CS230

Nuclear segmentation with Deep Learning

Antoine Bargé
Department of Management Science and Engineering
Stanford University

Abstract

Efficient and automated cell segmentation techniques are very important to pharmaceutical research. In this report, I present two convolutional neural networks and how they were applied to a cell segmentation problem. The dataset comes from the ISBI 2012 challenge and is composed of images of neuronal structures. The training strategy relies on data augmentation to use the available annotated samples more efficiently. SegNet and U-Net were implemented and compared.

1 Introduction

In the last few decades advances in the microscope technologies have led to rapid growth in the number and resolution of cell images being captured. To obtain biologically meaningful results, it is necessary to analyze large number of cells in multiple samples. However, doing these analyses manually is very inefficient. One of the key barrier of the analysis is image segmentation, which is identifying which parts of an image belong to which individual cells. Without this mapping, it is impossible to extract statistics on cellular properties. Cell segmentation is of significant interest to a wide range of medical imaging tasks. An example is breast cancer, where the tumor growth is an important indicator of patients' prospects. Nowadays, the most common method is performed by pathologists, who examine biopsies under a microscope. Although this method is very accurate in most cases, generally it is slow and prone to fatigue induced errors.

Cell detection methods are now evolving from employing hand-crafted features to deep learning-based techniques. Recent methods use convolutional neural networks to address this challenge. They perform better than other methods and are easier to be shared and applied on new data. In this project, the input of the network is composed of microscope images and the output are the labelled images where one can see the membranes of the cells. They are binary masks of same size of the input with background regions labeled as 0 and membranes as 1. CNNs are used to make the prediction.

2 Related work

Many imaging technologies produce microscopic images. For example, the MIBI (multiplexed ion beam imaging) technology produces images of cells to study cancer. In [2], one can read the results of its application to breast cancer. The study required cell segmentation methods and one package now widely used is described in [3]. This method being based on old techniques and not performing very well, I focused myself on new methods, such as [1] and [4] respectively describing the U-Net and SegNet. In [5] the Tiramisu framework is developed and can be also used for that problem. Those are state of the art methods for segmentation problems.

Cell segmentation was also studied by A. Haigh, F.vPaasschen and J.Murphy for the CS230 class in Spring 2018 and by W.Zhou and Y.Zhao in Fall 2018. They both use the U-Net but the applications are different.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3 Dataset and Features

3.1 Description of the dataset

The original dataset is from ISBI challenge *Segmentation of neuronal structures in EM stacks*. The goal of the project is perform an automatic segmentation of the neural structures. The images are representative of actual images in the real-world, containing some noise and small image alignment errors.

The training data is a set of 30 sections coming from the drosophila first instar larva ventral nerve cord. The corresponding binary labels are provided in an in-out fashion, i.e. white for the pixels of segmented objects and black for the rest of pixels (which correspond mostly to membranes). All the images are 512×512. The training and test data both contain 30 512×512 images.

On figures 1 and 2 one can see an example of the images and how they are labeled.

3.2 Data augmentation

To be able to feed a deep learning neural network, I performed data augmentation. To do so, I use a module called *ImageDataGenerator* in Keras. I augment my training examples with random transformations, so that my model never see the same picture twice. This helps prevent overfitting and helps the model generalize better. I use the following transformations:

- Rotation-range is a value in degrees (0-180), a range within which to randomly rotate pictures. I used 0.2.
- Width-shift and Height-shift are ranges within which to randomly translate pictures vertically or horizontally. I used 0.05 for both.
- Rescale is a value by which it is possible to multiply the data before any other processing.
- Shear-range is for randomly applying shearing transformations. I used 0.05.
- Zoom-range is for randomly zooming inside pictures.
- Horizontal-flip is for randomly flipping half of the images horizontally. I used 0.05.
- Fill-mode is the strategy used for filling in newly created pixels, which can appear after a rotation or a width/height shift. I used 'nearest'.

New training images were generated with data augmentation, following deformations such as on figures 3 and 4.

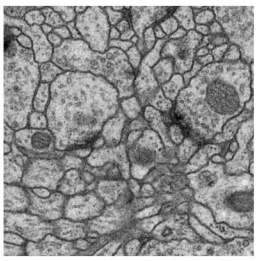

Figure 1: Example of training image.

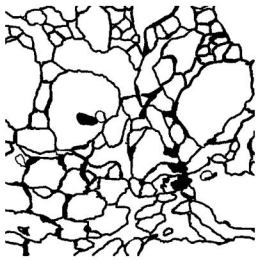

Figure 2: Corresponding beled image.

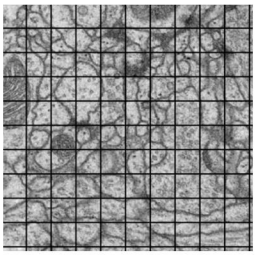

Figure 3: Image before la- transformation.

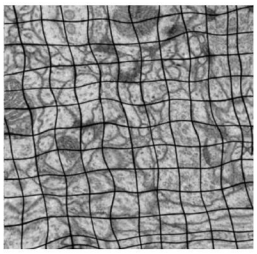

Figure 4: Corresponding transformed image.

4 Methods

4.1 U-Net

U-Nets are very appropriate for cell segmentation because they combine high resolution with local and global features.

It consists of a contracting path and an expansive path. The contracting path consists of repeated application of convolutions, each followed by a ReLU and a max pooling operation. Then, the spatial information is reduced and feature information is increased. The expansive path combines the feature and spatial information. It uses up-convolutions and concatenations with the features from the contracting path.

On the figure 5, one can see the U-Net architecture.

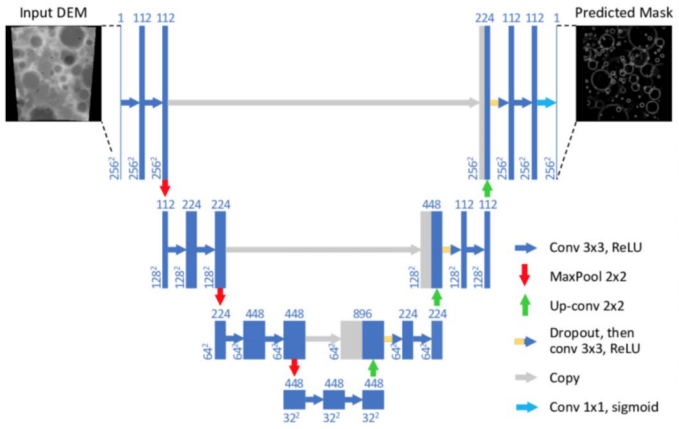

Figure 5: Architecture of the U-Net: starts with a 3×3 Convolution/ReLU. Then MaxPool 2×2 – 3×3 Convolution/ReLU are repeated three times. After that the MaxPool 2×2 are replaced by 2×2 Up-Convolution and the process is repeated again. The final layer is a 1×1 Convolution and gives a binary pixel mask. This mask is the output of the algorithm and is used for segmentation.

4.2 SegNet

SegNet also consists of a sequence of encoders and a corresponding set of decoders followed by a classifier. The encoder layers extract image features using deep convolutional network. Each of those are composed of convolutional filters, followed by ReLUs and max-pooling to downsample image features. The decoder layers upsample the feature map back to image resolution. The sparse encoding due to the pooling process is upsampled in the decoder using the maxpooling indices in the encoding sequence. The final output has the same number of channels as there are pixel classes.

On figure 6, one can see the SegNet architecture.

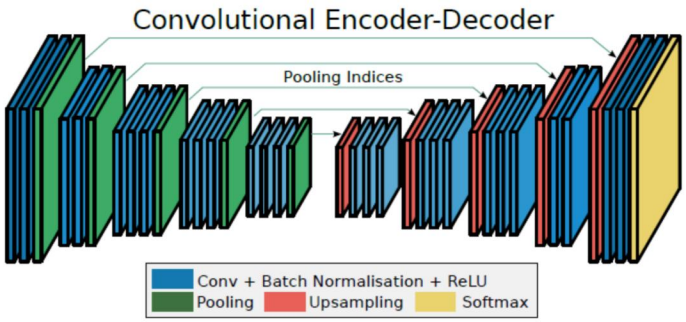

Figure 6: Architecture of the SegNet: at the encoder, convolutions and max pooling are performed. There are 13 convolutional layers and while doing 2×2 max pooling, the corresponding max pooling indices are stored. At the decoder, upsampling and convolutions are performed; there is a softmax classifier for each pixel. During upsampling, the max pooling indices at the corresponding encoder layer are recalled to upsample as shown above. Finally, a softmax classifier is used to predict the class for each pixel.

4.3 Differences between the two models

The loss used is the binary cross entropy:

$$loss(y, \hat{y}) = -(ylog(\hat{y}) + (1 - y)log(1 - \hat{y}))$$

U-Net was created for biomedical image segmentation. Instead of using pooling indices, the entire feature maps are transferred from encoder to decoder, then with concatenation to perform convolution. This makes the model larger and need more memory but it should capture the details more precisely.

5 Experiments/Results/Discussion

5.1 Hyperpameters and metrics

The chosen batch size is 2. It allows to have not a too big memory space. Also, since the training sample is small, having a small batch size makes sense. It made the training slower, with approximately 10 minutes per epoch for the U-Net, but the convergence was faster. There are 2000 steps per epoch and 5 epochs in total. Random data was indeed generated thanks to data augmentation. Therefore, the training set has a generated infinite size. Because of the random process of the data augmentation, there are never two identical training epochs.

Three optimizers were tested: SGD, Adagrad and Adam. Adagrad is an optimizer with parameter-specific learning rates, which are adapted relative to how frequently a parameter gets updated during training. The more updates a parameter receives, the smaller the learning rate. With these three optimizers I tuned the learning rate following a panda approach. The results of this experiments can be seen in the next section. The primary metric used is the accuracy.

5.2 Results

The best accuracy was reached with a U-Net using Adam optimizer with a learning rate of 10^{-4}. The main results can be seen in the table 1. Other values of the hyperparameters were also tried but gave lower scores, so they don't appear in the table.

Table 1: Accuracy of the U-Net and SegNet depending on the chosen hyperparameters.

Model	Optimizer	Learning Rate	Accuracy
U-Net	SGD	10^{-4}	0.885
U-Net	SGD	10^{-3}	0.867
U-Net	Adagrad	10^{-4}	0.878
U-Net	**Adam**	$\mathbf{10^{-4}}$	**0.967**
U-Net	Adam	10^{-3}	0.958
U-Net	Adam	10^{-2}	0.910
SegNet	SGD	10^{-4}	0.782
SegNet	Adam	10^{-4}	0.810
SegNet	Adam	10^{-3}	0.814

The U-Net models generally perform better than the SegNet on the current problem. The results vary a lot with the different optimizers and Adam performs the best. Adagrad and SGD give similar scores. With the best model, the accuracy is 96.7%.

5.3 Discussion

On the figures 7 to 10, one can see examples of predictions. They correspond to the best U-Net and SegNet models. It can be seen that U-Net performs better.

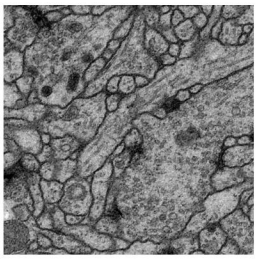

Figure 7: Example of test image.

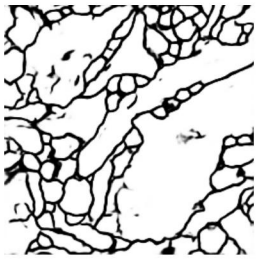

Figure 8: Corresponding prediction with U-Net.

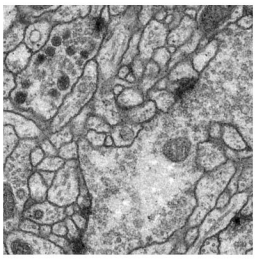

Figure 9: Example of test image.

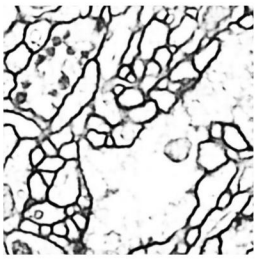

Figure 10: Corresponding prediction with SegNet.

Data augmentation, with random elastic deformations of the training sample, was essential to teach the network and reach the final accuracy. Without it the scores were much lower and the predicted images not coherent. With very few annotated images (initially there were 30 images in the training dataset), it is possible to reach high scores.

Looking at the image predictions, it can be seen that SegNet tends to miss out the finer details especially at the boundary between white and black areas. U-Net on the other hand, because of the skip connections from the lower levels, is able to capture the fine details more precisely. The boundaries are therefore better represented. There are big differences at the boundaries between the two frameworks. However, U-Net is not very powerful where one class is present in abundance. Also, one can observe random noise in the U-Net segmentation.

6 Conclusion/Future Work

The goal of the project was to perform cell segmentation on neuronal microscopic images using convolutional neural networks. The highest scores were achieved with a U-Net and the best score is 96.7% accuracy. The fact that U-Net performed better than SegNet is not surprising because it was created to segment biological images. Data augmentation was used due to the small size of the training dataset.

A possible future work could be to leverage transfer learning in order to compensate the fact that the amount of training data is low. Using the knowledge learned while solving a problem and applying it to the cell segmentation problem could be a good improvement. Other CNN frameworks, such as Tiramisu or U-SegNet [8], could also be implemented and compared to the U-Net. U-SegNet is another Fully Convolutional Neural Network (FCN) combining the architectures of the SegNet and the U-Net. Although the base architecture resembles to a SegNet, it has skip connections inspired from U-Net.

7 Code

The code is available at https://github.com/abarge/CS230. It contains the U-Net model in model.py, the SegNet in segnet.py and the data augmentation in generator.py.

References

[1] Ronneberger,O.,Fischer,P.,Brox,T.:U-Net: Convolutional Networks for Biomedical Image Segmentation. In: MICCAI (2015)

[2] Keren L, Bosse M, Marquez D, Angoshtari R, Jain S, Varma S, Yang SR, Kurian A, Van Valen D, West R, Bendall SC, Angelo M: AStructured Tumor-Immune Microenvironment in Triple Negative Breast Cancer Revealed by Multiplexed Ion Beam Imaging, in Cell, 2018

[3] David A. Van Valen, Takamasa Kudo, Keara M. Lane, Derek N. Macklin, Nicolas T. Quach, Mialy M. DeFelice, Inbal Maayan, Yu Tanouchi, Euan A. Ashley, Markus W. Covert: Deep Learning Automates the Quantitative Analysis of Individual Cells in Live-Cell Imaging Experiments, 2016

[4] V.Badrinarayanan, A.Kendall, R.Cipolla: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, IEEE Conference, 2015

[5] Jégou S., Drozdzal M., Vazquez D., Romero A., Bengio Y., The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation, IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017

[6] Ciresan, D.C., Gambardella, L.M., Giusti, A., Schmidhuber, J.: Deep neural net- works segment neuronal membranes in electron microscopy images, in NIPS, 2012

[7] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation, arXiv, 2014

[8] Pulkit Kumar Pravin Nagar Chetan Arora Anubha Gupta, U-SegNet: Fully convolutional neural network based automated brain tissue segmentation tool, arXiv, 2018

[9] Libraries used: Tensorflow, Keras, Matplotlib, Numpy, Scikit-image, Scipy