MASTER THESIS

Bc. Martin Studna

Neural Cell Segmentation from Fluorescent Microscopy Images of Mouse Brains

Department of Software and Computer Science Education

Supervisor of the master thesis:	doc. RNDr. Elena Šikudová PhD.
Study programme:	Computer Science
Study branch:	Artificial Intelligence

Prague 2022

I want to express my gratitude to my supervisor, doc. RNDr. Elena Šikudová PhD. for guidance and valuable advices. Furthermore, I would also like to thank everyone who gave me feedback on my thesis. Special thanks go to Mgr. Petra Nováková that was willing to consult the syntactic and semantic part of the thesis with me.

Title: Neural Cell Segmentation from Fluorescent Microscopy Images of Mouse Brains

Author: Bc. Martin Studna

Department: Department of Software and Computer Science Education

Supervisor: doc. RNDr. Elena Šikudová PhD., Department of Software and Computer Science Education

Abstract: Our objective is to propose a neural cell segmentation algorithm for fluorescent microscopic images of mice brains. We received a dataset from the Laboratory of Neurochemistry, Institute of Physiology of the Czech Academy of Sciences in Prague. We conducted various image segmentation experiments to identify a method that can most accurately segment neural cells. Our thesis will present the challenges linked with the segmentation of biological images. Then, we will describe the details of our experiments and the evaluation metrics used for measuring our methods' accuracy.

Keywords: Biomedical image segmentation, c-Fos, Neural cells, Machine learning, Deep learning

Contents

1. Introduction

The development of automatic detection and segmentation methods for cells and nuclei in microscope images plays a significant role in biological and medical applications. Performing such tasks manually is usually rather time-consuming, tedious, error-prone and slow. These methods allow researchers in the biomedical field to concentrate primarily on the mentally stimulating, creative and exciting part of any research. Nevertheless, the time of experts in medicine and biology fields is usually precious; having these automatic methods would help them make their work more efficient.

Recently, there has been a massive advance in computer vision approaches in biomedical fields. Most likely, two things have mainly influenced this situation: The introduction of Convolutional Neural Networks and especially the U-Net architecture that turned out to have very convincing results in image analysis. The increasing willingness of hospitals and clinics to provide X-ray radiography, computed tomography (CT), and magnetic resonance imaging (MRI) data. Open-source medical datasets are still rare since patient records usually remain private.[1] In order to gain access to any patient's data, the patient must give their approval for their medical or any other personal records to be used for research purposes. The data are scattered among different clinics and produced under specific protocols and different conditions.

The analysis of biomedical image data remains one of the most challenging tasks in computer vision. The reason is mainly due to various sources of ambiguity such as cellular and structural diversity, heterogenous staining conditions and challenging image acquisition processes. Griebel et al. [2021a] In Ronneberger et al. [2015] presented U-Net architecture that rapidly outperformed other methods for biomedical image segmentation tasks. U-Net's advantage is that the model is quite fast and does not necessarily require large amounts of data samples to produce satisfactory results. In the last decade, several other variants of U-Net have been proposed. For instance, in Oktay et al. [2018b] designed the attention U-Net that is able to learn to focus only on important regions and suppress areas that are irrelevant. Despite the fact that U-Net and Attention U-Net achieve great results in many cases, more complex architectures and deeper neural networks begun to be designed. Although, one problem arisen that is referred to as the vanishing gradient. As the gradients are propagated through the network, they tend to get smaller and smaller. Due to the decreasing size of the gradients, the weight change is insignificant, and the network fails in further training. To address this problem, the concept that is called residual learning is usually employed in networks. In Zhang et al. [2017] designed the U-Net model with residual learning called ResUNet. In Jha et al. [2021] introduced a modified version of ResUNet called ResUNet++ that also uses the squeeze and excitation block, the attention block and Atrous Spatial Pyramidal Pooling.

For our work, we received a dataset from the Laboratory of Neurochemistry,

[1]Patient records as well as any and all personal information in general cannot be made available just to anyone due to doctor patient confidentiality and also GDPR in the case of Europe. The situation is slightly different in Asia and the Americas as they have not ratified anything even remotely similar to the GDPR.

Institute of Physiology of the Czech Academy of Sciences, Prague. The dataset contains fluorescent microscopy images of mice brains, where we can see neural cells in great detail. Our task is to study and evaluate current image segmentation methods especially used for biomedical purposes and to report the results achieved with each algorithm to identify the most accurate method. The methods will also be evaluated on other two datasets - Deepflash2 and DataScienceBowl2018. Deepflash2 dataset contains c-Fos mRNA images that are very similar to the images we received from the Academy. DataScienceBowl2018 dataset contains not just c-Fos mRNA images but also other images from the biomedicine field. We will first discuss the traditional image segmentation methods based on thresholding, edge detection and light distribution. Then, we will present machine learning models such as Random forests, Support vector machines, and Perceptron or Logistic regression. Finally, we will mention the latest deep learning architectures used for medical image segmentation. In the theoretical section, we will describe methods and algorithms used in our experimentla

2. Theoretical background

2.1 Traditional techniques

Traditional image segmentation methods were mainly backed by the theoretical foundation of Digital Image Processing. The methods were based on concepts such as thresholding, region-growing, edge-based, watershed, or clustering methods. In this section, we will cover primarily techniques that employ thresholding, clustering and watershed segmentation.

2.1.1 Thresholding

Thresholding is one of the most elementary methods for image segmentation. It is time efficient and also not very computationally demanding. Thresholding is a transformation that maps the input image $f(i,j)$ into the output image $g(i,j)$ such that:

$$g(x,y) = \begin{cases} 1, & \text{if } f(x,y) \geq T \\ 0, & \text{if } f(x,y) \leq T \end{cases} \tag{2.1}$$

Essentially, thresholding is a technique that partitions images directly into regions based on intensity values and/or properties of these values. Gonzalez and Woods [2018] Let us imagine that we have an image composed of light objects on dark background in such a way that both the object and background pixels have intensity values grouped into two dominant modes. We can select such a threshold T that its value can be located between these two modes. There are three different strategies: Global, Local and Adaptive thresholding. Global thresholding selects just one threshold and applies it to all the pixels of an image. The problem is that the images usually do not have a bimodal histogram. A bimodal histogram is a histogram where the object of interest and background pixels have intensity levels grouped into two dominant modes. While global thresholding is fast and accurate for images with bimodal histograms, we need an algorithm that estimates the threshold also for images with more diverse histograms. The following iterative algorithm can be used for this purpose:

1. Select an initial estimate for the global threshold, T.
2. Segment the image using T. This will create two groups of pixels: G_1 consisting of pixels with intensity values $\geq T$; and G_2, consisting of pixels with values $\leq T$.
3. Compute the average (mean) intensity values m_1 and m_2 for the pixels in G_1 and G_2, respectively.
4. Compute a new threshold value midway between m_1 and m_2:

$$T = \frac{1}{2}(m_1 + m_2) \tag{2.2}$$

5. Repeat Steps 2 through 4 until the difference between values of T in successive iterations is smaller than a predefined value ΔT

Local thresholding partitions an image into separate regions and chooses a threshold for each segment. This approach partially addresses the problem of images where the background and foreground have similar intensity levels, and we cannot choose one threshold to separate them from each other. Thus, this method suffers from unexpected behavior on the borders between the regions. In contrast, in adaptive thresholding, we have an algorithm that automatically determines the threshold for a pixel in a small region around it. The algorithm usually derives the threshold from the lighting distribution of different image parts. Thresholding is usually not very sufficient for biomedical image data. This is mainly because biomedical images usually capture non-trivial biological material such as cells, vessels or organs that easily blend into the background or have similar intensities. Thus, there does not exist any suitable threshold that can segment the regions of interest. Although the thresholding is not much efficient in biomedical segmentation, better results can be achieved if we design a pipeline that employs morphological operators, edge detectors, noise reduction filters and other techniques. In the experimental part, we will show examples of how these pipelines can be designed to get some satisfactory results.

2.1.2 Clustering

Clustering algorithms aim to find groupings in data clusters that share a common property. Essentially, data in the same group share a common property, or they have multiple similar attributes. We can divide clustering algorithms into two groups: Partitioning and Hierarchical. In Hierarchical methods, clusters are formed from previous clusters in a bottom-up or top-down manner. In the bottom-up approach, each data point starts within its cluster, whereas in the top-down direction, all data begins in one cluster. One of the most popular clustering algorithms is a s algorithm. In k-means clustering, each data point is assigned to the cluster with the nearest mean, and each mean is referred to as a prototype of its cluster. The k-means algorithm is an iterative method that recomputes centroids of clusters until the convergence criterion is met.

Let us consider a set of observations $\{x_1, x_2, ..., x_Q\}$ where each observation is a d-dimensional vector. In image segmentation, these observations are represented by numerical values of pixels. For instance, if we have a RGB image, each observation is a 3-D vector, where each component is the intensity value of one of the three primary colors. In the case of grayscale images, the segmentation is based on just one intensity value. The goal of k-means clustering is to partition the set Q of observations into k disjoint cluster sets $C = \{C_1, C_2, ..., C_k\}$ in order to satisfy the following criterion of optimality:

$$\arg\min_{C}(\sum_{i=1}^{k}\sum_{z\in C_i}||z-m_i||^2) \tag{2.3}$$

where m_i is the centroid of the samples in set C_i and $||\cdot||$ is the vector norm. Usually, the Euclidean norm is used and then the term $||z-m_i||$ is a distance between a data point and a centroid. Unfortunately, k-means clustering is an

NP-hard optimization problem. Although, there are number of heuristics that improve the algorithm by finding approximations to the minimum. The algorithm is described in the following steps:

1. Initialize the set of centroids $m_i,\ i = 1, 2, ..., k$
2. Assign data samples to clusters: Each sample is assigned to the cluster whose centroid is the closest:
   $$x_q \rightarrow C_i \text{ if } ||x_q - m_i||^2 \leq ||x_q - m_j||^2 \ j = 1, 2, ..., k \ (j \neq i); \ q = 1, 2, ..., Q$$
3. Update the cluster centroids:
   $$m_i = \frac{1}{|C_i|} \sum_{z \in C_i} z \quad i = 1, 2, ..., k$$
   where $|C_i|$ us the number of data points in cluster set C_i
4. Test whether the reassignment of centroids happened or not.

The algorithm's application will be further discussed in the experimental section. We will present the results of the algorithm and also describe the algorithm run details.

2.1.3 Watershed segmentation

Image segmentation techniques are usually based on three principal concepts: edge detection, thresholding or region growing. Each of these methods has its advantages, for instance, the execution speed in the case of global thresholding and disadvantages include the need for post-processing, such as edge linking, in edge-based segmentation. Watershed segmentation utilizes all of these concepts, and it is thus able to produce more stable results. The idea of a watershed is based on visualizing an image as a topographic surface. Each point of a surface has two spatial coordinates. The spatial coordinates corresponding to the pixel's coordinates in the image. Additionally, the pixel's intensity value embodies the surface's elevation. Usually, the highest intensities represent peaks, whereas the lowest intensity values embody regional minima. The watershed transformation creates dams and deconstructs an image into catchment basins. The resulting segmentation is created by the water that springs in valleys, and as it continues to rise, it floods the surrounding landscape. Another dam is built if water begins to overflow into another basin. The computation terminates when the whole surface is flooded. The final dams correspond to the desired segmentation result. In the experimental section, we go more into the implementation details of such a method.

2.1.4 Gabor Filter

The machine learning algorithms are usually hard to be trained solely on image intensity values. Meaningful features need to be derived from images to make these algorithms more efficient. Suitable image features can be global information of the image, such as the average, standard deviation of image intensity values or the output from edge-based filters. One of the suitable feature extractors is the Gabor filter. The Gabor filter is obtained by modulating a sinusoid with a Gaussian. It is used for texture analysis, which means that it analyzes whether there is any specific frequency content in the image in particular directions in a localized region around the point or area of analysis. To better understand the Gabor filter, we can look at the following image with the white circle. We can see that each Gabor filter, each with a different setting, detects the edges of the circle from a different angle.

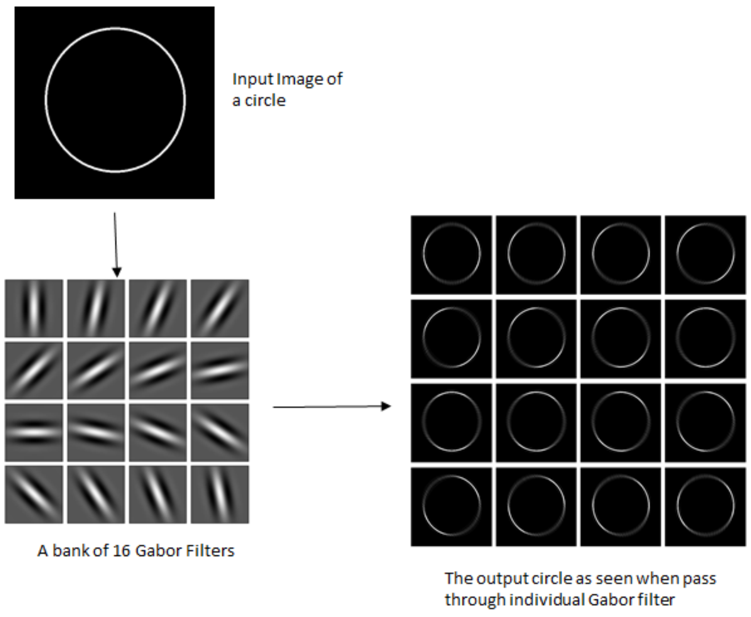

Input Image of a circle
A bank of 16 Gabor Filters
The output circle as seen when pass through individual Gabor filter

Figure 2.1: Gabor filter

Source: Shah [2018]

The filter has a real and an imaginary component representing orthogonal directions. Henriksen [2007] Both components together may be formed into a complex number or used separately. The individual components are defined below:

1. Complex
   $g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}) \exp(i(2\pi\frac{x'}{\lambda} + \psi))$
2. Real
   $g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}) \cos(2\pi\frac{x'}{\lambda} + \psi)$
3. Imaginary
   $g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}) \sin(2\pi\frac{x'}{\lambda} + \psi)$

We have described how is the Gabor filter derived and applied to the images. In the experimental section, we will show how is the filter used on our datasets and which parameter settings are employed.

2.2 Machine learning methods

Machine learning (ML) is the sub-discipline of Artificial intelligence (AI) which studies the properties of self-learning algorithms. The self-learning algorithms learn patterns in data and make decisions based on the knowledge they have gained. The ML models have various applications in computer vision, speech synthesis, robotics and etc. Usually, the data are split into three groups: training, validation and test samples. The training samples are used for fitting a model, which means the model parameters tuning to make accurate predictions. The validation samples are utilized to evaluate a model fit on the training samples. The test samples are used for the final evaluation of the trained model. In this section, we will describe several machine learning algorithms.

2.2.1 Perceptron

The perceptron is a linear model for binary classification problems. The idea of the Perceptron is to find a separating hyperplane by minimizing the distance of misclassified points to the decision boundary. Hastie et al. [2001] Let us define the target value $t \in \{-1, +1\}$, our objective is to find weights w such that for all train data the following condition holds:

$$t_i y(x_i; w) = t_i x_i^T w \geq 0 \tag{2.4}$$

The algorithm is defined in the following steps (Assuming that the train samples are linearly separable):

```
1: Initialize all weights $w$ to 0
2: Repeat, until all examples are classified correctly, continue with a next sample:
3:     $y = x_i^T w$
4:     **if** $t_i y \leq 0$ (sample si incorrectly classified) **then**
5:         $w = w + t_i x_i$
6:     **end if**
```

2.2.2 Logistic regression

The logistic regression models the probability that a sample x is classified into the class C_1 with respect to the weights w of the model. The probability is defined as:

$$\begin{aligned} p(C_1|x) &= \sigma(x^T w + b) \\ p(C_0|x) &= 1 - p(C_1|x) \end{aligned} \tag{2.5}$$

where σ is a sigmoid function s

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

The sigmoid function is employed in order to map any real predicted value into the range from 0 to 1. The logistic regression is fit by maximum likelihood estimation (MLE). The likelihood is defined as:

$$L(w) = p_{model}(X; w) = \prod_{i=1}^{N} p_{model}(x_i; w) \tag{2.7}$$

The Likelihood represents how likely the given random variable X will have value x with respect to the parameters w. MLE attempts to find such parameters w that maximise the likelihood function.

2.2.3 Support vector machines

The idea of Support Vector Machines (SVM) is to find such a hyperplane that separates two or n groups of data from each other and also has the largest margin from all of these groups. Let's assume we have a dataset $X \in R^{N \times D}, t \in \{-1, 1\}$, a feature φ and a model

$$y(x) = \varphi(x)^T w + b \tag{2.8}$$

where X is our dataset, t are our classes and b is the bias. If the output from the model is 1, we identify it with one class, whereas the output is -1, we identify the output as the second class. The distance between the decision boundary and a data point is $\frac{t_i y(x_i)}{||w||}$. Thus, the maximization of margin is formulated as:

$$\arg\max_{w,b} \frac{1}{||w||} \min_i [t_i(\varphi(x_i)^T w + b)] \tag{2.9}$$

To find the maximum distance between the data groups and the decision boundary, we need to find the closest point from each data group to our decision boundary. We can reformulate the above equation by the fact that the model is invariant to multiplying w and b by a constant. We know that the distance of a point from the decision boundary is equal to the model's prediction $t_i y(x_i)$ which is normalized by $||w||$. Then, if we multiply w by a constant then even the predictions are multiplied by this constant. Then, we can decide that all points will have the distance $t_i y(x) \geq 1$, which help us to redefine the maximization problem as:

$$\arg\min_{w,b} \frac{1}{2} ||w||^2 \text{ given that } t_i y(x_i) \geq 1 \tag{2.10}$$

When we were talking about Perceptron, we mentioned that Perceptron is not able to classify data which are not linearly separable. SVM is another linear model but it can address this problem with two approaches. Either we can transform the data with a method that is called the kernel trick, which maps the data into higher dimensional space, where the data are linearly separable, or we can introduce soft margin SVM. We will not talk more about kernel tricks because they will not be used in the experimental section and we will rather describe more in details the soft margin SVM. The soft margin SVM allows misclassifications. Therefore, we introduce the cost function that penalizes the misclassified samples. The cost function is called Hinge loss that is defined as:

$$L_{hinge}(t, y) = \max(0, 1 - ty) \tag{2.11}$$

Using the Hinge loss function, we need to incorporate the function into the optimization problem that SVM is solving. The problem is then reformulated as:

$$\arg\min_{w,b} C \sum_i L_{hinge}(t_i, y(x_i)) + \frac{1}{2}||w||^2 \tag{2.12}$$

The loss function has a regularizing effect on the model. The model also introduces the parameter C that controls this regularization. The C constant represents how much do we want to tolerate the misclassifications.

2.2.4 Random forests

Another machine learning algorithm is Random Forest, which is a part of the so-called Ensemble methods. Ensemble learning aims to build a prediction model by combining the strengths of a collection of simpler base models. Hastie et al. [2001] The essence of the method is to build multiple trees in randomly selected subspaces of feature space. Trees in different subspaces generalize their classification in complementary ways, and their combined classification can be monotonically improved. Ho [1995] Randomly selected subspaces are created with a method called bootstrapping. We have N data samples, and we select N samples with replacement, namely, samples are selected more than once or some of them are not picked at all. The regression is performed by averaging the predictions from all trees, while the classification is made by majority voting. The majority voting means that the model predicts the class which received the majority of votes from all estimators.

2.3 Deep learning methods

2.3.1 Convolutional neural networks

An image is stored in a computer as a matrix of numbers. However, these numbers do not tell us anything about an object of interest drawn in an image. These values also do not provide information about a shape of an object, its characteristics or any context about an image. In previous sections, we have described elementary techniques that distinguish the object of interest and the background by the intensity levels of the image. We want to come up with a more sophisticated approach. What if we had a self-learning algorithm that could derive the segmentation mask by looking at features that lie in the image? This is where the neural networks come in. The essential type of neural network is a feed-forward network (FFN). In image processing, each pixel value of an image is passed into the network. The network is trying to learn the intensity values of each pixel in the image. This approach has a significant disadvantage. With larger images, we have to train more robust networks. Another problem of FFN is that they react differently to input and its shifted version. This implies that FFN is not translation invariant. A more advanced approach to visual processing is Convolutional Neural Networks (CNN). CNN can extract features by applying image filters to the input. In such a manner, we can capture the intrinsic structure of features because kernels that go through images can calculate local dependencies between pixels.

2.3.2 U-Net

The U-Net architecture was designed mainly for the biomedicine field. It can achieve good results even with a small portion of training data. The architecture consists of two paths. The first one is called the Encoder, also called the contraction path, whose role is to capture the context of the image. The context is taken by image downsampling. The downsampling is done by consequently applying convolutional filters on the image. The second part is the Decoder, whose role is the image reconstruction into original resolution. The image reconstruction is achieved by consequent image upsampling. The Encoder contains convolutional and max pooling layers, whereas the Decoder consists of transposed convolutional layers and max pooling layers. One of the most significant distinctions from other CNN networks is that U-Net is a fully convolutional neural network. It does not contain any dense layers.

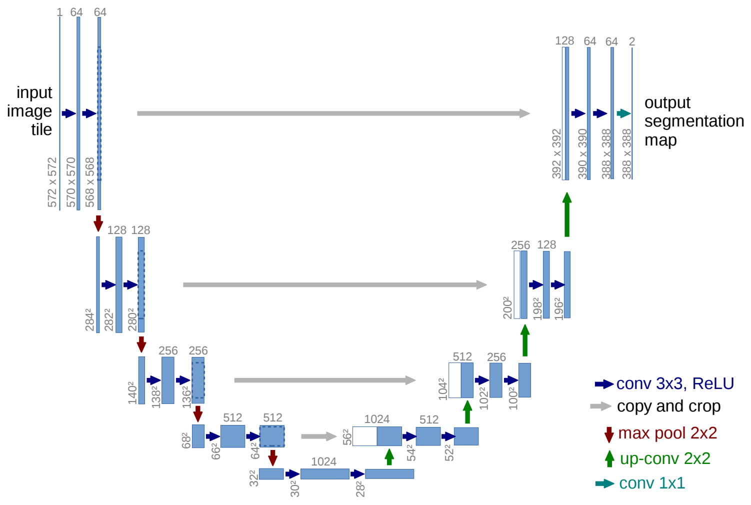


Figure 2.2: U-Net architecture

Source: Ronneberger et al. [2015]

2.3.3 Anatomically constraint neural networks

Another substantial deep learning architecture in biomedicine is the Anatomically Constrained Neural Network. ACNN was proposed by Ronneberger et al. [2015] and it is a generic method that improves the accuracy of convolutional networks by incorporating prior knowledge about organ shape and location to improve the performance of image analysis approaches. Most classification and regression models utilise a pixel-level loss function (e.g. cross-entropy or mean square error) that does not fully account for the underlying semantic information and dependencies in the output space.

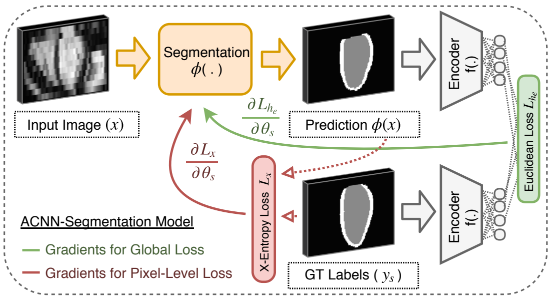


Figure 2.3: ACNN

Source: Oktay et al. [2018a]

The concept of ACNN is that we take an arbitrary convolutional neural network, but in addition, we encode the output of the network and the ground-truth mask using the so-called Encoder, which is nothing more than the first part of the U-Net, that we described more in details in the U-Net section. Finally, we compute the so-called shape regularisation loss, which is nothing more than the Euclidean distance between the encoded prediction and the ground-truth mask. The shape regularisation loss is defined below:

$$\begin{aligned} L_{h_e} &= ||f(\phi(x);\theta_f) - f(y;\theta_f)||_2^2 \\ &= \min_{\theta_s}\left(L_x(\phi(x,\theta_s),y) + \lambda_1 \cdot L_{h_e} + \tfrac{\lambda_2}{2}||w||_2^2\right) \end{aligned} \tag{2.13}$$

where L_x is cross-entropy loss, $\phi(x)$ is the U-Net prediction, θ_s correspond to training parameters of U-Net network, f is the Encoder, θ_f denotes the parameters of the Encoder, y is the ground-truth mask, λ_1, λ_2 determines the weights of shape regularisation loss and weight decay terms used in the training. Ronneberger et al. [2015]

2.3.4 Attention U-Net

One of the U-Net weaknesses is a relatively imprecise image reconstruction during upsampling. To address this problem, skip connections combine information from the downsampling path with the upsampling path. However, this brings many redundant low-level feature extractions, as feature representation is poor in the initial layers. Soft attention implemented at the skip connections will actively suppress activations in irrelevant regions, reducing the number of redundant features brought across. Attention, as its name suggests, is a method for emphasizing only the important regions in the image.

Hard attention

Hard attention emphasizes relevant regions by cropping the image or iterative region proposal. Since hard attention can only choose one region of an image at a time, it has two implications, it is non-differentiable and requires reinforcement learning. Since it is non-differentiable, the network can either pay attention or not to a given region in an image. Therefore, standard backpropagation cannot be used, and Monte Carlo sampling is needed to calculate the accuracy across various stages of backpropagation. Considering the accuracy is subject to how well the sampling is done. Oktay et al. [2018b]

Soft attention

Soft attention assigns weights to distinct parts of the image. Areas that are more relevant have larger weights and areas of low relevance have low weights. As the model is trained, more focus is given to the regions with larger weights. Due to the deterministic nature of soft attention, it remains differentiable and can be trained with standard backpropagation. Oktay et al. [2018b]

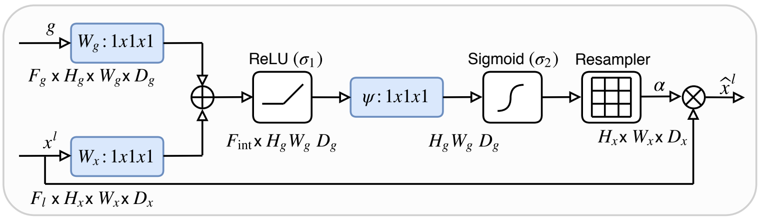


Figure 2.4: Attention gate

Source: Oktay et al. [2018b]

2.3.5 Residual U-Net

Neural networks with many layers trained with backpropagation algorithm tend to suffer from the vanishing gradient problem. Vanishing gradient occurs due to low value of gradient which causes very slight or negligible improvement in weights. Additionally, if we use sigmoid as the activation function for a hidden layer, gradients start to vanish even more rapidly due to the small derivative of the sigmoid function. In the backpropagation algorithm, the gradients tend to get smaller as we get closer to the input layer because small derivatives are multiplied with each other. There are two possible solutions how to address this issue. The first one is the ReLU function, which does not have small derivatives. The second solution are Residual networks. Residual networks implement the identity connection or skip connection that adds the original input value to the value that went through the activation function. He et al. [2015]

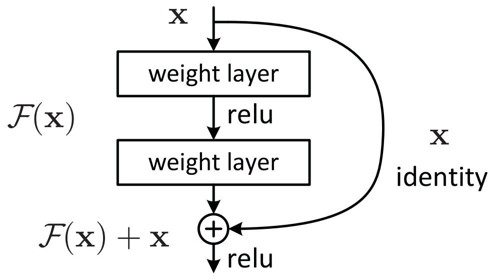


Figure 2.5: Residual learning: a building block

Source: He et al. [2015]

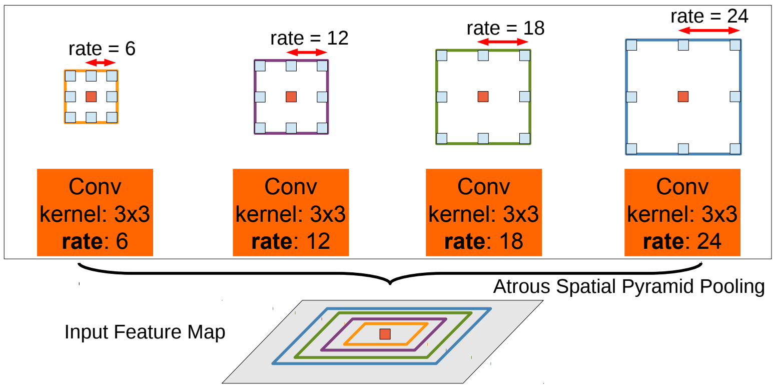
Zhang et al. [2017] address the problem of vanishing gradient in U-Net architecture by combining it with residual learning.

2.3.6 ResUNet++

In 2019 ResUNet++ was proposed by Jha et al. [2019] which is an improved ResUnet architecture for colonoscopic image segmentation. It implements residual connections such as the ResNet model but also introduces squeeze and excitation blocks, Atrous Spatial Pyramidal Pooling and attention blocks, which we talked about in the Attention U-Net section.

Atrous spatial pyramidal pooling

ASPP is based on the idea of spatial pyramidal pooling that enables deep networks to accept an image of any size. Moreover, it uses atrous convolution. Atrous convolution enables to control at which resolution are the feature responses computed within the deep convolutional neural networks. Chen et al. [2018] It also allows for the extension of the field of view of filters without increasing the number of parameters. It introduces a new parameter for the convolution operation called the dilation rate. It defines the distance between the elements in a kernel. ASPP addresses the fact that an object can exists in multiple scales. This can be simply solved by presenting rescaled versions of the same image to the CNN and then aggregating the feature or score maps. Papandreou et al. [2015], Chen et al. [2015], Kokkinos [2016] In Chen et al. [2018] shows that this approach definitely increases the performance of the network, but the feature responses need to be computed in each CNN layer for multiple scaled versions of the input image. Instead of subsequent resampling features, ASPP consists of multiple parallel atrous convolutional layers with different sampling rates that implement this mapping. Parallel branches produce the final result by bilinearly interpolating all feature maps from the layers together by taking the maximum responses across the different scales.



rate = 6
rate = 12
rate = 18
rate = 24
Conv kernel: 3x3 rate: 6
Conv kernel: 3x3 rate: 12
Conv kernel: 3x3 rate: 18
Conv kernel: 3x3 rate: 24
Atrous Spatial Pyramid Pooling
Input Feature Map

Figure 2.6: ASPP - Atrous spatial pyramidal pooling

Source: Jha et al. [2019]

Squeeze and Excitation block

Squeeze and excitation block emphasize informative channels and suppress less useful ones in order to learn to use global information. Hu et al. [2017b] The input is first passed through a squeeze operation that returns the average of every channel. The squeeze operation is followed by the excitation block that computes a weight for every channel using a sigmoid activation function, and each channel is then multiplied with its corresponding weight. An additional small hidden layer with $C/16$ neurons prevents the increase of parameters too much.

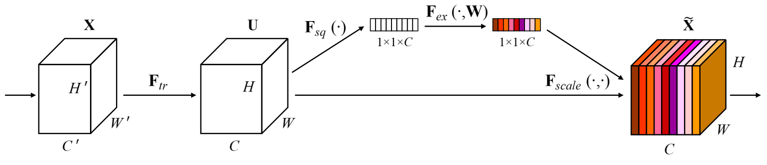


Figure 2.7: Squeeze and excitation block

Source: Hu et al. [2017a]

3. Experiments

The Laboratory of Neurochemistry from the Institute of Physiology of the Czech Academy of Sciences conducts experiments on mice to observe their cognitive functions during several behavioral tests such as nest building, locomotion in the open field, evaluation of food anticipation and etc. The tests are described more in detail in Abbondanza et al. [2022]. As part of standart testing procedure, the mice are also injected with specific stimulants, such as amphetamine (AMPH) or saline, in order to monitor any related behavioral changes, increase or decrease in brain activity or any potential motor impairments. Upon the completion of the testing phase, the mice are terminated and their brains extracted to be disected for further study. Each slice is processed with double-probe FISH procedure. FISH stands for Fluorescent in situ hybridization, a cytogenetic technique that uses fluorescent DNA probes to target specific chromosomal locations within the nucleus, resulting in coloured signals that can be detected using a confocal microscope. To study the neuronal activity of mice brains, the researchers observe c-Fos mRNA that is visualized by the FISH technique.

The Laboratory provided data that contains c-Fos mRNA expression images. Our task is to design an automative method that can segment neural cells in images as efficiently as possible. Unfortunately, the data, provided by the research team, lacked the standard relevant annotations which are necessary for the development of such a method, and also for the comparison between the results and the annotations. The only option that we had, was to create ground-truth labels by ourselves. The annotations were created in the application called ImageJ Fiji - Schindelin et al. [2012]. Using the application, we can label one of the areas that are significant for us and the application is then able to find and segment similar regions. It is a semi-automatic segmentation tool to create ground-truth masks. In order to evaluate the performance of our methods, we found datasets that also contain c-Fos mRNA expression images or other data used in biology or medicine research, and also the annotation made by experts from the biomedicine fields. Thus, the first dataset is DataScienceBowl2018 - HubMAP [2018] which was provided publicly at the Data Science Bowl 2018 contest. The goal of the contest was to develop an algorithm to automate nucleus detection. The dataset contains 670 images with annotations. The images were acquired under a variety of conditions and vary in the cell type, magnification, and imaging modality. The Deepflash2 dataset contains only c-Fos mRNA images, which were acquired from Griebel et al. [2021b]. Unfortunately, the dataset contains only 36 images, but the images are the most similar to the images that are in CAS dataset. In the following experiments, we are going to evaluate our methods on CAS dataset and datasets with expert annotations. In order to reproduce our experiments, we will include Python implementations of our methods, models and scripts for evaluation.

Since the training of machine learning and deep learning models usually requires to use high-performance hardware with a good GPU, we used the MetaCentrum servers. MetaCentrum is a virtual organization that manages distributed computing infrastructure consisting of computing and storage resources owned by CESNET as well as resources of co-operative academic centres within the

Czech Republic. MetaVO For our experiments we requested the cluster Adan that has the following technical specs: 32 CPUs Intel Xeon Gold 5218 2.3 GHz, 2 GPUs nVidia Tesla T4 16GB. In order to evaluate our methods, we use three widely adopted segmentation metrics, i.e, the precision, Intersection over Union, Mean Absolute Error and Dice score. We also introduce a metric that is not so frequently used and this is Enhanced-alignment measure.

The precision is nothing more than the ratio of the number of pixels that correspond to the correct pixel value in the ground truth mask and the total number of pixels in an image. The Intersection over Union (IoU) metric, also referred to as the Jaccard index, is the percentage of overlapping pixels between the ground-truth mask and the prediction output. The metric ranges from 0 to 1, where 1 signifies perfectly overlapping segmentation. IoU is defined as the intersection between the prediction result and the mask divided by the total number across both masks:

$$\mathrm{IoU} = \frac{X \cap Y}{X \cup Y} \tag{3.1}$$

The Mean Absolute Error measures the pixel-wise error between the ground-truth mask and the prediction output, which is defined as:

$$\mathrm{MAE} = \frac{1}{w \times h} \sum_{x}^{w} \sum_{y}^{h} |P(x,y) - G(x,y)|, \tag{3.2}$$

where P denotes the prediction result, G the ground-truth mask (x, y) denotes the coordinate of each pixel in G. Symbol ϕ is the enhanced alignment matrix.

The dice score measures the similarity between two samples. It was developed by Thorvald Sorensen. Sorenson [1948] It is defined as:

$$\mathrm{Dice} = \frac{2|X \cap Y|}{|X| + |Y|} \tag{3.3}$$

The Dice score is very similar to IoU, and it also ranges from 0 to 1, where 1 means the greatest similarity between two samples. Also, IoU and the Dice score are positively correlated with each other. This means that if one formula says that one algorithm is better than the other one, the other formula will say the same.

The enhanced-alignment measure (EAM) or E-measure has been recently proposed metric by Fan et al. [2018] for evaluating the local and global similarity between two binary maps. It is defined as:

$$E_\phi = \frac{1}{w \times h} \sum_{x}^{w} \sum_{y}^{h} \phi(P(x,y), G(x,y)), \tag{3.4}$$

where ϕ is the enhanced alignment matrix. Cognitive studies have proven that human vision is highly sensitive to both global information and local details in a scene. Fan et al. [2018] EAM incorporates global image statistics such as image-level mean into the metric but also uses local pixel values. The enhanced alignment matrix ϕ is defined as

$$\phi = f(\xi), \tag{3.5}$$

where f was selected in Fan et al. [2018] as:

$$f = \frac{1}{4}(1 + x)^2, \tag{3.6}$$

because the function worked best in their experiments. The alignment matrix ξ is defined as:

$$\xi = \frac{2\varphi_P \circ \varphi_G}{\varphi_G \circ \varphi_G + \varphi_P \circ \varphi_P}, \tag{3.7}$$

where $\circ$ denotes the Hadamard product (element-wise multiplication), φ_G is the bias matrix for the ground-truth mask, φ_P is the bias matrix for the prediction result. The bias matrix is defined as:

$$\varphi_I = I - \mu_I \cdot A, \tag{3.8}$$

where I is the input image, μ_I is the image-level mean, A is a matrix where all the element values are 1, and the size of A is the same as the size of I. The E-measure ranges from 0 to 1. The higher the score, the more accurate result was predicted.

3.1 Traditional techniques

3.1.1 Thresholding

In this part, we are going to present how is the thresholding efficient on biological images. One of the example images of the dataset can be found below in figure 3.1a

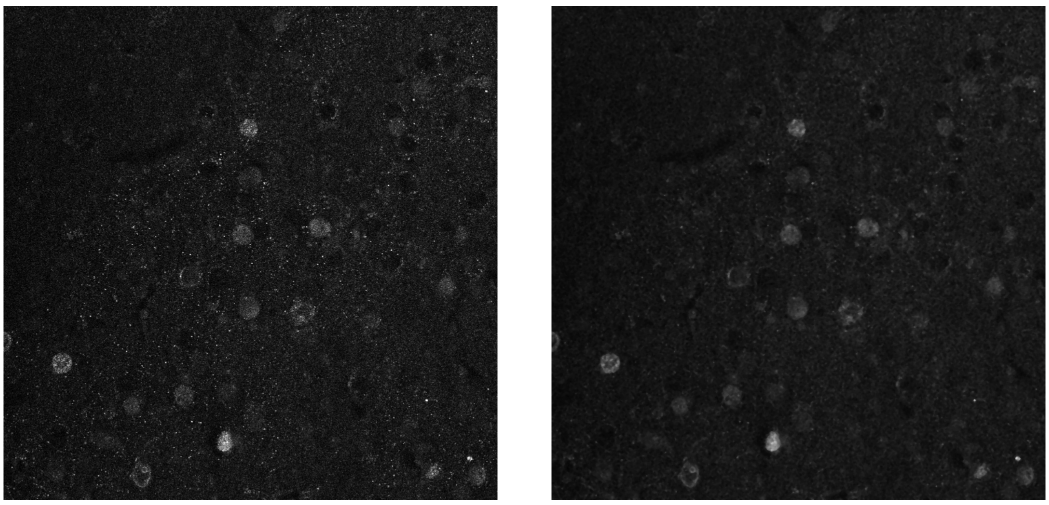

(a) CAS Image (b) Median filter

Figure 3.1: Median filtering

A lot of the images from the dataset contain noise in the form of sparsely occurring white pixels. This type of noise is referred to as Salt-and-pepper noise. Usually, this type of noise can be eliminated by Median filter. We have applied a Median filter with the kernel size equal to 3. You can see the result in figure 3.1b

After applying median blurring we used two thresholding strategies: manual threshold set to 70 and Otsu's algorithm. We can notice in figure 3.2 that automative approach does not seem to be efficient in this case. Otsu's algorithm struggle to omit noise that is in the background. This causes that the noise is linked with cells.

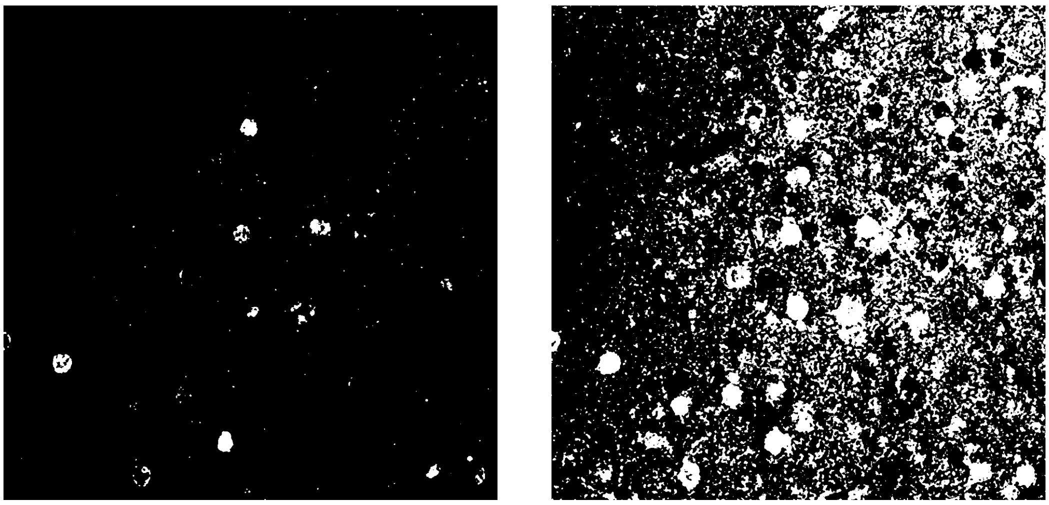
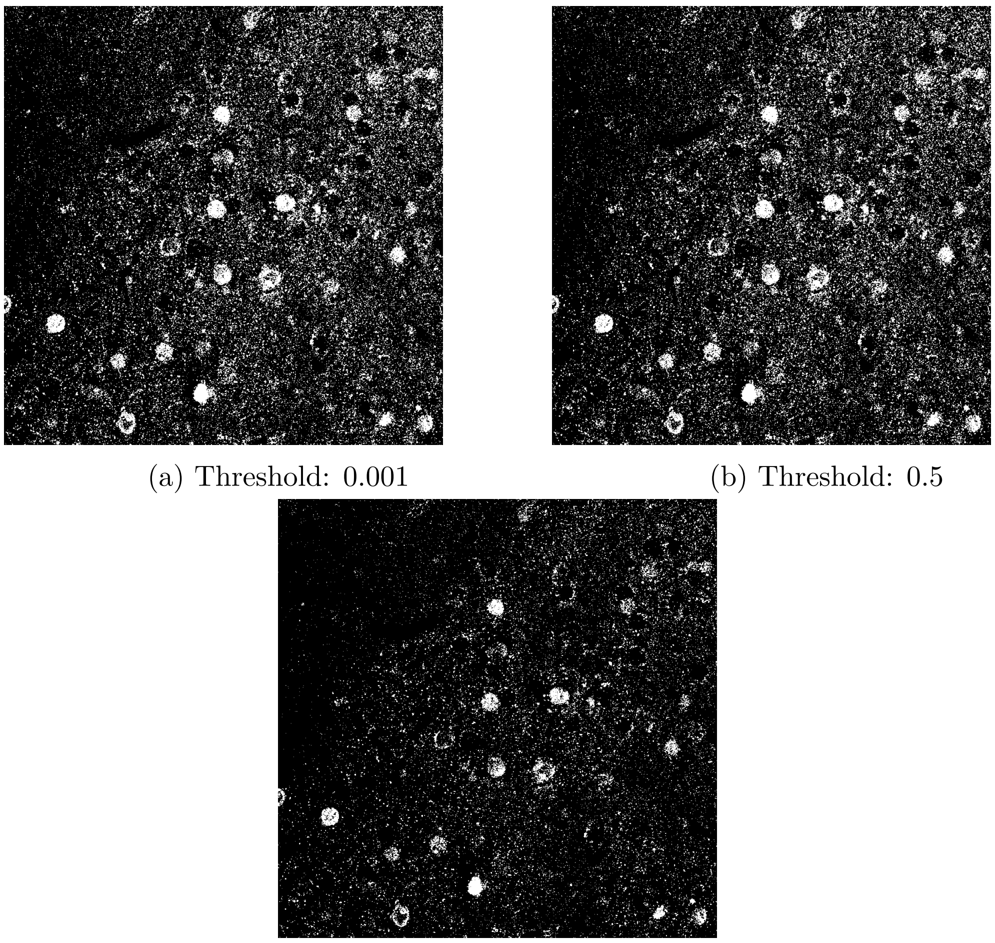

(a) Manual threshold set to 70 (b) Otsu's algorithm

Figure 3.2: Thresholding algorithms

3.1.2 Clustering

k-means clustering was performed on the image mentioned in the previous section. You can see the results in figure 3.3. We ran the algorithm with threshold equal to 0.001, 0.5 and 100. We can see that with a larger threshold, there are fewer pixels in the background misclassified as foreground. The algorithm was able to finish approximately after 15 epochs. As you can notice, the result contains again a lot of sparsely occurring white pixels. The disadvantage of the k-means clustering is that it always produces different results, and it is relatively slow compared to other algorithms, for instance, thresholding.

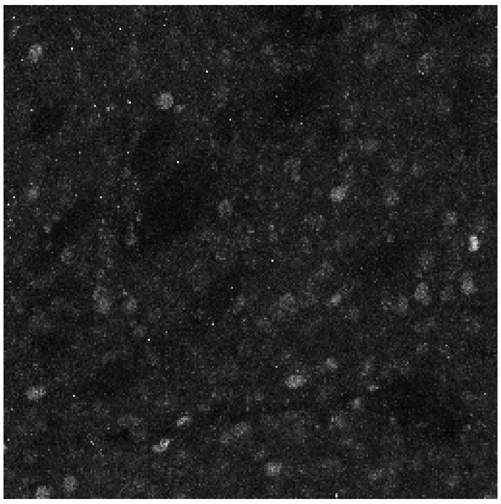

(a) Threshold: 0.001 (b) Threshold: 0.5

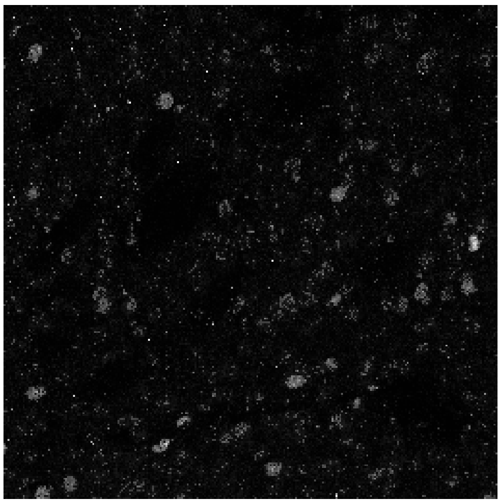
(c) Threshold: 100

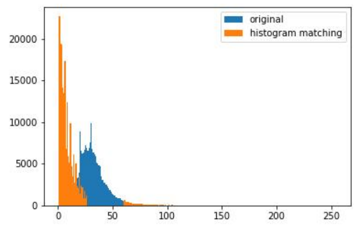
Figure 3.3: k-mean segmentation

3.1.3 Watershed segmentation

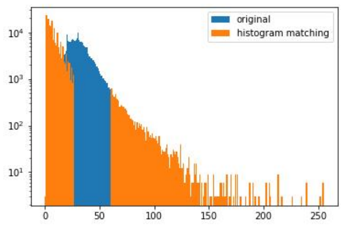
In this section, we are going to cover our experiment where we performed the watershed algorithm on our data. In order to improve the result of our algorithm, we made a few preprocessing steps. In our dataset, we struggle to select an appropriate threshold value that would distinguish objects of interest and the background. Several strategies can increase the intensity values of the foreground and lower the intensity values of the background. For instance, we could transform the histogram of the image to such a distribution that would satisfy these criteria.


(a) Input image


(b) Histogram result



(c) Histograms



(d) Logarithmic histograms

Figure 3.4: Histogram matching

By transforming the image's histogram into exponential distribution, we can yield to our image such properties. We can notice in figure 3.4. that dark tones got even darker, and the neural cells got lighter. If we look at the image that is the result of histogram matching, we can see that the image still contains white dots that are not part of the neural cells. In order to remove these dots, we use the non-local means algorithm. Non-local mean filtering takes a mean of all pixels in the image, weighted by how similar these pixels are to the target pixel. Buades et al. [2005] The result of the non-local means algorithm can be seen in figure 3.5.

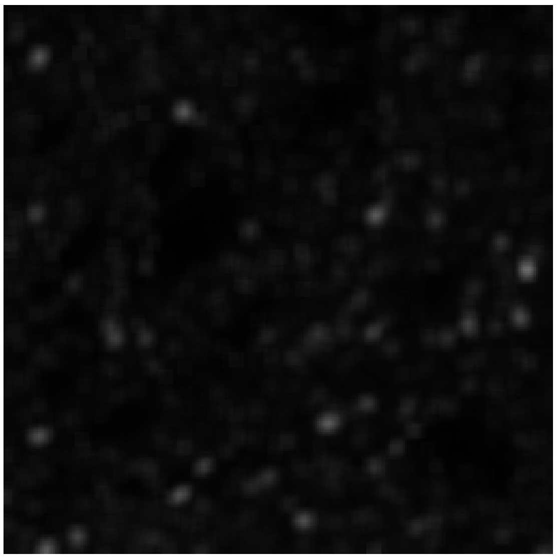
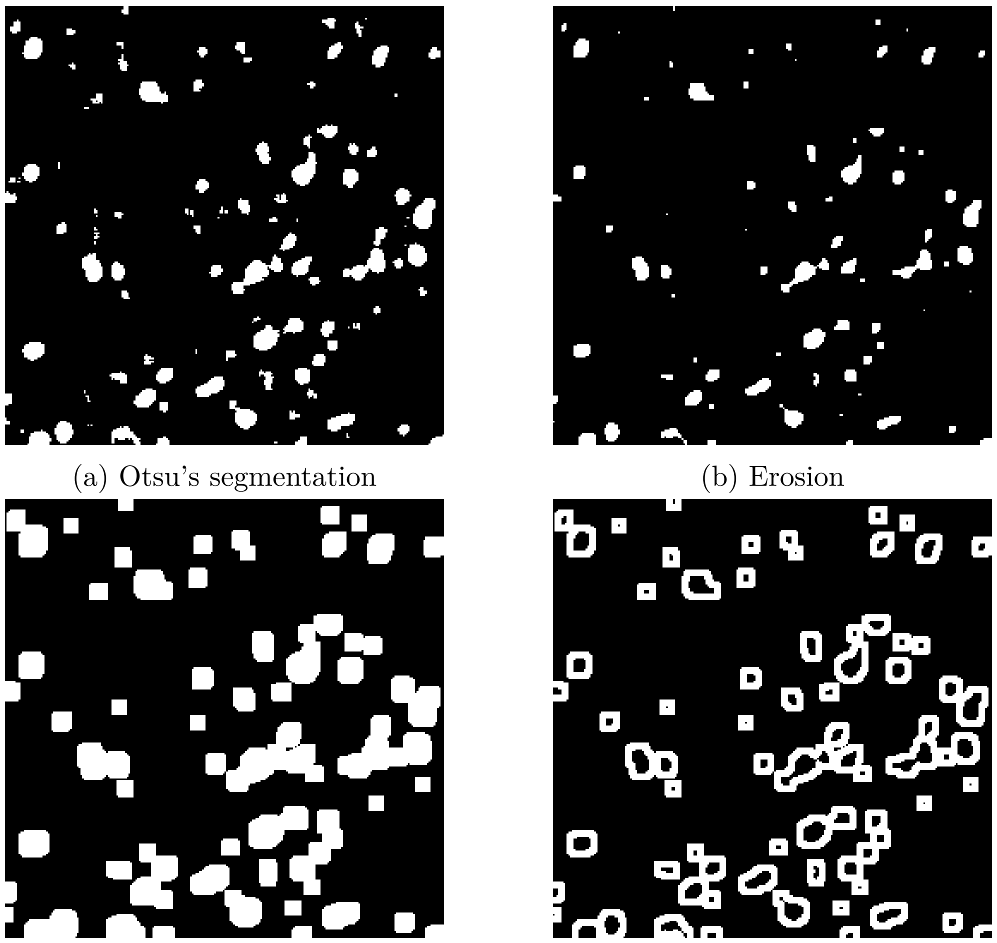


Figure 3.5: Non-local means

Now, we have our image denoised. We need to get somehow the segmentation mask and build barriers to the neural cells. We will use Otsu's algorithm again to get our mask. In order to get our barriers, we will apply erosion, dilation and, lastly, we will subtract the results of these two operations. Erosion removes boundary pixels. This way, we can get an area that is part of the foreground. If we dilate an image, we will also get the area that is not part of neural cells. When we subtract the results of erosion and dilation, we will get the boundaries of our cells. Finally, we can apply watershed. The results of Otsu's algorithm, Erosion, Dilation and Subtraction, can be found in figure 3.6. The final result of Watershed segmentation is in 3.7



(a) Otsu's segmentation (b) Erosion

(c) Dilation (d) Subtraction

Figure 3.6: Watershed preprocessing

We can notice that watershed algorithm is not well addressing the problem of overlapping cells. If there are multiple cells, the algorithm tends to merge local cells together.

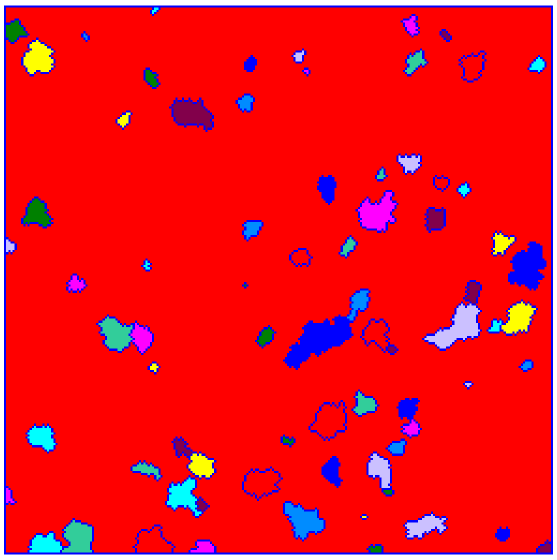


Figure 3.7: Watershed's result

3.2 Machine learning

In this section, we will describe the experiments that were conducted with machine learning models. For CAS and Deepflash2 datasets we decided to choose Random Forests and Support Vector Machines. Both of these models are quite different from each other, since each of them uses very distinct approach for training. We chose them because they are quite popular and frequently used for various image segmentation applications. Random Forests were set with 50 trees and SVM were using linear kernels.

For the DataScienceBowl2018 dataset we selected a slightly different approach and different type of models. The dataset is significantly bigger than the other two previously mentioned datasets, and it does not fit into our memory, which is in our case 16GB of RAM. To address this problem, our model needs to be trained on small portion of data that is able to fit into our memory. The partial fitting for linear models is usually done by Stochastic Gradient Descent (SGD). SGD is an iterative or incremental search for the model weights. It is either used when there is too much data, or the direct optimization is not feasible. SGD is a special type of Gradient Descent method, where we estimate gradients of our model weights using just a single random sample from the training data. Unfortunately, Random Forests cannot be trained incrementally by the SGD algorithm. There are variants of Random Forests that does not have to have all data in the memory. For instance, it is possible to train Random Forests in a distributed manner, where more computers are utilized. This seemed to us complicated so we decided that we will not use Random Forests on this dataset. For SGD training we used models such as Logistic regression, Perceptron, Support Vector Machines (SVM) using Hinge loss and Squared Hinge loss. Also, SVM were employed with linear kernels and l2 regularization.

Initially, the models were trained solely on pixel values of the images; however, the results were not sufficient. To achieve better precision, we prepared features that more easily explain the image's foreground and background and yield more information than pixel values. We generated 32 variants of the Gabor filter maps with different parameter settings. Then, we also created edge maps produced by various edge filters: Canny, Sobel, Robert, Prewitt and Scharr.

Lastly, each image was convolved with a Gaussian filter with sigma equal to 3 and 7, a Median and Variance filter with kernel size equal to 3. The filters extract essential information about neural cells, i.e. shape, structure, boundaries, etc. Respectively, we need all the elementary features to get these cells' segmentation masks.

Table 3.1: Machine learning models metrics

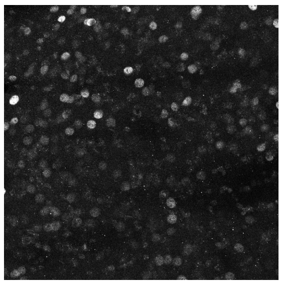
Dataset	Method	Mean Dice	Mean IoU	MAE
CAS	SVM - Squared Hinge l.	0.544	0.411	0.071
	RF	0.585	0.396	0.091
DSB2018	Log. r.	0.581	0.454	0.066
	Perceptron	0.389	0.268	0.121
	SVM - Hinge l.	0.338	0.235	0.121
	SVM - Squared Hinge l.	0.05	0.053	0.137
Deepflash2	SVM - Squared Hinge l.	0.375	0.249	0.004
	RF	0.512	0.352	0.004

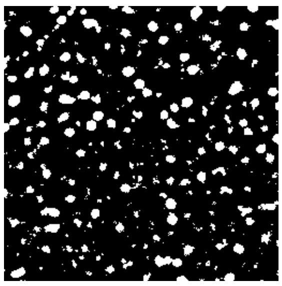
It can be seen from table 3.1 that Random Forests and SVM using Squared Hinge loss achieve both of them almost similar results. Although SVM has slightly higher accuracy, overall, it has better results than Random Forests. The other notable difference between these two models is their size. The SVM model takes approximately 2 KB of memory, whereas the size of the Random Forests model is 4 Gb. Lastly, the SVM prediction time is in the matter of hundreds of milliseconds, while the prediction of Random Forests runs around 14 seconds. The Perceptron model has probably the worst results. Even though the model returns considerably good accuracy, it has the smallest dice score and IoU value of all models.

Table 3.2: Machine learning models metrics

Dataset	Method	Accuracy	E_ϕ
CAS	SVM - Squared Hinge l.	83.7%	0.031
	RF	78.84%	0.045
DSB2018	Log. r.	75.4%	0.743
	Perceptron	80.09%	0.389
	SVM - Hinge l.	81.9%	0.5
	SVM - Squared Hinge l.	83.4%	0.313
Deepflash2	SVM - Squared Hinge l.	99.5%	0.667
	RF	99.4%	0.8

In table 3.2 we can see that SVM accuracy is better than the accuracy of Random Forests. If we also look at the prediction in figure 3.8 and 3.9, we can see that in general, Random Forests tends to predict more foreground pixels incorrectly than the SVM algorithm.


(a) Input


(b) SVM - Squared Hinge loss

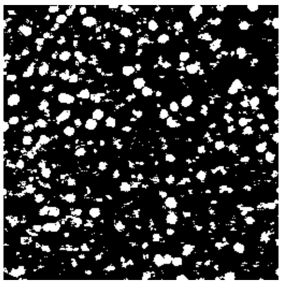
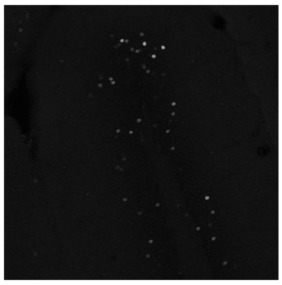
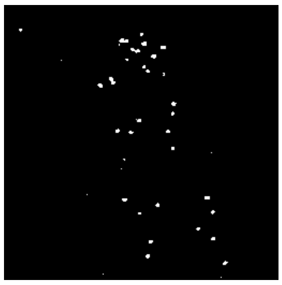

(c) Random Forests

Figure 3.8: ML CAS results


(a) Input


(b) SVM - Squared Hinge loss

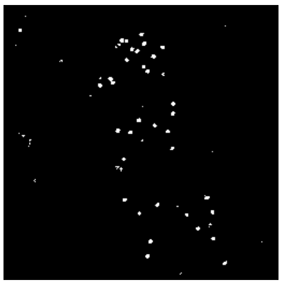
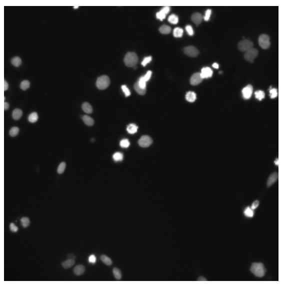
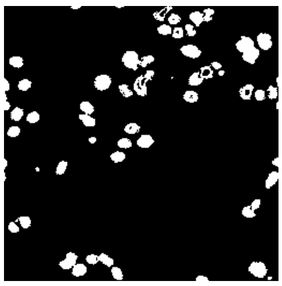
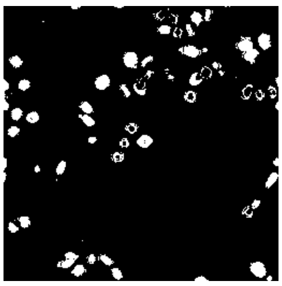
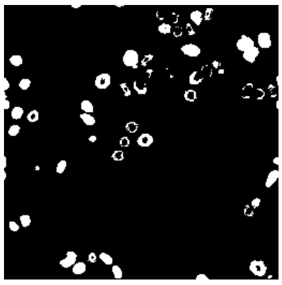

(c) Random Forests

Figure 3.9: ML deepflash2 results


(a) Input


(b) Perceptron


(c) SVM - Sq. Hinge


(d) Log r.

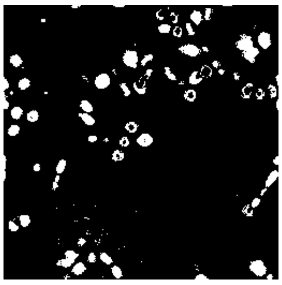
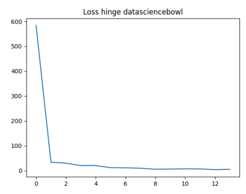

(e) SVM - Hinge

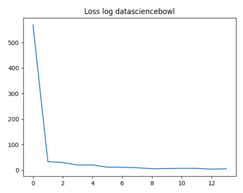
Figure 3.10: ML DataScienceBowl2018 results

If we look in figure 3.10, there are no significant differences between the predictions that each model produced. Only in the image 3.10c we can notice that

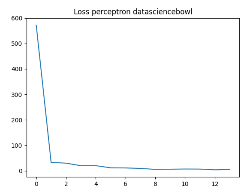
SVM using Squared Hinge loss tends to emit white dots in the dark background which are not part of the neural cells.



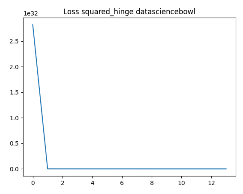
(a) SVM - Hinge loss



(b) Logistic regression loss



(c) Perceptron's loss



(d) SVM - Squared Hinge loss

Figure 3.11: The losses of models trained on DataScienceBowl2018

If we look at the loss functions of individual models in figure 3.11, we can notice that all of them follow the same progress. At the beginning the loss function rapidly decreases and then it either stays the same, or it progresses down slowly.

3.3 Deep learning

This section will cover our experiments with deep learning models such as U-Net, U-Net with EfficientNetB0 as the backbone, Attention U-Net, Attention ResUNet, ResUNet++ and ACNN.

In order to improve the performance of our models and the ability to generalize, we performed data augmentation. Data augmentation is a technique that expands the size of a dataset by creating modified versions of images. The modified versions of the images are created by various transformations, such as shifts, flips, zooms, and etc. This way, a model can learn how the same object on an image looks from different angles, positions and under different lighting conditions. This technique also generally helps us when we lack sufficient data.

In our deep learning experiment, we used a data generator that generates images that are randomly rotated in the range of 0 to 90 degrees, are randomly horizontally shifted in the range of 0.3 fraction of their total width and vertically shifted in the range of 0.3 fraction of their total height, randomly distorted in the counter-clockwise direction in the range of 0.5 degrees, randomly zoomed in the range of 0.3 of total image size, randomly horizontally and vertically flipped and the rest of the image filled with the reflection of the image. Additionally, all the image pixels are scaled on the scale from 0 to 1.

We conducted the following experiments. We trained each model for 50 and then for 100 epochs. The batch size was set to 8 samples, and we tried to set the learning rate to $1e-4$ and $1e-2$. The simple U-Net model used Adam as the optimizer and Binary Crossentropy as the loss function. The same settings were applied for U-Net with EfficientNet as the backbone. On the other hand, the Attention U-Net and ResUNet utilize Binary Focal Loss with gamma set to 2 as the loss function. ResUNet++ uses Nadam as the optimizer and the dice loss. Nadam is the Adam optimizer with Nesterov momentum. ACNN employs the Binary CrossEntropy for U-Net parameter tuning and the Euclidean loss between encoded prediction and encoded ground-truth mask. The number of steps for each epoch is the training data sample count divided by the batch size.

Dataset	Method	Mean Dice	Mean IoU	MAE	E_ϕ
CAS	U-Net	0.187	0.105	0.296	0.628
	U-Net+Eff.Net	0.045	0.023	0.204	0.363
	Att. U-Net	0.005	0.002	0.194	0.261
	Att. ResUNet	0.029	0.014	0.2	0.342
	ResUnet++	0.0002	7.646e-05	0.192	0.25
	ACNN	0.186	0.104	0.403	0.496
DSB2018	U-Net	0.849	0.772	0.032	0.929
	U-Net+Eff.Net	0.716	0.609	0.079	0.879
	Att. U-Net	0.147	0.082	0.227	0.525
	Att. ResUNet	0.059	0.031	0.124	0.632
	ResUNet++	0.78	0.666	0.051	0.886
	ACNN	0.055	0.029	0.568	0.236
Deepflash2	U-Net	0.352	0.221	0.016	0.648
	U-Net+Eff.Net	0.003	0.0	0.006	0.25
	Att. U-Net	0.003	0.0	0.006	0.25
	Att. ResUNet	0.003	0.0	0.006	0.25
	ResUNet++	0.01	0.006	0.993	0.25
	ACNN	0.001	0.0005	0.055	0.439

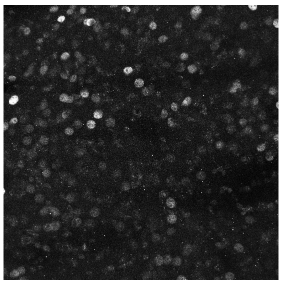
Table 3.3: Deep learning models metrics

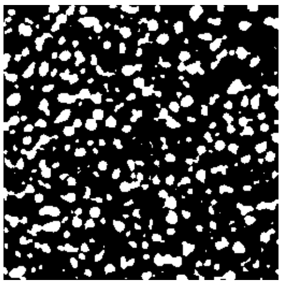
Table 3.3 shows that U-Net has the best results in almost all metrics than the other architectures. Although from table 3.4 it can be seen that it does not have the highest precision. Despite this, accuracy is not the most decisive factor. We came across situations where the model returned an utterly black image, and during the comparison between the prediction and ground-truth mask we still got an accuracy of at least 90%. This is because the neural cells occupy only a small part of the overall image in some cases. U-Net achieves slightly worse results with EfficientNetB0 as the backbone. EfficientNetB0 was pre-trained on the ImageNet dataset, and the backbone architecture weights were set as non-trainable. This is probably because images in ImageNet are significantly different from those in our dataset, so the pre-trained weights in EfficientB0 will not help us. Unfortunately, more complex architectures on all datasets performed rapidly worse than the standard U-Net network. This may mean that either we chose the wrong settings for the models, e.g. optimizer, loss function etc., we lack a sufficient amount of data in CAS and Deepflash2 dataset, or we selected the wrong data augmentation. Moreover, the other models have more trainable weights, so it is possible that if we changed the batch size or the number of epochs, the models would be more accurate.

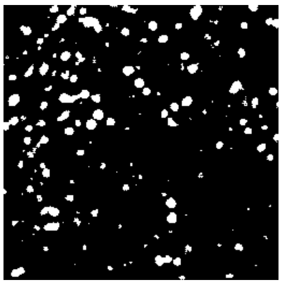
Table 3.4: Deep learning models accuracies

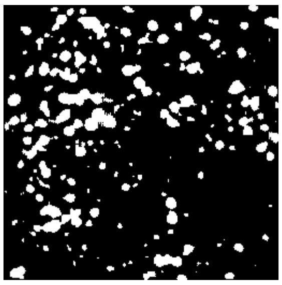
Dataset	Method	Accuracy
CAS	U-Net	69.1%
	U-Net+EfficientNet	78.2%
	Attention U-Net	79.3%
	Attention ResUNet	78.7%
	ResUnet++	79.5%
	ACNN	59.9%
DSB2018	U-Net	84%
	U-Net+EfficientNet	81.1%
	Attention U-Net	75.7%
	Attention ResUNet	87.2%
	ResUnet++	82.5%
	ACNN	46.1%
Deepflash2	U-Net	97.9%
	U-Net+EfficientNet	99.7%
	Attention ResUNet	99.7%
	Attention U-Net	99.7%
	ResUnet++	0.2%
	ACNN	94.7%

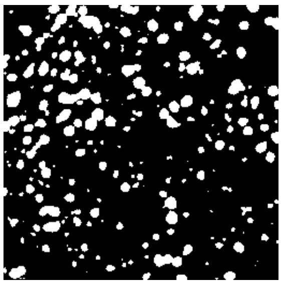
In table 3.4 you can see the average accuracy that we achieved with each model on the validation dataset. The precision in this case is not the best metric for the model comparison. In some cases the neural cells cover very small part of the image. It can happen that a model predicts a mask with all pixels set to zero, and our tests will report around 80% to 95% for the model.


(a) CAS image


(b) U-Net result


(c) U-Net + Efficient-Net result

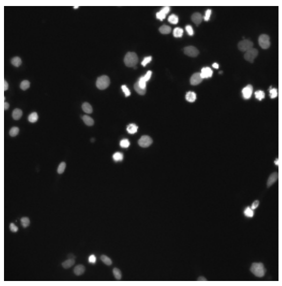

(d) Attention U-Net result

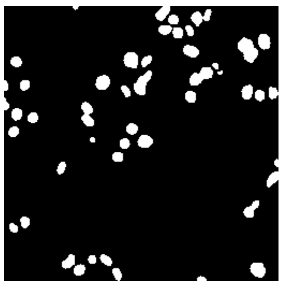

(e) ResUNet++ result

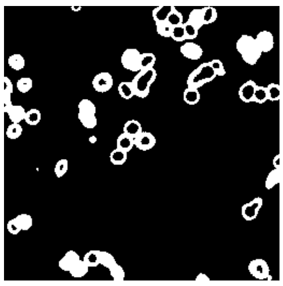
(f) ACNN result

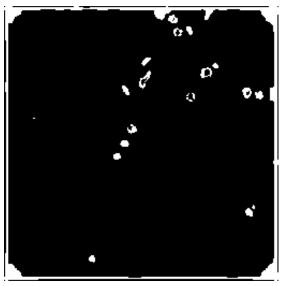
Figure 3.12: CAS deep learning results

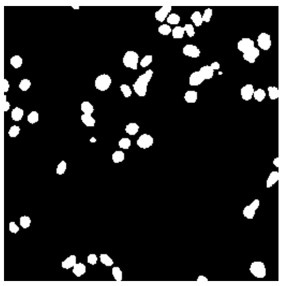
In figure 3.12 we can see that most of the neural cells were recognized by the U-Net model, whereas the ACNN's result is probably the worst. As a result, it does not contain anything resembling neural cells. This is not the standard output from ACNN. We were able to achieve better results with ACNN, but during the final evaluation, we were unable to reproduce these results again.


(a) CAS image


(b) U-Net result


(c) U-Net + Efficient-Net result


(d) Attention ResUNet result


(e) ResUNet++ result

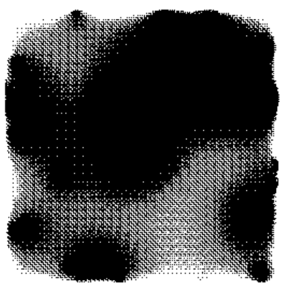

(f) ACNN result

Figure 3.13: DataScienceBowl2018 deep learning results

In figure 3.13 the most convincing results seem to be from U-Net but also from ResUNet++. For some reason U-Net with EfficientNetB0 tends to create black holes in neural cells. ACNN again performs very poorly.

Conclusion

Our task was to study image segmentation methods for biomedical research and discover the most accurate approach for neural cell segmentation from c-Fos mRNA expression images. We described and compared traditional approaches based on thresholding, edge detection and pixel-intensity distribution with the state-of-the-art methods based on machine learning and deep learning. We introduced various image segmentation metrics such as Dice score, Mean Absolute Error, Enhanced-alignment measure, Intersection over Union and precision in order to evaluate these methods. Each method was tested on three datasets: the CAS, Deepflash2, and DataScienceBowl2018. The CAS dataset was provided on request from the Laboratory of Neurochemistry of the Czech Academy of Sciences. Due to the incompleteness of the data provided we had to use extra sets of publicly available data to verify the results produced by the methods mentioned in our work.

Our assumption that more advanced architectures such as ResUNet++ or Attention U-Net will produce more accurate results than their standard U-Net predecessor was found as false. We attempted to train the models with various parameter settings but in every case, U-Net exceeded other models in terms of accuracy or prediction time. The most likely reason is that usually more complex architectures have more trainable parameters and it is often more difficult to train them with small datasets.

There are a few possible ways how the experimental part could be improved. During our study of evaluation metrics for segmentation methods, we found a metric that we were not able to implement due to lack of time. The metric is referred to as structure similarity measure (SSIM) that has been recently proposed in Fan et al. [2017]. SSIM is used for shape similarity comparison between an object that is in the prediction and an object that is in the ground-truth mask. With this test, we are able to tell more precisely whether a model returns a correct result or not.

Bibliography

Alice Abbondanza, Irina Ribeiro Bas, Martin Modrak, Martin Capek, Jessica Minich, Alexandra Tyshkevich, Shahed Naser, Revan Rangotis, Pavel Houdek, Alena Sumova, Sylvie Dumas, Veronique Bernard, and Helena Janickova. Nicotinic acetylcholine receptors expressed by striatal interneurons inhibit striatal activity and control striatal-dependent behaviors. *J. Neurosci.*, February 2022. URL `http://dx.doi.org/10.1523/JNEUROSCI.1627-21.2022`.

Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65 vol. 2. Institute of Electrical and Electronics Engineers, July 2005. URL `https://www.researchgate.net/publication/4156453_A_non-local_algorithm_for_image_denoising`.

Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L. Yuille. Attention to scale: Scale-aware semantic image segmentation. *CoRR*, abs/1511.03339, 2015. URL `http://arxiv.org/abs/1511.03339`.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-Decoder with atrous separable convolution for semantic image segmentation. February 2018. URL `http://arxiv.org/abs/1802.02611`.

Deng-Ping Fan, Ming-Ming Cheng, Yun Liu, Tao Li, and Ali Borji. Structure-measure: A new way to evaluate foreground maps. August 2017. URL `http://arxiv.org/abs/1708.00786`.

Deng-Ping Fan, Cheng Gong, Yang Cao, Bo Ren, Ming-Ming Cheng, and Ali Borji. Enhanced-alignment measure for binary foreground map evaluation. May 2018. URL `http://arxiv.org/abs/1805.10421`.

R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Pearson, 2018. ISBN 9780133356724. URL `https://books.google.cz/books?id=0F05vgAACAAJ`.

Matthias Griebel, Dennis Segebarth, Nikolai Stein, Nina Schukraft, Philip Tovote, Robert Blum, and Christoph M Flath. Deep-learning in the bioimaging wild: Handling ambiguous data with deepflash2. November 2021a. URL `http://arxiv.org/abs/2111.06693`.

Matthias Griebel, Dennis Segebarth, Nikolai Stein, Nina Schukraft, Philip Tovote, Robert Blum, and Christoph M. Flath. Deepflash2 dataset, 2021b. URL `https://drive.google.com/drive/folders/1r9AqP9qW9JThbMIvT0jhoA5mPxWEeIjs`.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. December 2015. URL `http://arxiv.org/abs/1512.03385`.

Jesper Juul Henriksen. 3D surface tracking and approximation using gabor filters. 2007. URL `https://www.yumpu.com/en/document/view/44234347/3d-surface-tracking-and-approximation-using-gabor-filters-covil`.

Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, August 1995. URL `http://dx.doi.org/10.1109/ICDAR.1995.598994`.

Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-Excitation networks. September 2017a. URL `http://arxiv.org/abs/1709.01507`.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017b. URL `http://arxiv.org/abs/1709.01507`.

HubMAP. Kaggle 2018 data science bowl, 2018. URL `https://www.kaggle.com/competitions/data-science-bowl-2018/data`.

Debesh Jha, Pia H Smedsrud, Michael A Riegler, Dag Johansen, Thomas de Lange, Pal Halvorsen, and Havard D Johansen. ResUNet++: An advanced architecture for medical image segmentation. November 2019. URL `http://arxiv.org/abs/1911.07067`.

Debesh Jha, Pia H Smedsrud, Dag Johansen, Thomas de Lange, Havard D Johansen, Pal Halvorsen, and Michael A Riegler. A comprehensive study on colorectal polyp segmentation with ResUNet++, conditional random field and Test-Time augmentation. *IEEE J Biomed Health Inform*, 25(6):2029–2040, June 2021. URL `http://dx.doi.org/10.1109/JBHI.2021.3049304`.

Iasonas Kokkinos. Pushing the boundaries of boundary detection using deep learning. In *4th International Conference on Learning Representations, ICLR 2016*, San Juan, Puerto Rico, May 2016. International Conference on Learning Representations, ICLR. URL `https://hal-centralesupelec.archives-ouvertes.fr/hal-02432711`.

MetaVO. Metacentrum (metavo) - virtual organization. [online], Accessed: 19. 7. 2022. URL `https://metavo.metacentrum.cz/en/`.

Ozan Oktay, Enzo Ferrante, Konstantinos Kamnitsas, Mattias Heinrich, Wenjia Bai, Jose Caballero, Stuart A Cook, Antonio de Marvao, Timothy Dawes, Declan P O'Regan, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Anatomically constrained neural networks (ACNNs): Application to cardiac image enhancement and segmentation. *IEEE Trans. Med. Imaging*, 37(2):384–395, February 2018a. URL `http://dx.doi.org/10.1109/TMI.2017.2743464`.

Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention U-Net: Learning where to look for the pancreas. April 2018b. URL `http://arxiv.org/abs/1804.03999`.

George Papandreou, Iasonas Kokkinos, and Pierre-André Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 390–399, 2015. doi: 10.1109/CVPR.2015.7298636.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation, 2015. URL `http://dx.doi.org/10.1007/978-3-319-24574-4_28`.

Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, Jean-Yves Tinevez, Daniel James White, Volker Hartenstein, Kevin Eliceiri, Pavel Tomancak, and Albert Cardona. Fiji: an open-source platform for biological-image analysis. *Nat. Methods*, 9(7):676–682, June 2012. URL `http://dx.doi.org/10.1038/nmeth.2019`.

Anuj Shah. Through the eyes of gabor filter. [online], 2018. URL `https://medium.com/@anuj_shah/through-the-eyes-of-gabor-filter-17d1fdb3ac97`.

T. Sorenson. *A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Its Application to Analyses of the Vegetation on Danish Commons*. Biologiske skrifter. I kommission hos E. Munksgaard, 1948. URL `https://books.google.cz/books?id=rpS8GAAACAAJ`.

Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual U-Net. November 2017. URL `http://arxiv.org/abs/1711.10684`.

List of Figures

List of Tables

A. Attachments

A.1 Source Codes

The source code attachments were written in Python. They were compiled with Python 3.8.13 on MacOS and Ubuntu machines. The following implementations are included:

- The source code of ML and DL models.
- The scripts that run the training of models
- The scripts that evaluates segmentation methods
- The implementation of evaluation metrics and other algorithms