אוניברסיטת
בן-גוריון בנגב

Department of Electrical and Computer Engineering

Deep learning and its applications to Signal and Image Proccessing and analysis

361-2-1120

Final project

Dor Livne: 204528251

Moshe Koziashvili: 204038079

Semester b, 2019

abstract:

In this Project we implement segmentation of biological cells using a compressed version of the known U-Net [1] architecture. In order to overcome the problem of small size of training data, we implemented Elastic Augmentation [2] along with more standard ones (random rotation, mirroring), and train the Network on small regions of the image. In the experimental section we examine the performance of the Network variants (augmented data, Layers batch Normalization, Batch size, training on regions of the image), the of different kinds of learning methods (such as Curriculum Learning [3]) and present some analytical results using TensorBoard tools.

Introduction:

in recent years, there has been an arise in usage of Deep Neural Network (DNN) architectures in order to perform Image-Processing Tasks such as image classification and Instance/Semantic Segmentation. The U-Net [1] is a CNN architecture which was published in 2015 and gained popularity since then. The publishers utilize the U-Net in order to apply Instance Segmentation of Biological Cells. The original structure of the U-Net is:

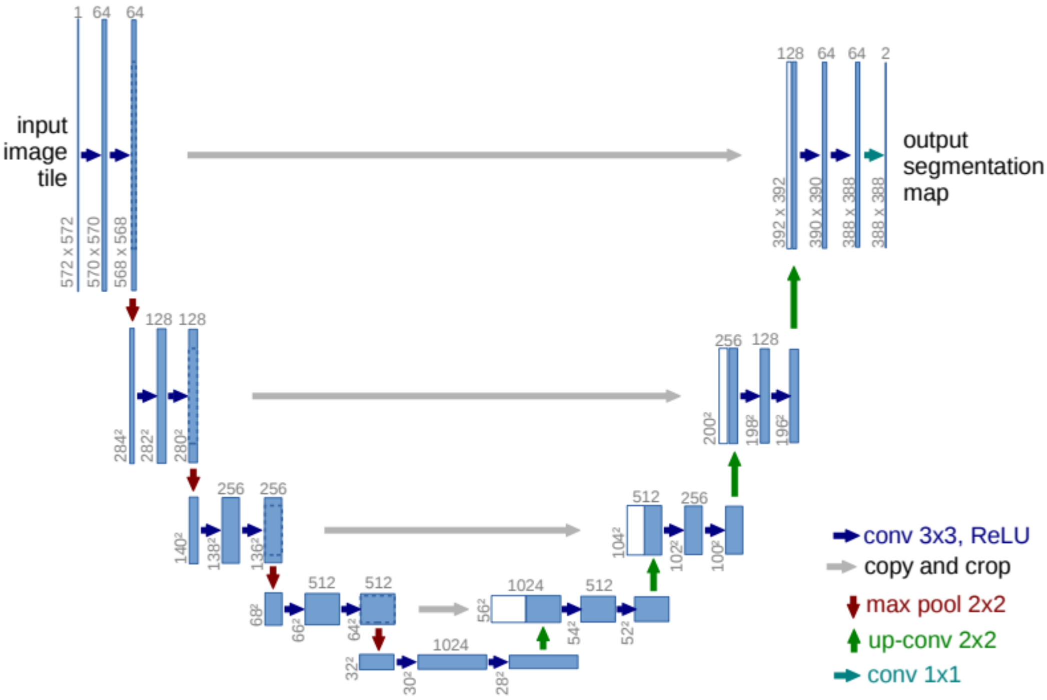


Figure 1: U-Net structure [1]

From Fig.1 we can see that we U-Net performs as a Variant of Auto-Encoder, where the left side in analogue to the Encoder (the Max-Pooling compresses the size of the image), and the right side is the Decoder. We can also see that for every level in the decoder, the corresponding level in the Encoder is been concatenated in order to preserve more information about the input image. Another issue that the publishers address is the separation of different cells in the image (as opposed in Semantic Segmentation). The overcome that challenge, the authors applied a weighted Cross-Entropy loss function, where the boundaries of two adjacent cells are heavily weighted. The loss function is:

$$-\sum_{x=1}^{W}\sum_{y=1}^{H}\sum_{c=0}^{C}\omega(x,y,c)\cdot P(x,y,c)\cdot\log(Q_\theta(x,y,c))$$

Where (x,y) is the spatial location of the pixel in the image, $P(x,y,c)$ is the original probability mass function of the classes in pixel (x,y) (one-hot vector) and $Q_\theta(x,y,c)$ in the estimated probability mass function of the Network with parameters θ. The weights are:

$$\omega(x,y,c) = \omega_c(x,y) + \omega_0\cdot e^{-\frac{\left(d_1(x,y)+d_2(x,y)\right)^2}{2\sigma^2}}$$

Where $\omega_c(x,y)$ is the normalized frequency weights (one over the frequency of the class in the current image), $d_1(x,y)$ and $d_2(x,y)$ are the Euclidean distances to the closest and second closest cell respectively. ω_0 and σ are hyper-parameters (typical values are $\omega_0 \approx 10$, $\sigma \approx 5$). This method of weighting causes to pixels which corresponds to boundary between two cells (thus, $d_1(x,y)$ and $d_2(x,y)$ are both small) to be heavily penalized. So the model learns to separate cells even if they are very close or overlapping each other.

Our Architecture:

In this work we apply supervised learning task of segmenting biological cells. The provided training data (before augmentations) are the original images and Ground-Truth segmentations, an example of the provided data:

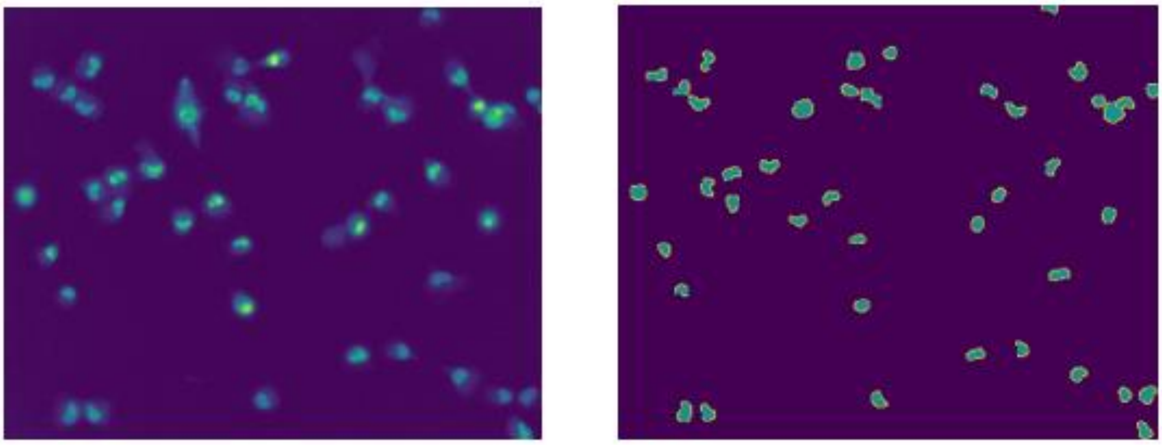
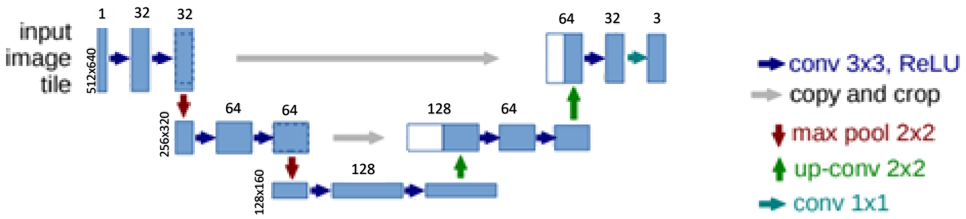


Figure 2: provided training data example. Original image and labeled image for every instance

In this work we implemented a compressed version of the presented U-Net and we will discuss the pros and cons of the exponential component in the weight map of the loss function. Our architecture is as follows:



We compressed the network both for computational complexity and to avoid overfitting due to small diversity in training data.

Our network outputs the unnormalized probability of each of the three classes.

Note: in training we noticed that the output in converging to a blank image (background only). in our opinion, the reason is that there is a local minimum to the cost function (cross entropy) for blank output because of the fact that most of the image is background. We solved that issue by substitute the activation functions in the last two layers from ReLU to linear.

Our model includes Batch Normalization for every convolutional layer (except the last two layers) before the activation function.

Augmentation:

To enhance the data size and diversity, we applied augmentations on the given data. Among standard augmentation methods such as random rotation, we also applied more sophisticated ones:

Mirroring Augmentation: in this augmentation we split the input image into 4 quarters (thus increasing the size of the training data). Every quarter in being mirrored with respect to the horizontal axis, and then the new image is being mirrored again with respect to the vertical axis. The constructed image is:

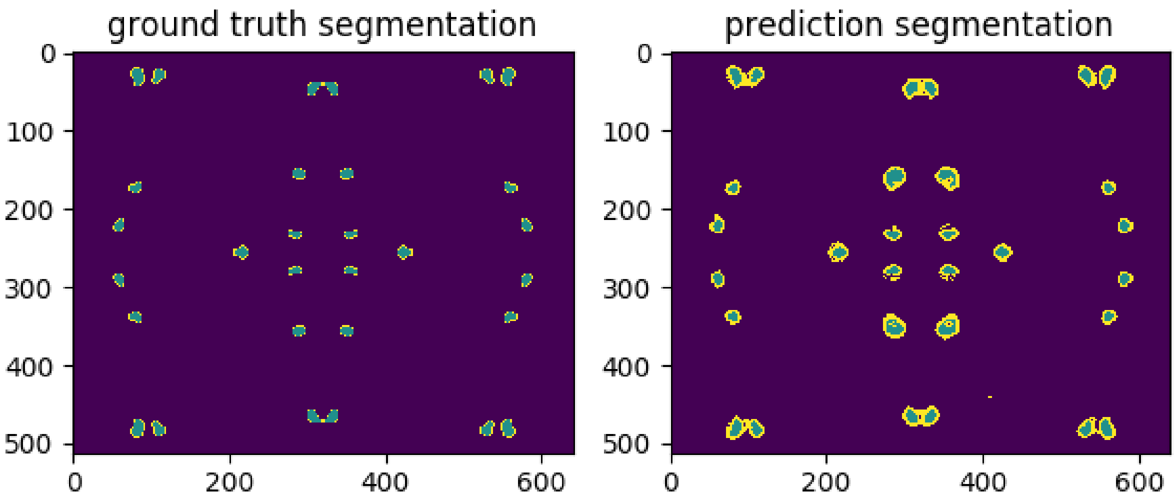


Figure 3: Ground-Truth segmentation and Predicted one, after mirror augmentation

The drawback of this augmentation is the spatial dependencies between quarters, that could cause unwanted bias in the training process (the network thinks that there is a spatial dependence for the validation images, which is obviously a wrong conclusion). For this reason, we used just a small amount of this kind of augmented data.

Elastic Augmentation [2]: The image deformations were created by first generating random displacement fields $(\tilde{x},\tilde{y})$, generated with a uniform distribution in (-1,1). The fields $(\tilde{x},\tilde{y})$ are then convolved with a Gaussian filter of standard deviation σ (in pixels).

$$(\tilde{x},\tilde{y}) = [(\tilde{x},\tilde{y}) * Gaussian_{0,\sigma}(x,y)] \cdot \alpha$$

Where α is the Influence Hyper-parameter. If σ is large, the resulting values are very small because the random values average 0. If σ is small, the field looks like a completely random field. The displacement fields are then multiplied by a scaling factor α that controls the intensity of the deformation. The new coordinates are $(x_{new}, y_{new}) = (x + \tilde{x}, y + \tilde{y})$. An example of the Elastic Augmentation is:

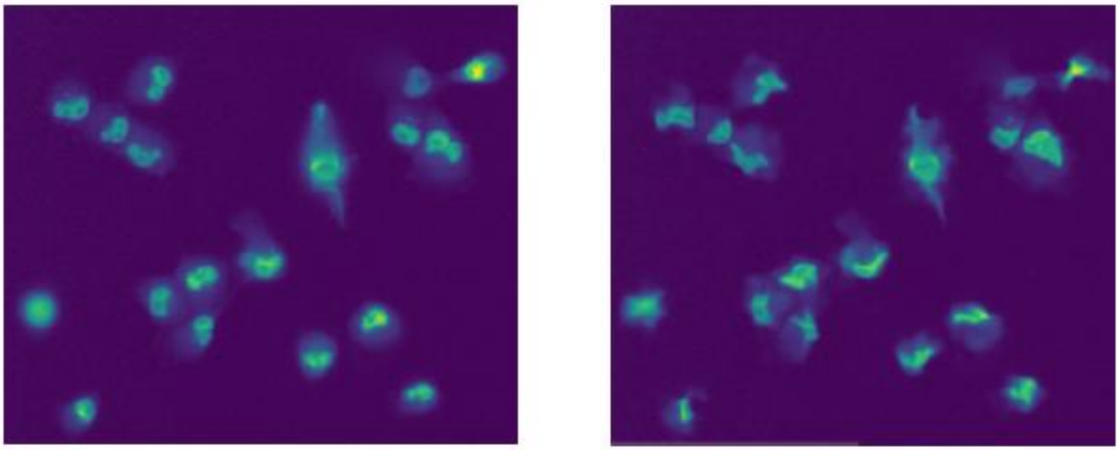


Figure 4: left- original image, right- elastic augmented image with α= 60 and σ = 4

After the elastic deformation, we also applied random rotation and mirroring to enhance the diversity of the training data.

Training:

- We used the Batch-Generator implemented in homework 1#, and added the described random augmentation in order to construct the training data.
- we performed the training of various Batch sizes (in the experimental section we compare the performance of the network for Batch sizes 1,4,8).
- Every epoch includes 6 training steps and 5 validation step (1 Batch for every step)
- Our measure of accuracy is IOU, where we measure the Intersection Over Union for the Ground-Truth cells and the predicted ones:

$$IOU = \frac{|A_{GT} \cap A_{pred}|}{|A_{GT} \cup A_{pred}|}$$

Where A_{GT} and A_{pred} are the area of the Ground-Truth and the predicted cells respectively.

- The unnormalized probabilities, obtained from the network's output, are been fed into the loss function (the function is described in the introduction section).
- We added convergence constraints for the training, which terminates the training session in case there was no improvements in validation accuracy for 10 consecutive epochs.
- Training specifications:

optimizer	Adam
epochs	400
Loss function	Weighted Cross Entropy
processor	Nvidia Titan RTX

- Learning Rate scheduler was applied as follows:

epochs	Learning Rate
1-20	1e-3
21-100	5e-4
101-150	1e-4
151-200	5e-5
201-250	1e-5
250-400	5e-6

Experiments:

In this work we performed several experiments with aim to optimize our network.

Experiment 1:

In this experiment we analyze the performance of Curriculum Learning (CL) on our network (batch size = 4). In this method the level of complexity of the training data increases further down the training. In this experiment we start the training session with only elementary augmentations, such as mirroring and random rotations. when the network achieves IOU accuracy of 0.7 in validation, we start feeding the network with Elastic augmented data with random augmentation parameter $\alpha \sim Uniform(0\,,50)$. The results of the experiments are:

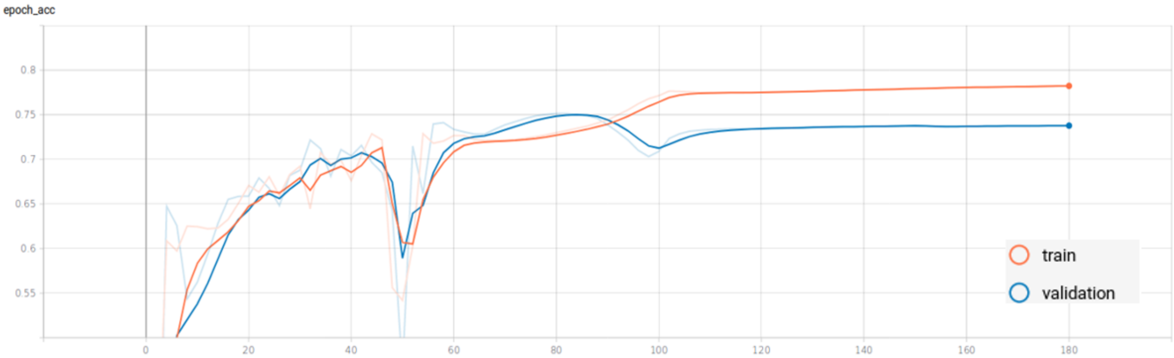


Figure 5: IOU accuracy as function of epochs for training without CL

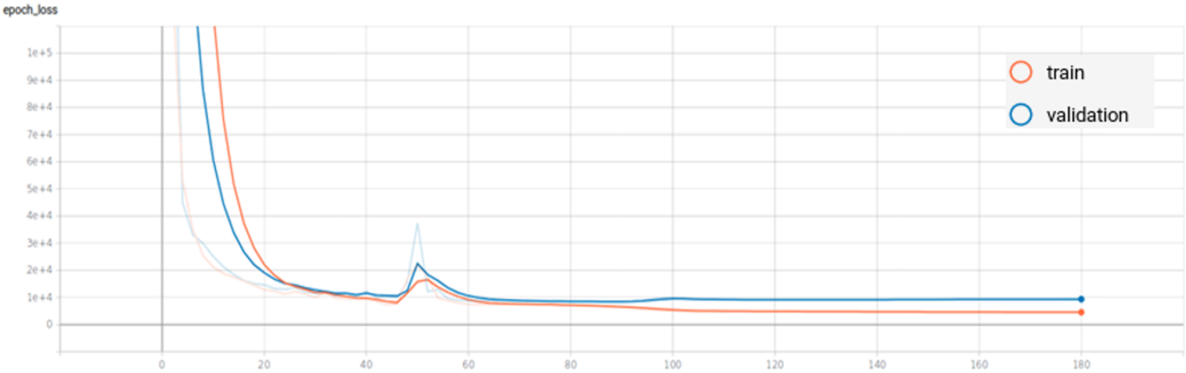


Figure 6: Loss as function of epochs for training without CL

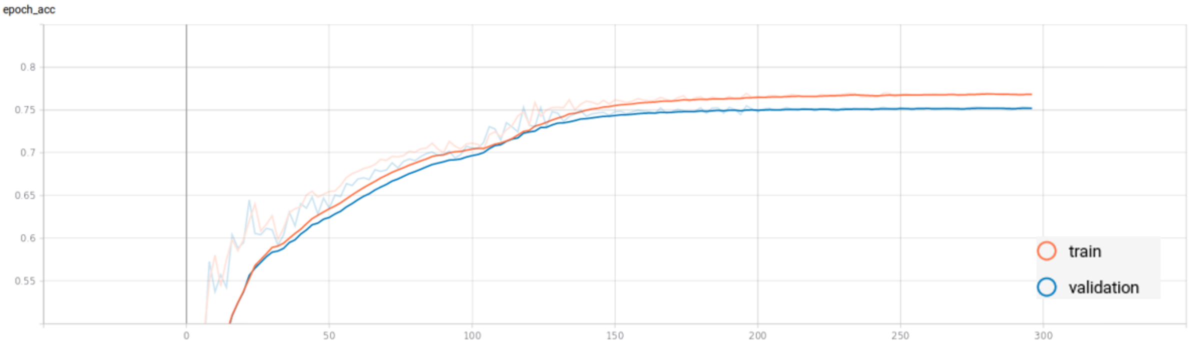


Figure 7: IOU accuracy as function of epochs for training with CL

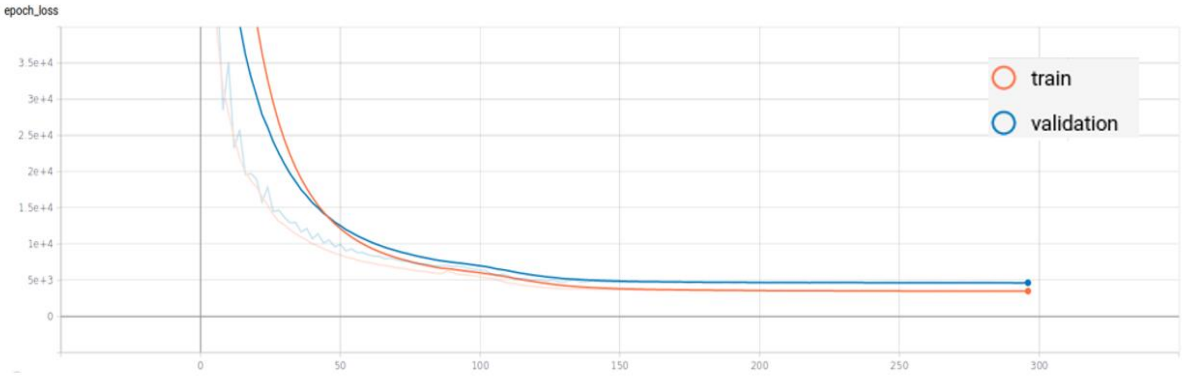


Figure 8: : Loss as function of epochs for training with CL

As mentioned above, in Figs. 4 and 5 we present learning without CL. We observe two phenomena: the first is an anomaly during training in epoch 50, which cause degradation in both IOU accuracy and the loss function. The network goes "back on trace" rapidly and the training continue. Second phenomenon is the overfitting occurs in epoch 90, which can be examined both from the degradation of the validation accuracy as oppose to the training, and in increase of the validation error while the training error decreasing. At the other hand, we can examine a monotonic improvement when training with CL, around accuracy of 0.7 the model begins to saturate, but when we increase the complexity of the data (by applying Elastic augmentation) we observe a sudden increase in accuracy (starts in epoch 100 in Fig.6) and the model continue to improve. We can conclude that training with CL improves the learning session. A visualization of the trained network's output (for validation score of 0.754 IOU):

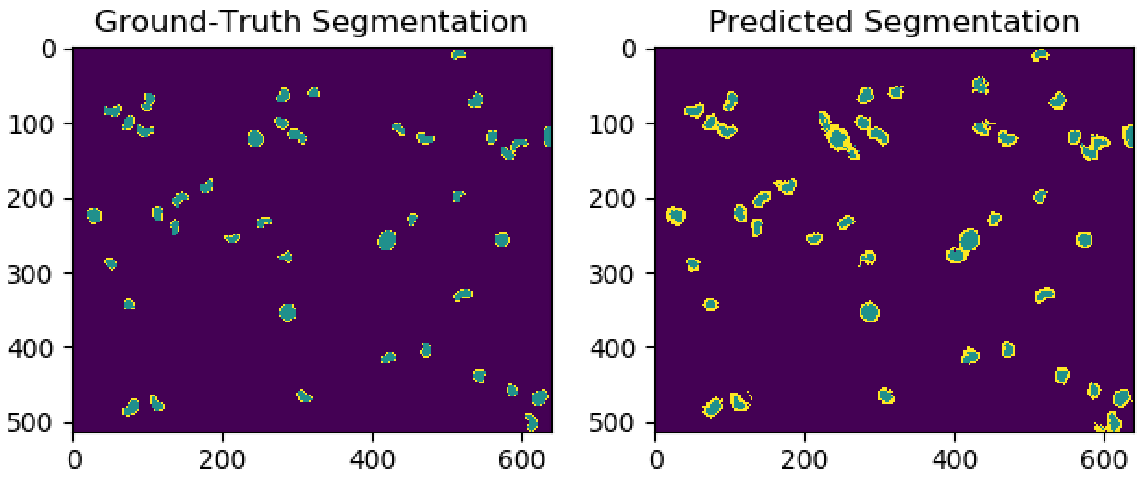


Figure 9: comparison of Ground-Truth and predicted labels

Experiment 2:

In this experiment we examine regional training of our model. The question we try to answer is "wither or not training the network on small patches of the training input improves the accuracy?"

The motivation behind this query (besides improving the network accuracy) is the reduction of the computational complexity (we parted the image into 4 sub-regions, so the network is now quarter the original size) or conversion of the computational complexity to increase the number of channels in each layer and construct more sophisticated network.

Every input to the network is been divided into 4 regions, after predicting the labels, the network's output in being reconnected to create the complete prediction image. Also, in this experiment we used both variations of learning (with and without CL):

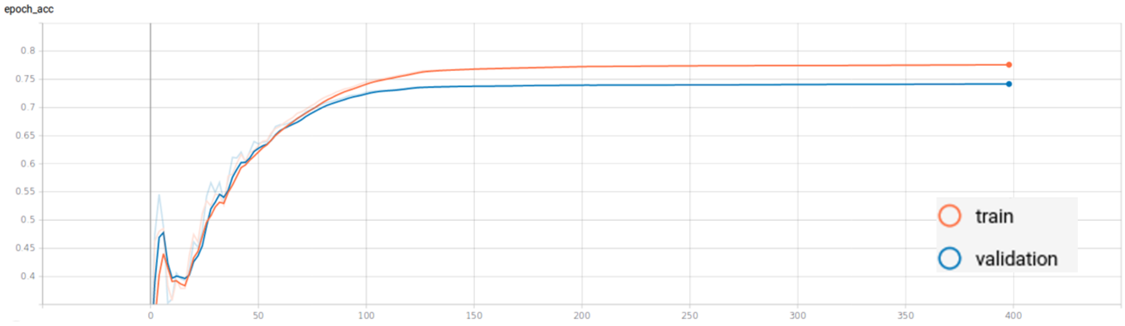


Figure 10: accuracy as function of epochs with CL

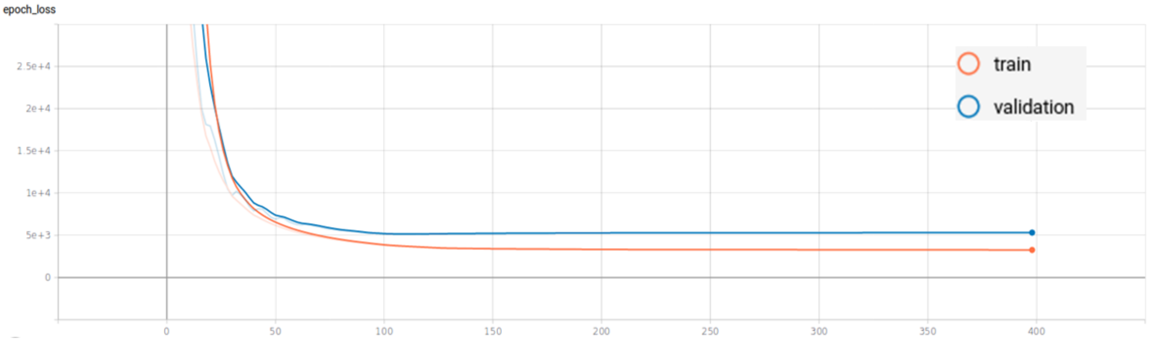


Figure 11: Loss as function of epochs with CL

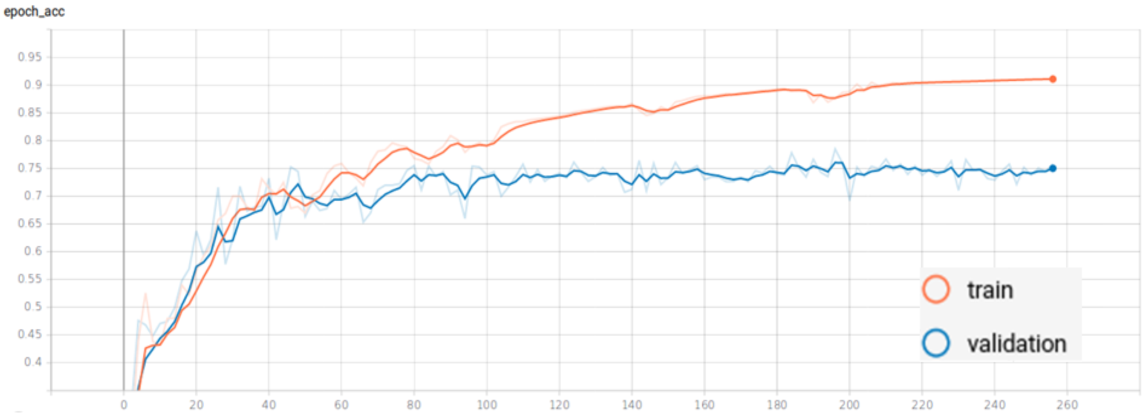


Figure 12: accuracy as function of epochs without CL

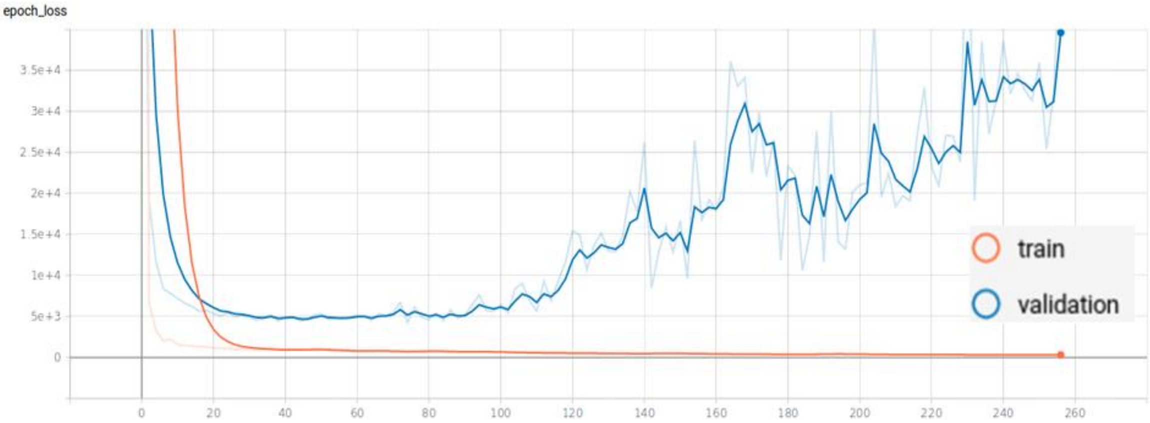


Figure 13: Loss as function of epochs without CL

Although there was no improvement in accuracy for the validation IOU score, on both case of the experiment, we examine the same conclusion as in experiment 1. While training with CL is monotonically increasing, training without CL tends to overfitting (which is presented clearly in Fig.12). an interesting phenomenon we examine was that this model was having a hard time in patching back the images in the beginning of the training, but as the training process advances, the Network learns how to patch all the tiles in:

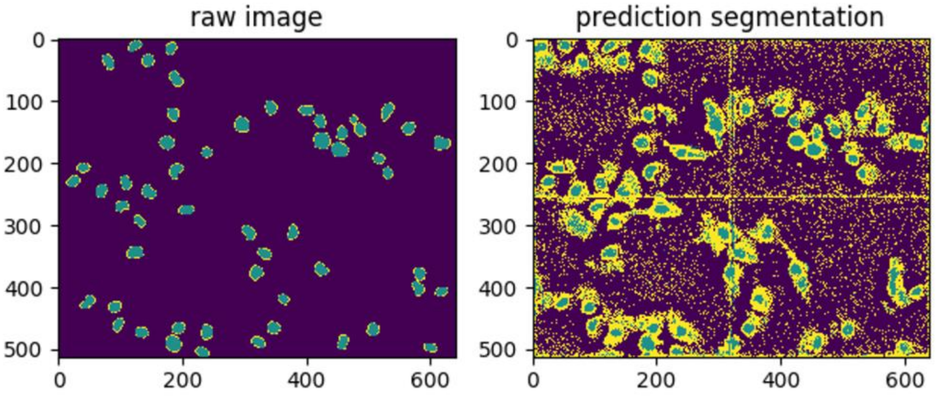


Figure 14: prediction result after 3 epochs

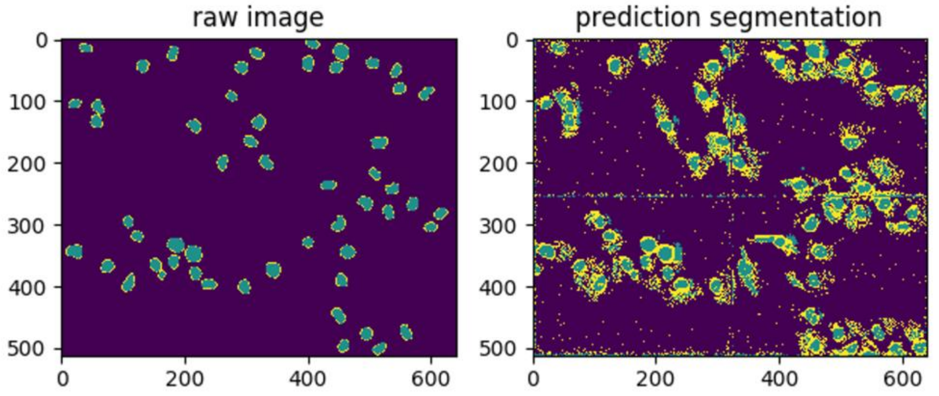


Figure 15: prediction result after 15 epochs

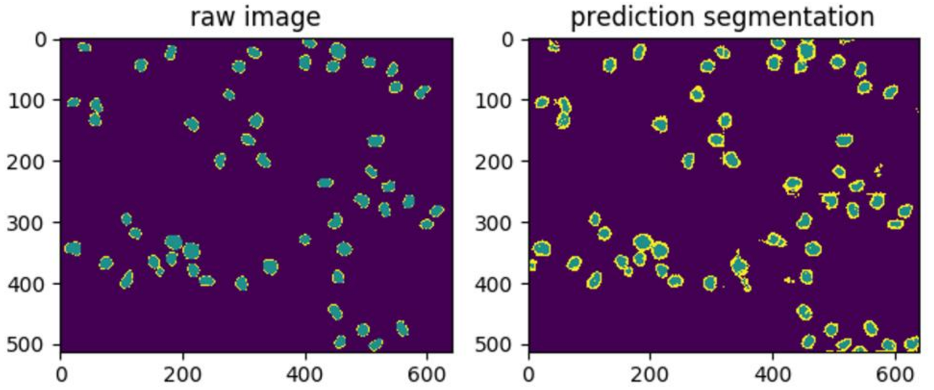


Figure 16: prediction result after 100 epochs

As we can see from the above figures, although the tiles model has lower accuracy than the other models in the first stages of the training, it eventually overcomes the difficulty of patching back the tiles and gains the approximately the same accuracy.

Experiment 3:

In this experiment we examine the influence of the Batch size on the IOU accuracy of the validation data. We train the network over Batch sizes of 1,4 and 8, this time without CL (just random rotations, flipping and mirroring). The result are as follows:

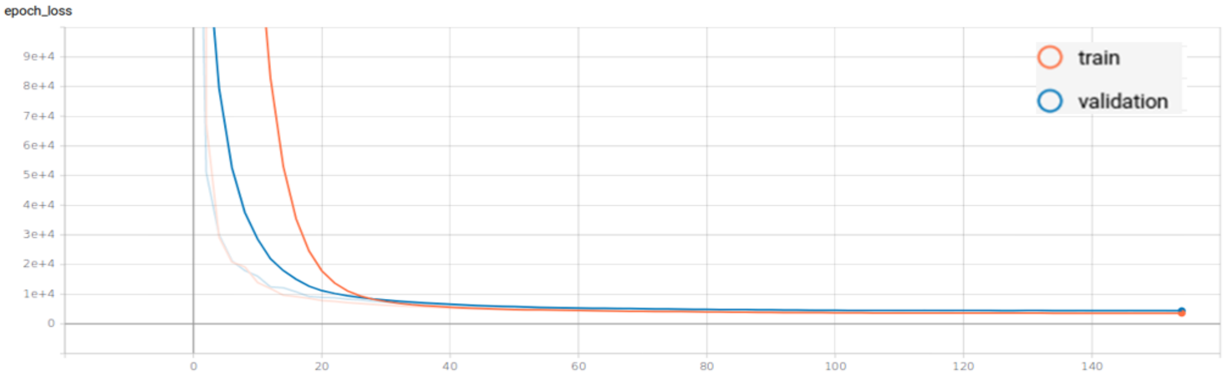


Figure 17: Loss as function of epochs for Batch size=1

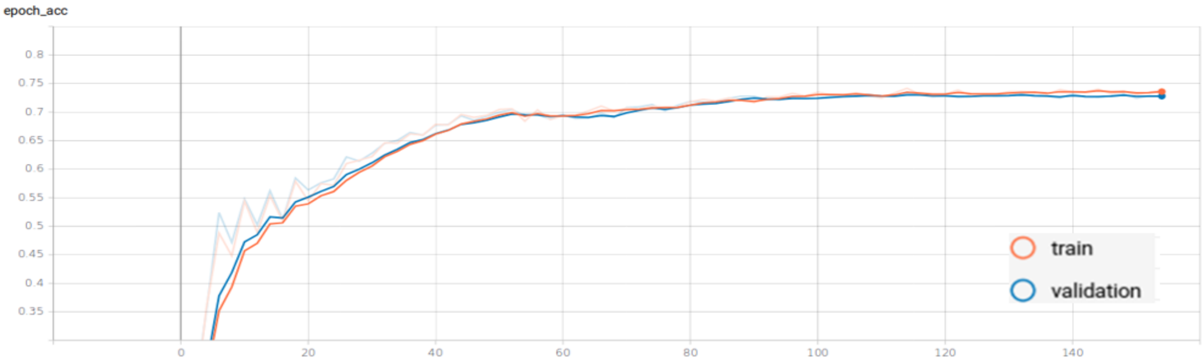


Figure 18: accuracy as function of epochs for Batch size=1

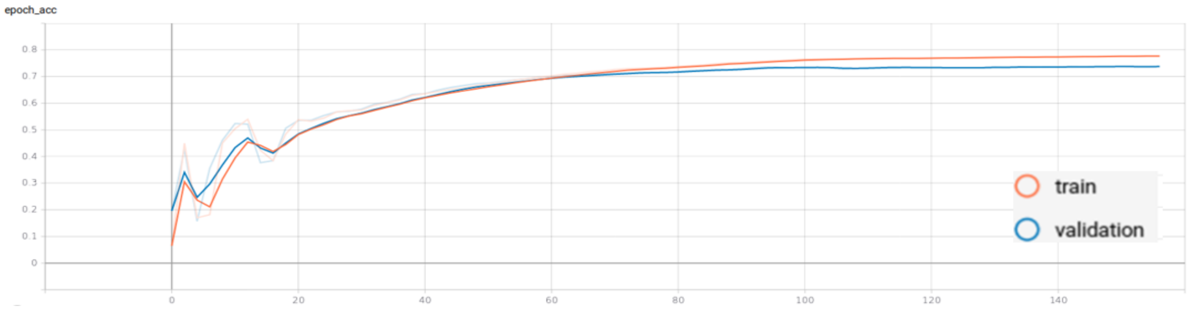


Figure 19: accuracy as function of epochs for Batch size=4

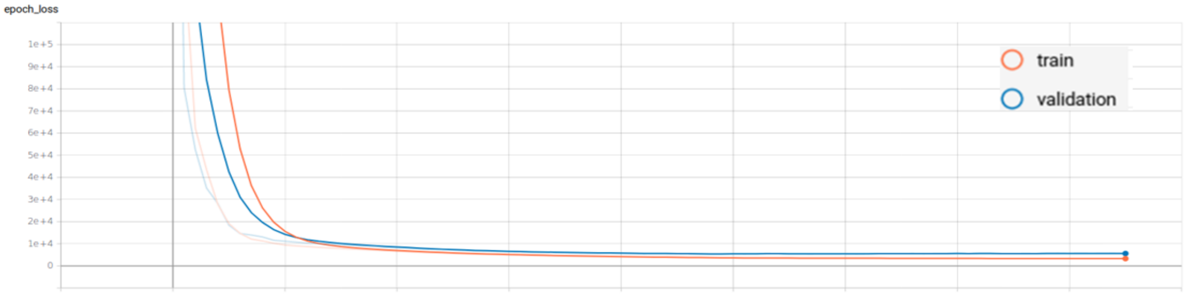


Figure 20: Loss as function of epochs for Batch size=4

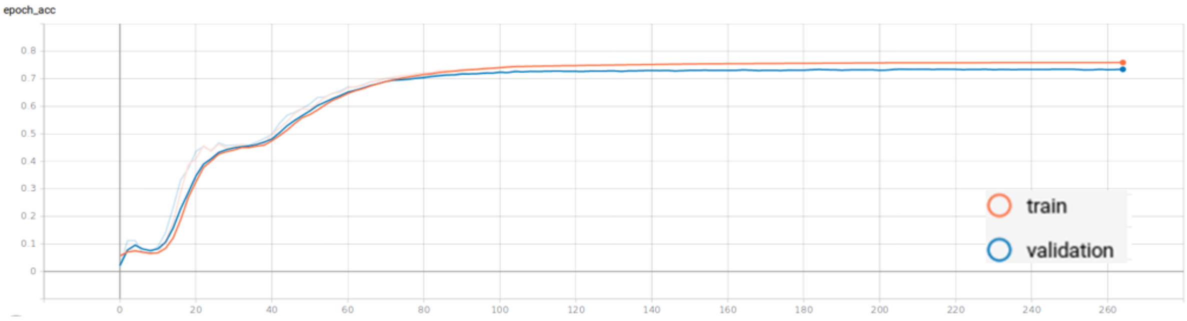


Figure 21: accuracy as function of epochs for Batch size=8

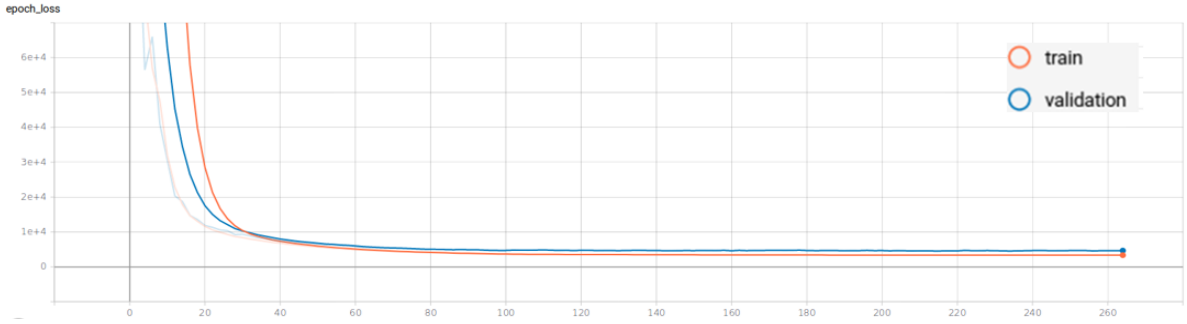


Figure 22: Loss as function of epochs for Batch size=8

From Figs. 14-19 we conclude that Batch size has minimal effect on the IOU accuracy and of the U-Net's loss function. So future experiments can be conducted regardless of the Batch size.

The next table summarizes the experimental section for the validation stage:

Tested U-Net models:	IOU (with CL)	SegMeasure (with CL)	IOU (without CL)	SegMeasure (without CL)
Vanilla (batch size = 4)	0.754	0.760	0.727	0.734
Trained on tiles	0.76	0.77	0.69	0.72
Batch size = 1	0.737	-	-	-
Batch size = 4	0.72	-	-	-
Batch size = 8	0.729	-	-	-

Discussion:

As we can see from the above table, gradual training (CL) improves the accuracy of our model, as presented also in [3], increasing the complexity of the data down the training yields more accurate models than just feed the network with randomly sampled data (this kind of training is also more analogue to the human learning process than just random observations). After the model "realizes" the basic features of the cell's structure, we challenge it with more sophisticated (Elastic Augmented) data, and by that we also avoid Overfitting (by preventing the model from memorizing in input data).

An Interesting issue that affects the score of our network is the fact that its able to detect cells that aren't present in the Ground-Truth segmented image. An example for this phenomenon:

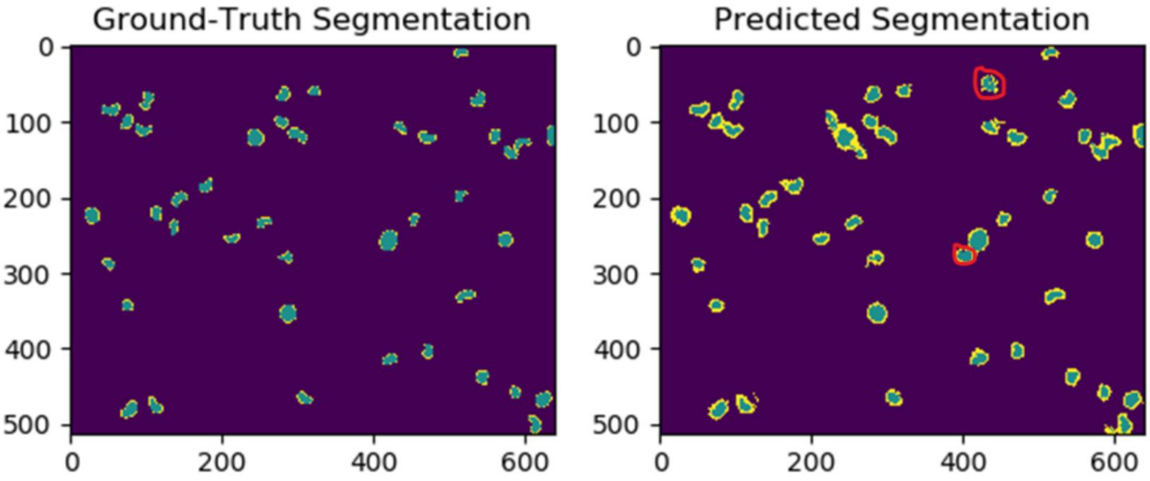


Figure 23: cells that are not present in the Ground-Truth images

This issue decrease the IOU score we measured, but the realistic results are very satisfying, and we believe that the score of our network we improve for better Ground-Truth segmentations.

From the above measurements table, we can also conclude that the SegMeasure score is higher than the IOU for all the experiments. We assume that the reason for that is the fact that oppose to the IOU score, the SegMeasure does not punish for False-Positive detections (which occurred because our network detected cell that where not present in the Ground-Truth segmentation images).

From the 3td experiment we conclude that for the vanilla architecture, the size of the Batch has negligible effect on the accuracy of the model. In the other hand, from Figs.14-19 we conclude that the model converges faster as the size of the batch goes smaller (a well-known attribute in Deep Learning).

Conclusion:

In this work we present a compressed version of the U-Net of biological cells' segmentation task. The sophisticated weight map and architecture of the U-Net makes it very suitable for instance/semantic segmentation tasks. In our work we performed a gradual training method which calls "Curriculum Learning" developed by **Yoshua Bengio** et al. this training method (which is closer to human learning process) increase the complexity of the input data further down the training, and besides increasing the accuracy of the model, helps avoiding Overfitting. We implemented the CL using the Elastic Augmentation method. Our experiment focused on the effects of the CL process as well as the effects of the different batch sizes (which seems to be negligible of vanilla U-Net). Future work will be to combine those two experiments and test the effects of the Batch sizes under CL process. An interesting experiment is to test also different kinds of dropout methods (standard, Gaussian, additive) on the U-Net (those experiments where left out of this work to allow extensive discussion about the presented topics).

References:

[1] **U-Net: Convolutional Networks for Biomedical Image Segmentation**, Olaf Ronneberger, Philipp Fischer, and Thomas Brox. Computer Science Department and BIOSS Centre for Biological Signaling Studies, University of Freiburg, Germany.2015

[2] **Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis**. Patrice Y. Simard, Dave Steinkraus, John C. Platt. Microsoft Research, One Microsoft Way, Redmond WA 98052. 2003.

[3] **Curriculum Learning**. Yoshua Bengio et al. U. Montreal, P.O. Box 6128, Montreal, Canada A2iA SA, 40bis Fabert, Paris, France. 2009.

[4] our code is available on GitHub: https://github.com/dorlivne/Segmentation