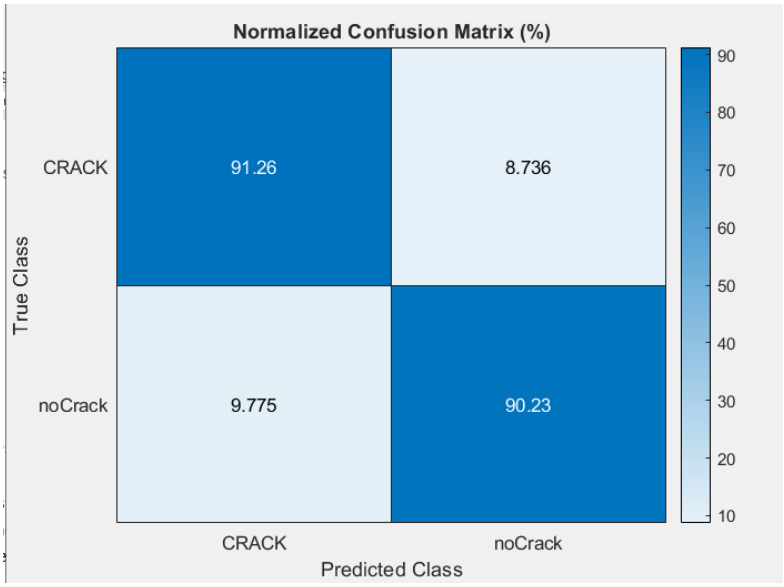
Example of prediction compared to real image

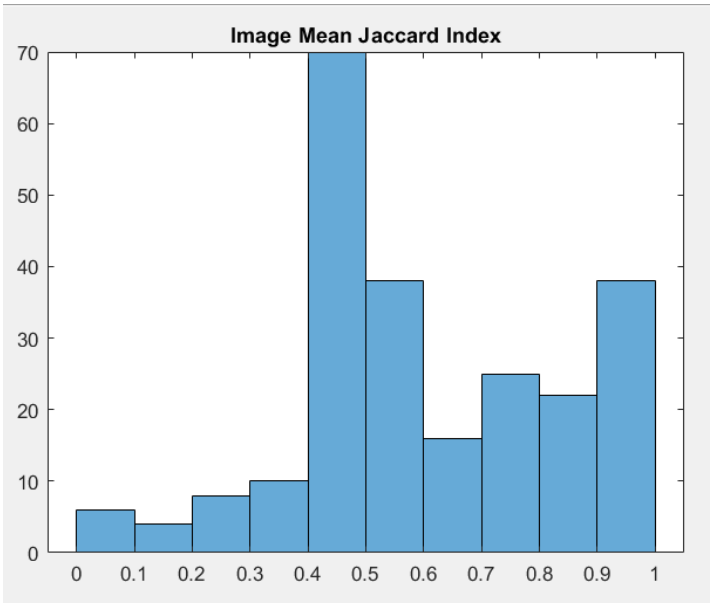
Results:

This section is dedicated for results discussion. An important point before commenting the results is that this type of problem takes a lot of time to compute, between 10 and 40 minutes to train a network depending on parameters. I have tried to optimize the network but for technical limitation the results can be improved.

These results are from net1.m and net2.m two pre trained networks furnished with the code.

Net 1 :

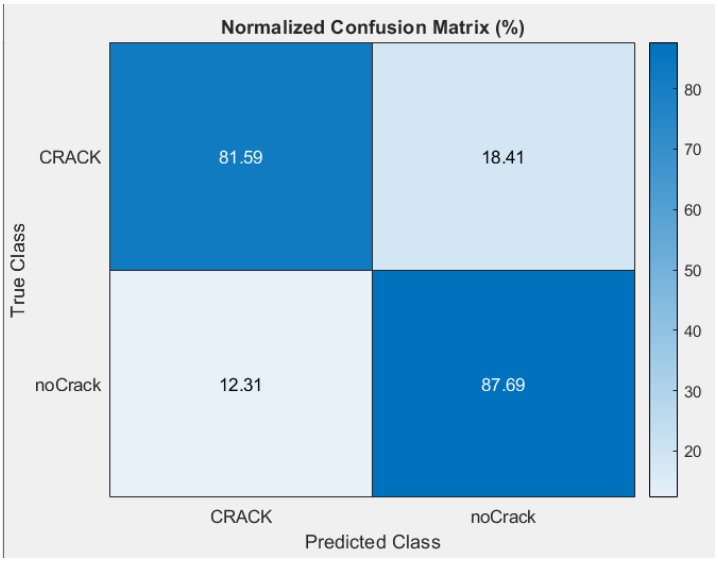


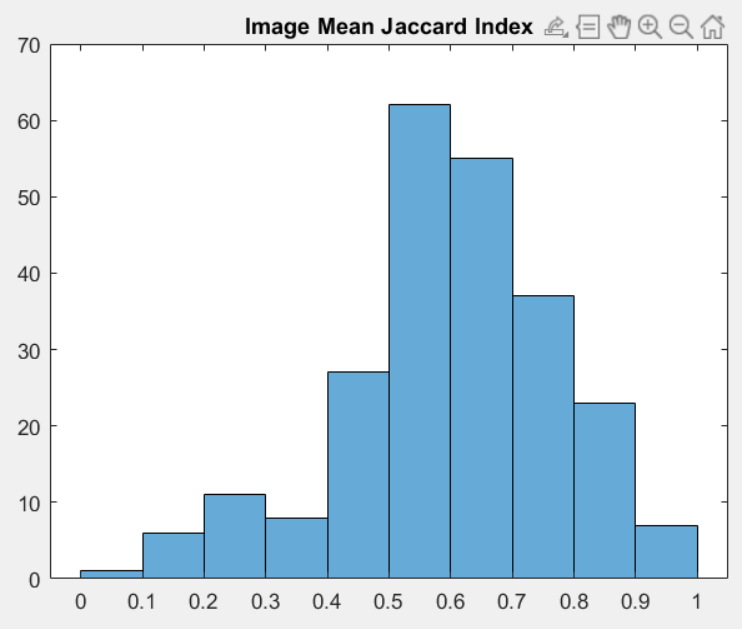


	Accuracy	IoU	MeanBFScore
CRACK	0.91264	0.14101	0.45517
noCrack	0.90225	0.90085	0.76464

GlobalAccuracy	MeanAccuracy	MeanIoU	WeightedIoU	MeanBFScore
0.90244	0.90745	0.52093	0.88751	0.63465

Net 2 :





	Accuracy	IoU	MeanBFScore
CRACK	0.81593	0.21923	0.37989
noCrack	0.8769	0.86966	0.46053

GlobalAccuracy	MeanAccuracy	MeanIoU	WeightedIoU	MeanBFScore
0.87426	0.84642	0.54445	0.84152	0.42021

Here IoU (Intersection over Union) is used for the Jaccard index. BF Score is also known as F1 score.

Here we will compare two trained networks that shows the difficulty of this problem. One has high accuracy and F1 score but lower Jaccard Index and net 2 has the opposite.

For net 1 : As we can see the results are different depending on the used metric. As explained before due to label balancing the global accuracy is high but the other metrics are low. Overall mean Jaccard index is greater the 0.5 which is a good score, but it is low for detecting crack pixels even if it tends more than 0.5. False positive is less than 10% for crack on no crack pixels which is a good score.

For net 2 : For net we have overall lower results with a lot of false positives 18% for crack and 12% for no crack pixels. But here the algorithm is more accurate when locating the pixels. This shows the importance of the metrics to use and depending on the goal of algorithm as every metric indicates a different scope for the problem. I have also tried other networks but due to lack of time I couldn't improve the results.

To conclude, pixel classification is a hard task as it is easy to trick the model. I have tried multiple solutions to improve the score (by adding weighted classes or varying the training parameters) but working with images need high performance machine. Using this type of classification was a good experience as it is the first time where I use different metrics rather than most common ones.