EPFL

CS-433

Machine Learning

Project 2

Detection and Segmentation of Brain Sections using Deep Neural Networks

---

Authors:
Kevin Pelletier
Eliott Joulot
Jelena Banjac

Professor:
Martin Jaggi

Lab Supervisor:
Thomas Templier

December 20, 2018

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

***Abstract*—We present an implementation of Mask R-CNN on dataset containing segmentation of ultra-thin sections of biological tissue in light microscopy imagery. The goal was to accurately segment each of the sections in a large image in order to determine the section coordinates. These coordinates will later be used for automated image acquisition in a high resolution microscope such as electron microscope. This Machine Learning course project at EPFL was done in collaboration with the Center for Interdisciplinary Electron Microscopy (CIME) at EPFL. We applied our knowledge to solve a practical real-world problem that involved the implementation of complex machine learning models for image segmentation and feature detection. Code has been made available at: https://github.com/BrainSegmentation where tissue-parts-detection [1] contains the notebooks and augmented data, and JekeelMaskRCNN [2] contains the files used for training the models.**

I. Introduction

Observing all neurons, their fibers and synapses in a small volume of brain tissue can only be performed with electron microscopy in volumes typically smaller than a tenth of a millimeter cube. To deal with this problem, Thomas Templier, the scientist from CIME lab, uses images of extremely fine cuts of the brain, and develops an automated imaging pipeline for the acquisition and assembly of large electron microscopy datasets.

To collect these images, the pipeline implemented requires the precise location of brain sections in a global image that can contain several hundred of these sections.

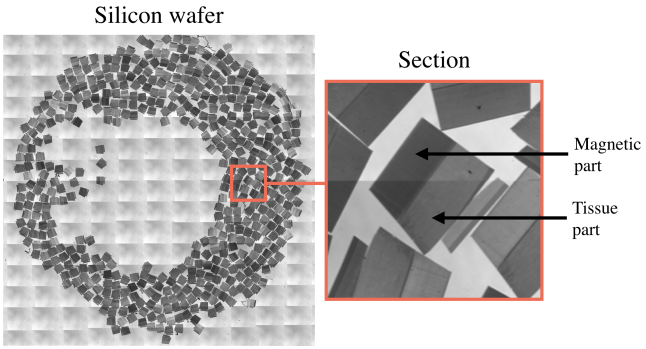

Figure 1: Brain section inside a silicon wafer

The main image provided is a silicone wafer on which the sections are positioned. Our objective was therefore to recognize and locate each of these sections precisely.

II. Data exploration

A. Datasets

Our data for this project were images of silicon wafers, for which the positions of the sections inside were preliminarily collected. The first particularity of our dataset is therefore the fact that we did not have a large number of different images, but that each of them contains a lot of different features. We started the project with 3 different wafers fully labelled.

Then, a second specificity is the fact that, although the images are totally different because the sections are (more or less) randomly located, the sections are very similar and there is in fact not so much difference from one section to another.

B. Features

As shown in the figure 1, a section consists of a part containing the brain section, and another which is a magnetic material used to recognize and order each section. Our work will therefore be, in a given image, to locate each of these parts for all the sections found. A section also often contains a "dummy part" (smaller), which is not relevant to extract, and only there to help during the slicing of the sections. It can be attached to the section or not, so its position is very random from one wafer to another.

Finally, in addition to the grayscale images just mentioned, we also used images in which only the magnetic parts are present. We will then see that they will help us to differentiate the magnetic parts from those containing a brain section.

III. Data preprocessing

A. Data preparation

So we have just seen that at the input of our system, we will send very large images that all contain a lot of features. However, we will see later on that we will use complex deep learning models that work better on small images. Thus, we will work from now only on smaller patches of our images. The first pre-processing step is therefore to divide a wafer image into multiple smaller images, each containing about 4 or 5 sections. More precisely, we will create images of size 512x512 to train on.

Our initial dataset was containing the gray scale images of wafers and corresponding fluorescent images that are highlighting magnetic resin part. This information was used to create three channel images (RGB images) by combining the grayscale image in two channels and the fluorescent image in a third channel. We will therefore work with

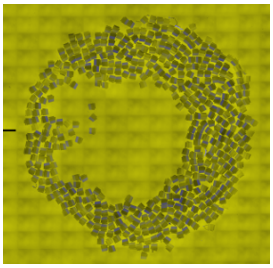

Figure 2: RGB image containing 3-channels: whole wafer, fluorescent magnetic part, blank image.

RGB images, in which we believe colour would make it possible to differentiate the magnetic parts from the brain tissue during the training. The example of RGB image can be seen in the figure 2.

B. Artificial data generation

After carefully analyzing our data in the previous section, we derive from this work a key point: The amount of data is maybe not large enough to optimally train complex deep learning models. However, the characteristics of our images, and their similarity, will allow us to solve this problem.

Indeed, in view of the images we were provided to address the problem, we can quickly notice that the recognition of the sections is relatively simple, since our data is only made up of these sections, and of the background. So we implemented a script that allowed us to artificially generate new images on which to train our model.

The first part of this work was to extract a large number of sections from the images. With these, we were then able to project them randomly onto a predefined background, resulting in an image similar to the one that would be provided to our final model. We had to make sure that two sections did not overlap, but that they remained as close as possible to the type of image shown in the figure 1. A result of our implementation is shown in figure 3. We then implemented a script to generate these data, allowing the choice of the original wafer base image, the number of image produced and the size. These data were particularly useful as the original wafers are highly crowded with sections, and do not show a lot of background. Our generated images helped for the distinction between sections and background.

Afterwards, we used a lighter second version of the image generator, which allows to produce artificial images on manually labellized dataset, using the labelme tool, based on unknown images. Indeed, we were provided with new unknown wafers without ground truth. Our task was then to be able to run the model on these. The workflow was to labellize a few of the new sections manually, and generate several artificial images based on these.

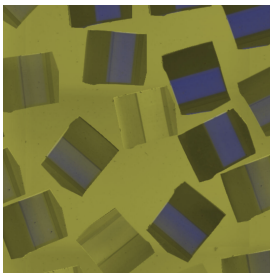

Figure 3: Patch generated artificially

C. Data augmentation

The idea behind the data augmentation can be seen on figure 4. Now that we have succeeded in generating and obtaining a very large dataset, we will try to improve it a little in this part. Indeed, to train an efficient neural network, we need to think of images taken under all sorts of conditions. Making our model able to cope with all these changes that may occur means making it more robust.

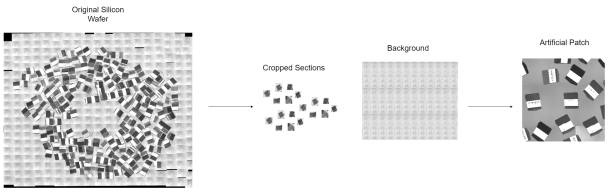

Figure 4: Idea behind the data augmentation

What we are trying to do here is in fact to teach our neural network about something called invariance, which is the ability to recognize the feature, regardless of the conditions that it is presented in. This process consists of using an image in the dataset, on which the model will theoretically have little difficulty, and applying all these modifications artificially to it, creating as many new images as changes applied. But we must be careful when applying all these changes to our images. Indeed, we need to make sure that the data is still relevant to our specific ideal model. For example, the shape of the sections we are trying to detect will always be a polygon. Applying a distortion to an image that would modify our sections shape would therefore be a mistake, because we would teach the network incorrect information.

Then, when the images are loaded to the model, we apply several augmentation techniques using random crops, random rotations, gaussian blurring, and random horizontal and vertical flips.

IV. Models & methods

A. Baseline model

The first challenge of our model will be to detect a section in the image, and thus successfully determine a region of interest in which the section is located. This is called detection.

Then, once our model has located a part of the image in which a section is located, it will be necessary to extract precisely the contours to obtain its orientation for example. This is segmentation.

B. Mask-RCNN

The implementation used is based on existing implementation of Mask-RCNN by Matterport Inc. which is itself based on the open-source libraries Keras and Tensorflow [3], [4], [5]. This implementation is well documented and easy to extend for our purposes. For additional information, we refer readers to our GitHub repository notebooks [1] and [2]. Mask-RCNN relies on region proposals which are generated via a region proposal network. It follows the Faster-RCNN model of a feature extractor followed by this region proposal network, followed by an operation known as ROI-Pooling to produce standard-sized outputs suitable for input to a classifier. Mask and class predictions are divided and independent. Masks are covering the parts of the sections (whether brain or magnet) and class predicts whether it is brain or magnet. The mask network predicts the mask independently from the network predicting the class. The Mask-RCNN allows very accurate instance segmentation masks as well as it adds a small fully convolutional neural network [6] to produce the image segmentation. This entails the use of a multitask loss function $L = L_{cls} + L_{bbox} + L_{masks}$. Mask-RCNN is built on a backbone convolutional neural network architecture for feature extraction which can use a feature pyramid network (FPN) such as ResNet-50 in its backbone to obtain great performances in both accuracy and speed.

C. Transfer Learning

Indeed to optimize the detection process we used the Resnet50 model as a backbone to train our model. The transfer learning is particularly useful to gain accuracy with these dedicated models. Additionally, once we trained using the original dataset and the artificial images, we used the newly generated weights as a pretrained model to train only the head layers on the new wafer images provided. The new wafers are not provided with ground truth labels, we then implemented the tool to generate artificial images based on a few manually labellized sections. This allows the model to discover new shape of sections and to train a bit further.

D. Model Implementation

Using our data generation scripts, we created and organized 900 images for the training set, 90 for the validation set and 50 for the test set, all associated with their associated masks, with size of 512 or 1024 depending on the section sizes. We configured our specific model based on the Nucleus sample. We then set the number of validation steps per epoch to be more than 100, relating to our validation set size. We modified the anchors size to go from 16 to 256, as we sometime have big section, but we kept small anchors to be able to detect the cropped parts in the edges. As we have highly some images containing a lot of sections, we set the parameters linked to the number of maximal detections or ground truth to a high value (more than 300). Furthermore, we implemented the model by setting the resizing mode as cropping with a size of 512, which is similar to our dataset. Finally we trained all the layers of the model on 40 epochs as the loss was still decreasing.

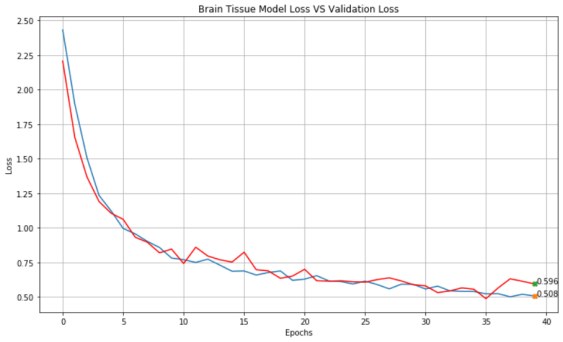

Figure 5: Evolution of the loss (blue) and the validation test loss (red)

As we can see on the figure 5, the model does not seem to overfit as the variance between the loss and validation loss is low, neither to underfit. However, we can observe a growth in the variance for the epoch 40, probably leading the idea of overtfitting. That is why we kept the model from the epoch 35. We finally computed the mean averaged precision (mAP) and the mean recall based on our separated test set images, comparing the ground truth bounding boxes with the predicted ones. These computations were made by setting the Intersection over Union (IoU) threshold to 0.5.

Averaged Precision	Recall
0.8072	0.8134

E. Hardware Details & Specifications

Cloud used for Machine Learning pipeline is Paperspace [7]. The machine used had following characteristics: RAM: 30 GB | CPUs: 8 | HD: 100 GB | GPU: 8 GB

We needed additional hard disk size since we were generating GBs of new data. We used an optimized machine to work with dedicated environment as cuDNN and the multiple python modules.

V. Results

A. Image segmentation

One result of our training could be seen on the figure 6 where we implemented our model on an cropped unknown wafer. Depending on the provided wafer, we are able to modify the detection confidence threshold from 0.9 to 0.99. As these images were not included in the training set, it confirms that the model is not overfitting.

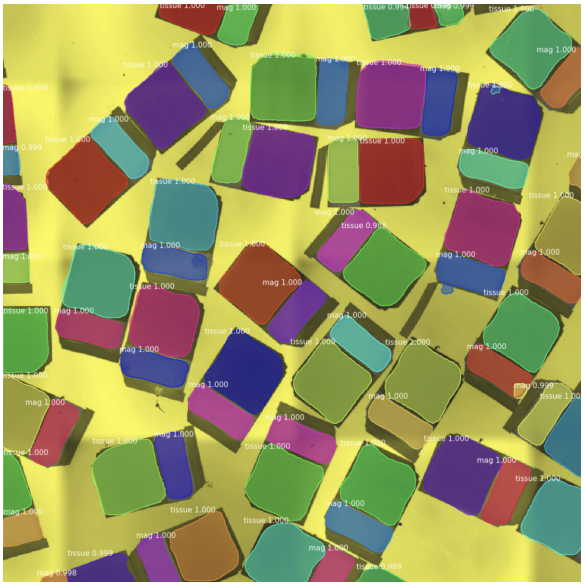

Figure 6: Brain Segmentation on new wafer

B. Pipeline

The model is sometime producing false positives or is missing some targets on the new wafers. It depends on the shape of the unknown sections. Indeed, if the shape is similar to those from the training set it will perform efficiently, whereas if the shapes are different there could be some detection problems. We then implemented a pipeline, where for some new wafers, we will be able to labelize some sections of them manually in a json file, and to produce artificial data based on them with the associated masks. Additionally, we will generate the 3-channels images from the original and the fluo images, to fit with the usual images used in the model. We will then be able to set up a new small training, focusing only on the heads layers with a few epoch, starting from the previously implemented model. Finally, as the wafer images are usually big, the model can not apply directly the detection which is causing memory issues. We then implement a script to crop the entire image into several overlapping images, which are then implemented and segmented. Furthermore, we apply a method to filter the bounding boxes on each crop, depending on the position with the next overlapping crops, to finally produce a general image.

VI. Discussion

At the end of this project, we have a complete and operational pipeline. In addition, we have discussed in this report different techniques that we have implemented to make our model more efficient and robust. However, we may ask ourselves how to go a step further and optimize our pipeline even more.

The first area of progress is in data pre-processing. Indeed, our script that enables us to artificially generate training images imitates very well the ideal experimental behavior that gives the wafer images, but not enough all the small defects that can appear in them. In practice, for example, some of the sections may be slightly ripped, or the photography technique may show black bands in the images. Today, our model perfectly detects a correct section, but makes mistakes on these practical flaws. We could try to produce more training data that would include all of those in them.

Then, a second area of improvement for our work is to think about other types of machine learning models, perhaps even more advanced and effective in our case. For example, we can think about using an architecture similar to MaskRCNNN2Go developed by Facebook [8]. It uses the Mask R-CNN framework to detect objects in an image, while simultaneously predicting key points and generating a segmentation mask for each object. In our specific problem, the key points would be the corners of the sections, and we believe such framework could give interesting results.

VII. Summary

Instance segmentation is generally challenging because it requires the correct detection of all objects in an image while also precisely segmenting each instance. In our project, we had to work on the images of the wafers that were containing sections of brain tissue. Using the state-of-art framework developed by Facebook AI Research called Mask R-CNN [9]. However, in order to use this framework on our dataset, we needed to perform modifications to the initial implementation as well as perform data augmentation in order to train the model and have the smallest loss of mask possible. In the end, we successfully managed to detect and precisely segment the sections depending on the class, whether it was brain part or magnetic part.

Acknowledgements

We would like to thank Thomas Templier for all the help he has given us throughout this work, first by introducing us to his project and its challenges, and then by being attentive to the difficulties we have faced and ready to help us when possible. Carrying out a machine learning project on such a practical case requires a lot of communication to successfully identify the specific issues and risks involved. It is therefore thanks to the very easy communication with Thomas that we have succeeded in obtaining an operational model that best meets the problem at stake.

We would also like to thank Krishna Kanth Nakka for explaining us and supporting our development process and ideas during the office hours.

Finally, we thank our professor Martin Jaggi who accepted and supported the idea of the collaboration with another EPFL laboratory.

Without these people, this project would not have been possible.

References

[1] "Tissue parts detection," https://github.com/BrainSegmentation/tissue-parts-detection/tree/master/notebooks.

[2] "Mask r-cnn implementation for brain tissue detection," https://github.com/BrainSegmentation/Jekeel_Mask_RCNN.

[3] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow," https://github.com/matterport/Mask_RCNN, 2017.

[4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016. [Online]. Available: http://arxiv.org/abs/1603.04467

[5] Keras, "Keras: The python deep learning library," https://keras.io/.

[6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[7] Paperspace, "The first cloud built for the future. powering next-generation applications and cloud ml/ai pipelines." https://www.paperspace.com/.

[8] T. F. A. C. Team, "Maskrcnnn2go," https://research.fb.com/enabling-full-body-ar-with-mask-r-cnn2go/, 2018.

[9] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron," https://github.com/facebookresearch/detectron, 2018.