

DEEP LEARNING (DL) ASSIGNMENT:   
FV Cells – defects detection

**Master in Robotics and Control System**

*Team*

Wojciech Kirejew

Gorka Holguin

# INTRODUCTION

In the recent years there were dynamic development of various Deep Learning applications. One of them is inspection of manufactured goods in search for defects and therefore defected products. This application is rather broad, as designing appropriate solutions one can encounter various challenges and therefore different approaches can be the most optimal in different cases. The description of Photovoltaic Cells inspection case and discussion on possible solutions with it’s advantages and disadvantages is covered in the next chapters.

The purpose of this study is therefore to identify challenges that are present in the chosen case, discuss possible solutions and choose most promising ones and thereafter make an implementation of this solution/s in this real world scenario.

What is intended in this practice is to find a method through Deep Learning that facilitates the inspection of solar cells in their production process. For this, photographs of different cells will be analysed, they will be classified as "good" or "bad" depending on whether they have defects or not, and the defects of the bad cells and their magnitude will be identified.

As for the Deep Learning model to be used, there will be a database with photographs of "good" and "bad" cells that will be used to carry out the training.

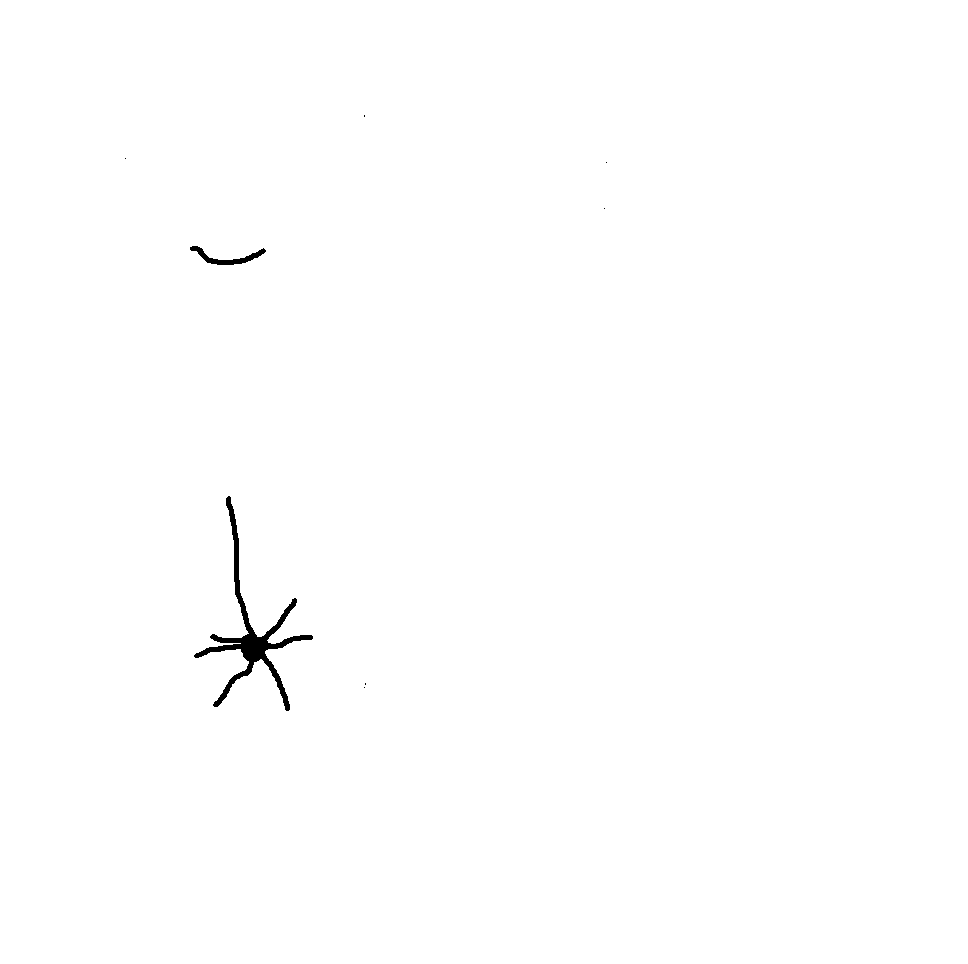
# BRIEF LOOK AT OUR CASE DATABASE AND CONSIDERED CHALLENGES

The chosen case – FV Cells defect detection – consists of a database of “.bmp” images of FV Cells. These images have different sizes of around 950x950 px and are split into two groups – one for undefected cells and another for defected cells. The second group (defected cells) consists of images of cells and corresponding black-white images of the defects itself (called Region of Interest) that are of exact same size as the images of cells. The examples of those images are shown beyond.

Obraz zawierający ściana, meble

Opis wygenerowany automatycznie 

*Figure 1. Undefected FV Cell. Figure 2. Defected FV Cell.*



*Figure 3. ROI of defected FV Cell.*

The dataset consists of only a small number of images – around 150 of each type and therefore it is the first obvious challenge for our DL solution to this classification problem. Another challenge is that the images are of different sizes and that their size is very big in general. This big size of the images results in big RAM usage when DL models try to process entire images.

After initial work and testing the dataset with prepared models, there arose another challenge that is the unhomogeneity of undefected parts of cells images. Even at the sample images that are shown above, one can easily notice the darker spots, that don’t correspond to the defects, but makes training way harder (as our work shows in later chapters) or even impossible, so there should be used some image preprocessing that allows to eliminate this very unhelpful attribute. There is also one more challenge, namely the difference in the brightness between images. This property shouldn’t be of much concern for some models, but it makes training harder in general and taking into account the very little number of training examples it complicates the task and adds to its challenges.

1. **GENERAL DEFECT DETECTION SOLUTIONS WITH DEEP LEARNING**

In general to perform defect detection of images databases one can choose many various Deep Learning approaches. The simplest solution is to use a simple (or actually as advanced as one wants to fulfill her goals) Convolutional Neural Network, that processes entire images and by binary classification asses images to good or bad class (binary classification). Another simple approach is to use the Autoencoder model and train it to retrieve the good examples and afterwards take an anomaly detection approach to find defected samples. This solution will be further discussed in this chapter and results of this approach will be presented as well.

Those simple solutions are however limited by the amount of training data it requires for a proper learning process and to achieve good enough accuracy, requires far more data than we can provide in this case. To make these solutions work better there is possible to perform image segmentation or other data pre-processing/augmentation that helps resolve these problems, or makes them less impactful. This data segmentation will be discussed in more detail in further part of this chapter.

The more complex solution that can be applied to defect detection and does not require extended data processing is the usage of the U-NET model for segmentation. Details of this approach will also be discussed in a later part of this chapter providing advantages and results of this type of Convolutional Network.

# Autoencoder approach

An Unsupervised Autoencoder will be used to detect these defects. It will be trained only with images of good cells. The aim is to train the model so that it can reconstruct the good images as similar as possible to the previous images and, in turn, be unable to correctly reconstruct the images that contain some type of defect. That is, compare the original images with the images reconstructed by the autoencoder and see how the images that contain good cells are practically the same, while there will be differences with the images with bad cells.

Regarding the Autoencoder, a convolutional layer with a RELU activation function will be applied, in order to induce non-linearities. After this, another convolutional layer with fewer feature maps than the previous one will be used. Afterwards, a Max Pooling layer will be applied after which all the values in a vector will be reordered with Flattering. The matrices will be obtained by undoing the Flattering using the Decoder and the original images will be reconstructed by applying another convolutional layer with RELU activation.

Finally, as mentioned above, the similarities of the original images with the reconstructed ones will be analyzed to detect the differences and thus be able to detect possible anomalies. In what corresponds to the processing, say that the smaller the size of the segmented images and the greater the number of segments made with each original image, the defects can be identified with greater precision. This would increase the computational cost of training, so an ideal relationship between precision and said computational cost will be sought.

Training the model with full images does not make much sense, as it has only around 200 images to train which is way less than required for this type of model. However this solution was tested and the results as expected are not good and shown on Figure 4 below.

# Images segmentation

Image segmentation is a computer vision task that segments an image into multiple areas by assigning a label to every pixel of the image. It provides much more information about an image than object detection, which draws a bounding box around the detected object, or image classification, which assigns a label to the object.

Segmentation is useful and can be used in real-world applications such as medical imaging, clothes segmentation, flooding maps, self-driving cars, etc.

There are two types of image segmentation:

* Semantic segmentation: classify each pixel with a label.
* Instance segmentation: classify each pixel and differentiate each object instance.

It will begin by segmenting the photographs into reduced images of 64x64 pixels to analyze each cell as if they were more than one and thus, by means of a cumulative, detect the pixels of the original images that contain some defect.

The segments of the images will be analyzed to determine if it corresponds to a good or bad segment and thus classify the cells (good or bad) and if they are bad, to be able to detect in which area of the cell the defect is located.

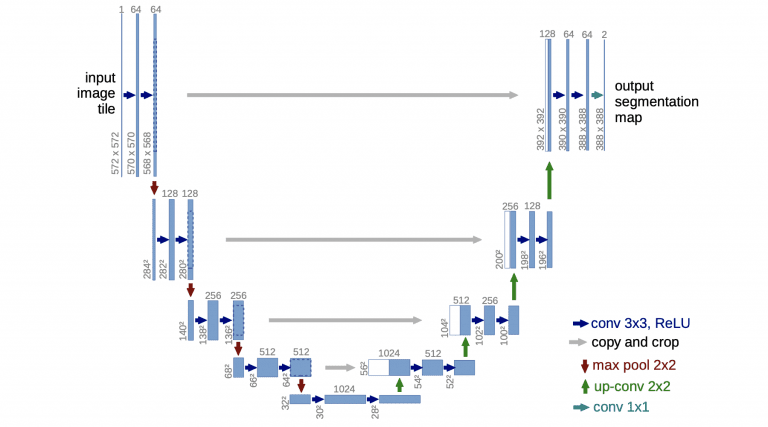
The model trained with good examples created this way and saved in the folder “segmentation/train/good” should train way better, as the number of training examples is more than 10 times higher.

# U-NET Architecture

U-Net is a semantic segmentation technique originally proposed for medical imaging segmentation. It’s one of the earlier deep learning segmentation models, and the U-Net architecture is also used in many GAN variants such as the Pix2Pix generator.

U-Net was introduced in the paper, U-Net: Convolutional Networks for Biomedical Image Segmentation. The model architecture is fairly simple: an encoder (for downsampling) and a decoder (for upsampling) with skip connections. As Figure 8 shows, it shapes like the letter U hence the name U-Net.

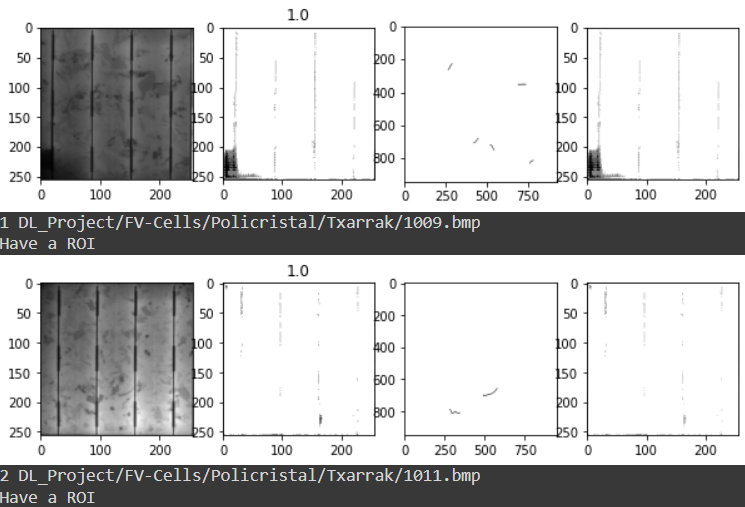
The gray arrows indicate the skip connections that concatenate the encoder feature map with the decoder, which helps the backward flow of gradients for improved training. [1]



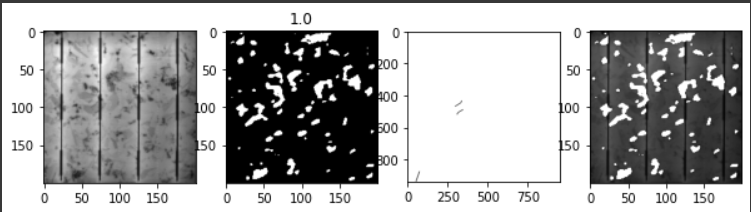
In our case we used three different UNET models (that are basically very similar, but come from different sources).

The first model comes from the U-NET Image segmentation tutorial[2] and is implemented with some adjustments, to change the output from 3 dimensional to binary, but therefore these changes could impact the quality of obtained results. The summary of this model is shown on the figure in the Colab Notebook, as the size of the image is too big to put it in this document and provide necessary visibility.

The second model is derived from the model for salt prediction based on seismic photos of the ground.[3] This model is working as a binary classification model and therefore is well suited for our case. The summary of this model is shown on the figure in the Colab Notebook, as the size of the image is too big to put it in this document and provide necessary visibility. In this model as an input we used images resized to 512x512 px and normalized <0;1>. The results however are far from ideal and some examples are shown on the Figure beyond.



Finally, the third model is derived from the Mono\_Cristal\_UNET\_100 provided in resources to the Deep Learning course. In this case we used a pretrained model and provided it input images of proper format and then visualized the results of its work. Despite the pretrain that was made with very similar images as in our dataset, the displayed results are far from ideal, what is shown in more detail in the Colab Notebook, but one example is also shown on the Figure beyond.



# SUMMARY OF THE WORK AND CONCLUSION

To sum up our work, we discussed and implemented various solutions to defect detection in the FV cells case, however we were not able to overcome all the challenges that this database and furthermore case provided. Anyways the results and the implementations of subsequent solutions are of some value. We can discuss reasons for the obtained results for subsequent solutions and propose ways to resolve these issues.

Moreover we understood the idea and the implementation of UNET networks for image segmentation, not only for binary classification but also for other implementations.

Unfortunately due to our lack of programming prowess we were not able to implement solutions that address detected issues, namely to preprocess the images of FV cells as that they are well suited for the Deep Learning Models.

To conclude our work, there are two main ways to obtain a DL solution to the defect detection case with a limited number of training examples. Simpler one, but more difficult in implementation is anomaly detection using an autoencoder with image segmentation. The other is to use a more complex DL Model - U-NET architectured CNN, that allows to train effectively even with a small number of examples and is easy to implement and is way faster that 1st solution.

# BIBLIOGRAPHY

[1] “U-Net: Convolutional Networks for Biomedical Image Segmentation”

Olaf Ronneberger, Philipp Fischer, Thomas Brox, arXiv:1505.04597, 2015

[2] <https://pyimagesearch.com/2022/02/21/u-net-image-segmentation-in-keras/>

[3] [https://towardsdatascience.com/understanding-semantic-segmentation-with-](https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4)

-[unet-6be4f42d4](https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4)b47