FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Semantic Segmentation of Image Sequences Using a Spatio-Temporal U-Net

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Manuel Danner, BSc
Matrikelnummer 00825678

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Mitwirkung: Univ.Lektor Dipl.-Ing. Dr.techn. Roman Pflugfelder

Wien, 26. August 2020

Manuel Danner	Robert Sablatnig

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Semantic Segmentation of Image Sequences Using a Spatio-Temporal U-Net

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Manuel Danner, BSc
Registration Number 00825678

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Assistance: Univ.Lektor Dipl.-Ing. Dr.techn. Roman Pflugfelder

Vienna, 26th August, 2020

Manuel Danner | Robert Sablatnig

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Erklärung zur Verfassung der Arbeit

Manuel Danner, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 26. August 2020

Manuel Danner

Danksagung

Zuerst gebührt mein Dank an Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig und Univ.Lektor Dipl.-Ing. Dr.techn. Roman Plugfelder, die meine Masterarbeit betreut und begutachtet haben. Für die hilfreichen Anregungen und die konstruktive Kritik bei der Erstellung dieser Arbeit möchte ich mich herzlich bedanken.
Weiters möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Masterarbeit unterstützt und motiviert haben.

Acknowledgements

At this point, i would like to thank my supervisors Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig and Univ.Lektor Dipl.-Ing. Dr.techn. Roman Plugfelder for their continuous support. Their constructive feedback and the numerous discussions helped me to keep the right track through this work.
Furthermore, i want to thank all, who supported and motivated me throughout the years.

Kurzfassung

In der biomedizinisches Forschung, werden die Strukturen von Gewebe, Zellen, Organellen und makromolekulare Komplexen mit der Hilfe Elektronenmikroskopie Bilder erforscht. Deshalb existieren viele hochauflösende Bilder von biologischen und klinischen Proben. Infolgedessen besteht ein Bedarf an computergestützten Werkzeugen, die eine kostengünstige Lösung für die Krankheitsdiagnose bieten können. In dieser Arbeit wird eine neuartige Methode namens Elastic Gradient Transformation (EGT) vorgeschlagen, welche den Bild Gradienten verwendet, um realistisch aussehende Bild Transformationen zu machen. Die neuartige Method hilft dem Netzwerk sich selbst bei kleinen Datensätzen (wie den ISBI 2012 Datensatz) zu generalisieren, und vermeidet Overfitting. Die U-Net Architektur von O. Ronneberger, P. Fischer und T. Brox wird mit einem zusätzlichen Encoder Pfad erweitert. Das neue Netzwerk heißt SiamU-Net und verwendet zwei sequentielle Bilder t und t+1 als Eingabe. Die Ausgabe des Netzwerkes ist eine Matrix mit Wahrscheinlichkeiten. Die Encoder Pfade teilen sich keine Gewichte, und werden im latenten Raum wieder zusammengeführt. Der einzelne Decoding Pfad wird mit den skip connections des zu segmentiertenden Bildes aus dem Encoder Pfad verbunden, um ein verbessertes Upsampling zu erzeugen. Für die Evaluierung, vergleichen wir eine eigene Implementation des U-Nets und SiamU-Nets und nehmen an der ISBI 2012 Challenge teil. Weiters verwenden wir den Video Datensatz DAVIS 2016, um den Unterschied beider Netzwerke bei der Verwendung von temporalen Daten zu verdeutlichen.

Abstract

In biomedical research, detailed structure of tissues, cells, organelles and macromolecular complexes is investigated with electron microscopy (EM) images. On this account, large amounts of high resolution images from biological and clinical specimens exist. As a result, there is a need for computer assisted tools that can provide a cost effective solution for disease diagnostics. This thesis illustrates a novel elastic image transformation method called Elastic Gradient Transformation (EGT), which uses the image gradient to generate realistic looking deformations of cell structures. The novel EGT method helps our neural network to generalize on little cell datasets (like the ISBI 2012 dataset), without overfitting. The U-Net architecture by O. Ronneberger, P. Fischer and T. Brox, is adapted to contain an additional encoder path. The proposed network is called SiamU-Net, and takes two sequential images t and t+1 as input. The output is a class probability map of image t+1. It is important that both encoder paths do not share weights, and are fused together in the latent space of the network. The single decoding path uses skip connections from the encoding path of image t+1 to generate an improved up-sampling. To evaluate the adaptation, both U-Net and SiamU-Net use the novel elastic gradient transformation method and participate in the ISBI 2012 challenge. To highlight the impact of temporal image information on the two networks, a comparison of both networks is made on a video dataset called DAVIS 2016.
At the ISBI 2012 challenge, the proposed SiamU-Net with the EGT method is placed at rank 31, while the original U-Net is placed at rank 61, out of 223 participants. On the Davis 2016 challenge, the SiamU-Net achieves a 0.0776 point higher Jaccard Index than the U-Net architecture, which proves the advantage of adapting the U-Net with an additional encoder path.

Contents

CHAPTER 1

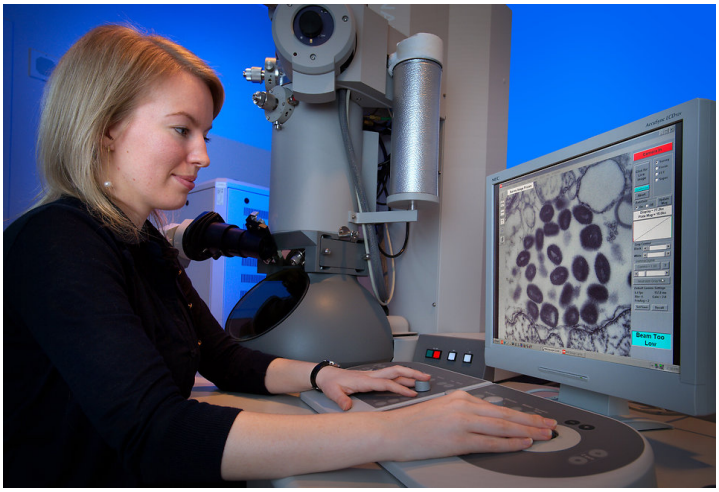
Introduction

Modern electron microscopy (EM) obtain high resolution images of biological and non-biological specimens. In biomedical research, detailed structure of tissues, cells, organelles and macromolecular complexes is investigated with EM images. On this account, large amounts of high resolution images from biological and non-biological specimens exist. As a result, there is a need for computer assisted tools that can provide a cost effective solution for disease diagnostics. Figure 1.1 shows an example of a Disease Control and Prevention intern working with a transmission electron microscope. The identification of key regions, such as organelles, tissues or cells, is important to support clinical analysis and medical interventions.

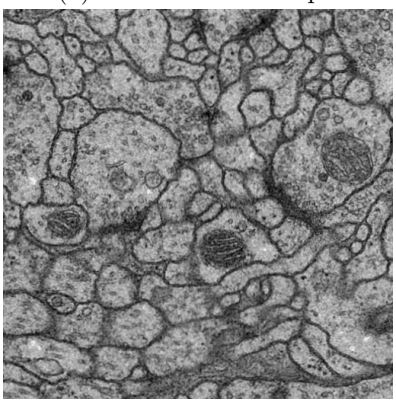
Traditionally, the manual process of a pathologist examining a tissue specimen under a microscope looking for indicators of cancer, has been a de facto standard. The manual processing is still applied in clinical settings. However, a manual examination is often times subjective and not scalable to hundreds of tissues. The amount of data which needs to be evaluated exceeds the available capacities. Unfortunately, meaningful information can not be extracted in a timely manner. Therefore, the development of machine learning and image analysis methods which provide insight into latent sub-cellular tissue characteristics would improve clinical research immensely [1].

Conventional image processing methods rely on hand-crafted models to extract and detect prominent features [2]. Good features allow the separation of data. As a result, different objects can be recognized correctly. No method produces efficient solutions on all types of medical images [3], as there exists no universal learner. A prior knowledge of cells is needed to create hand-crafted models. For cell segmentation, main characteristics of cells need to be handled properly. Cells touch and occlude each other, heaps of cells occur and flowing transitions between cell states and cell functions exist. Furthermore, EM images have low contrast and contain noise. To reduce the amount of human labour

needed by semi-automated systems, state-of-the-art methods for cell segmentation use supervised learning with deep convolutional neural networks (CNNs) [4]. A condition for CNNs to generate accurate results are extensive labelled image datasets [5]. For example, the ImageNet dataset [6] contains roughly 1.2 million training images, 50.000 validation images and 150.000 testing images [7]. It is not feasible to label biomedical datasets as huge as ImageNet by hand. Due do to the diverse structure of cells, their non convex curvatures and the low contrast of images, even experts in the field are labelling the same data differently [4]. Fortunately, depending on the biomedical dataset, data augmentation can be applied to increase the size of the dataset artificially.



(a) Electron microscope



(b) EM image

Figure 1.1: The top image depicts a Disease Control and Prevention intern, working with a transmission electron microscope [8]. The bottom image shows an EM image from the ISBI 2012 challenge dataset [4].

1.1 Motivation

I. Arganda-Carreras et al. [4] organized the first international challenge on 2D segmentation of electron microscopic (EM) images of the brain, to spark an interest in the topic and to attract new talent. Before the ISBI 2012 challenge, only researchers who worked closely with neuroscientists had access to labelled EM images. Up until now, more than 223 different participants are registered at the official leader board, which confirms the success.
To participate in the challenge, a registration is needed. After the registration is confirmed, one can download the 30 train and 30 challenge images. The train images contain the ground truth, while the ground truth of the challenge images is kept secret. Besides uploading a binary segmentation map, the competition allows the use of a probability map. The organizers apply a threshold at 0.1 steps at each image and convert them to binary segmentation maps. After several days, the results are registered at the leader board.
The first winning team of the ISBI 2012 challenge [4] by D. Ciresan et al. [9] trained a deep neural network (DNN) as a pixel classifier. Each pixel label is predicted by a square window centred around the raw pixel values. Mild smoothing and a threshold is applied to the pixel probabilities. To create three million training examples, all membrane pixels are used as positive examples, while the corresponding amount of non-membrane pixels are used as negative examples. In their work, four networks with different depth and window size are trained. The largest used architecture contains 11 Layers. One input layer, four convolutional layers each followed by a max pooling layer, and two fully connected layers in the end.
Even though the ISBI 2012 challenge was held in 2012, the participation on the public leader board is still possible. A network especially designed for the use in biomedical areas is called U-Net. O. Ronneberger, P. Fischer and T. Brox [10] designed the network in 2015 and achieved the lowest warping error in the challenge. Because of the popularity of the ISBI 2012 challenge, the evaluation metrics were adapted [4]. Currently, U-Net is placed at rank 61 with a rand score of 0.9727 [1]. It achieves state-of-the-art performance in semantic segmentation. O. Ronneberger, P. Fischer and T. Brox created a fully convolutional network with the Caffe deep learning framework [11] to segment biomedical images. It is highlighted, that excessive data augmentation is needed to train their network on the few annotated training samples available in the ISBI challenge. The structure of the network is separated in two paths. First, a contracting path is used to capture context. Two convolutions in succession are followed by a pooling layer multiple times. As a next step, the path is expanded again to enable precise localization of labels. Their network is able to train on few images, with the use of excessive data augmentation. The segmentation of a 512×512 image takes less than a second on a modern GPU.
The interest in the U-Net architecture reached a recent high, when an adaptation of it won the Kaggle 2018 Databowl challenge for nucleus detection [12].
Without a background in connectomics and bio-medical image segmentation, we wanted

[1] `http://brainiac2.mit.edu/isbi_challenge/leaders-board-new` accessed: 26.6.2020

to improve the original U-Net and take part in the ISBI 2012 challenge. We ported the network architecture to PyTorch [13] and developed a novel data augmentation called Elastic Gradient Transformation (EGT), which is used to artificially increase the small ISBI 2012 dataset. Our implementation of the U-Net with the EGT method reaches rank 37, with a Rand Score of 0.9762 [2]. This is an improvement of 24 places to the original U-Net, while only using a different elastic image transformation. However, we still saw room for improvement. Because the cell images are in sequential order, we wanted to use the additional temporal information in our network. While researching, we found the architecture of kU-Net by J. Chen et al. [14], which uses multiple U-Net stacked behind each other. Instead of stacked U-Nets, we developed a different approach. The encoder stream of the U-Net is mirrored until the bottom level, without sharing any weights. In the latent space, we combine both parts together and use a single decoder path to create our segmentation. The adaptation is called SiamU-Net and improved our rank on the ISBI 2012 even further. Our best scoring model achieves rank 10, with a rand score of 0.9772, which surpasses the original U-Net by 30 ranks.

1.2 Problem Statement

For the task of biomedical image segmentation, O. Ronneberger, P. Fischer and T. Brox. [10] created a network called U-Net. In this thesis, to increase the segmentation accuracy on the ISBI 2012 challenge, an adaptation of the U-Net is proposed. The new network is called SiamU-Net.

We want our network to be trained end-to-end on consumer grade hardware in a reasonable period of time. Therefore, any researcher should be able to train and use the proposed neural network without the need of cloud computing or expensive hardware. Deep networks, for example the kU-Net by J. Chen et al. [14] which uses multiple stacked U-Nets, or the ResNet [15]) with 152 layers, need powerful hardware and long training times. As a consequence, already pre-trained models for popular networks are used as backbone to increase the achieved results and decrease long training times. It is not possible to determine on which exact data these models are trained on. Therefore, it is time-consuming and often times impossible for researchers to reproduce the results of such networks, without access to computational facilities.
Our proposed SiamU-Net is trained end-to-end on a consumer grade GPU (NVIDIA GeForce 1070, 8 GB) in less than one to two hours (for the ISBI 2012 dataset). Therefore, our results are easy to reproduce and usable by researchers.

For the ISBI 2012 dataset, the ground truth boundary maps for the 30 training images are created by an expert in the field of brain connectomics. To avoid overfitting of our model, we increase the size of the labelled dataset by a customized elastic image transformation method. Our proposed method should generate images, which can hardly be distinguished from regular cell images. O. Ronneberger, P. Fischer and T. Brox [10], use random elastic transformations to artificially increase the size of the ISBI 2012 dataset for their training.

[2] http://brainiac2.mit.edu/isbi_challenge/leaders-board-new accessed: 26.6.2020

Random transformations do not take the additional information about the underlying data structure into their calculations. To utilize the existing image information, we introduce a semi randomness which relies on the image gradient. The gradient of the image is used to guide the transformation between the predefined random parameters ranges. The new augmentation is called elastic gradient transformation. This method should decrease the possibility to distinguish artificially altered images from the original ones. Furthermore, an early overfitting to the small original dataset is avoided.

The ISBI 2012 challenge provides users with 30 images for training, which are in a sequential order. A spatio-temporal relationship exists between the images. Therefore, it is possible to track a cell over multiple image frames. The network should use these spatio-temporal cues to increase the segmentation accuracy. The standard U-Net uses single images as input. As a result, the temporal relationship between the images is lost. In our proposed SiamU-Net, two sequential images are used as input. Both images run through a mirrored encoder path. In comparison to a Siamese Neural Network, we do not allow the sharing of weights in the encoder stream. Both paths are individual steams. In the bottom level of the network, both images are fused together, creating a temporal dependency on each other. As a result, the temporal image information is used to increase the image segmentation.

The introduction of SiamU-Net results from the idea to exploit temporal image cues in a U-Net architecture. Therefore, the importance to exploit temporal image cues should be tested. In comparison to the cell image data from the ISBI 2012 dataset, the DAVIS 2016 dataset contains videos with different moving objects. Multiple objects are visible, but only the object in the focus should be segmented. In the DAVIS 2016 dataset, videos contain frames which have a high similarity to the previous and following frame. The relationship between the similar frames in videos is called temporal coherence. To show the impact of temporal coherence, we compare the U-Net with SiamU-Net. Both networks are trained semi-supervised. The ground truth of the first video frame for each validation video is known. Before inference, the network model is trained with the first video frame multiple times. We want to highlight the added value of using temporal image cues, in comparison to a U-Net architecture.

1.3 Contribution

The contribution in this thesis contains the following topics:

- implementation of the U-Net Neural Network in PyTorch [13] to benefit from strong GPU acceleration
- novel data augmentation method called elastic gradient transformation
- novel network architecture based on U-Net to include temporal image information (called SiamU-Net)

- participation in the ISBI 2012 challenge with U-Net and SiamU-Net, both using the elastic gradient transformation
- comparison of the original U-Net and our proposed SiamU-Net on the ISBI 2012 challenge (U-Net rank 61, SiamU-Net rank 31 [3])
- comparison of U-Net and SiamU-Net on the DAVIS 2016 dataset

The thesis is organized as follows. Necessary terms and design patterns to build and train a network are explained in Chapter 2. The following Chapter gives a short introduction to CNNs, which highlights the improvement of CNNs in recent years. Besides the proposed network architecture and the introduction of our novel elastic gradient transformation method, the ISBI 2012 and the DAVIS 2016 dataset are explained. The evaluation of the network and the achieved results are highlighted in Chapter 5. Finally, in Chapter 6, the thesis is summarized and the strength and weaknesses of the proposed approach and future work is discussed thereafter.

[3] http://brainiac2.mit.edu/isbi_challenge/leaders-board-new accessed: 26.6.2020

CHAPTER 2

Convolutional Neural Networks

Y. LeCun et al.[16] introduced a Convolutional Neural Network (CNN) as early as 1989. Their network was used to recognize handwritten zip codes. Instead of using feature vectors as input, an image of a handwritten number is directly fed into the network. This work applied the backpropagation algorithm by D. E. Rumelhart, G. E. Hinton and R. J. Williams [17], which laid the foundation for training neural networks and their success in machine learning. CNNs are a subtype of neural networks, which are able to process data with grid-like topology [18]. In at least one of their layers, a mathematical operation called convolution is used. Grids allow applying discretized convolutions, which are a general formulation of applying filters to 2D-signals such as images.

2.1 Preliminary

As mentioned in the previous chapter, a condition for CNNs to generate accurate results are extensive labelled image datasets. Instead of labelling images by hand, which is often not feasible, the existing data can be augmented. Data augmentation is used to artificially increase the size of a dataset. Depending on the objects in the image, the augmentation needs to be adapted. For example, the ISBI 2012 contains cells, which can shrink and expand in any direction. Elastic image transformations as explained in Section 4.2 are used to generate plausible looking cell transformations. For the DAVIS 2016 dataset, which contains videos of animals, humans and cars, no image transformation is usable. The perspective and the shape of the object in focus is important, which can change when transformations are applied.
In the following, a short explanation of frequently used basic augmentation options is given. Afterwards, often used notation in CNN research is explained.

Basic Data Augmentation

As a preprocessing step, rotation can be applied to the training images of a network. To avoid missing image information, due to rotation, the image is reflected at the borders. This effect can be seen at the bottom left of Figure 2.1b and in the top right corner of Figure 2.1d. Depending on the characteristics of the data, a reflection at the borders can introduce new errors.

(a) Original (b) 20° (c) 30° (d) 40°

Figure 2.1: 10° rotations on the test image. Image reproduced from [19].

To expand a trainings dataset further, horizontal and vertical flipping can be used. This augmentation generates mirrored images at the given axis. Figure 2.2 shows the two different directions of flipping. The necessity of flipping depends on the use case and the training images. For cell images, horizontal and vertical flipping can make sense. Due to the limitations of microscopes, cells are recorded in a top down view on a slide or in a Petri dish. In these cases, the orientation of the cells is arbitrary. Vertical and horizontal flipping of the cell image will not introduce errors. For datasets where the image perspective does matter, these augmentations need to be used with care. For example, when training a network for frontal face recognition, it would not make sense to allow vertical (top to bottom) flipping, while horizontal (left to right) flipping can be used.

(a) Original (b) Vertical flip (c) Horizontal flip

Figure 2.2: Horizontal and vertical flipping applied to the test image. Image reproduced from [19].

Convolution

A convolution is a mathematical operation of two functions. It is a weighted average of function f, related to function g. Equation 2.1 shows the mathematical description of a

convolution, which is denoted by an asterisk symbol. The value of $f(\tau)$ is weighted with $g(x-\tau)$.

$$(f \otimes g)(x) = \int_{-\infty}^{\infty} f(\tau) \cdot g(x-\tau) d\tau \tag{2.1}$$

In the context of CNNs the function f is referred to as the input (image), while g is referred to as kernel function. The output of the convolution is called a feature map. In machine learning, N-dimensional arrays of data are called tensors. Data in form of a scalar, a vector, a matrix or a cube are therefore called: rank-0, rank-1, rank-2 or rank-3 tensor. For a two-dimensional image I, with a corresponding two-dimensional kernel K, we adapt the formula as seen in Equation 2.2.

$$(I \otimes K)(i,j) = \sum_m \sum_n I(m,n) K(i-m, j-n) \tag{2.2}$$

The kernel is flipped relative to the input. The flipping of the kernel has no impact on the network. For this reason, many machine learning libraries implement the cross-correlation instead [18], where the kernel is not flipped. The cross-correlation is calculated according to Equation 2.3.

$$(I * K)(i,j) = \sum_m \sum_n I(m,n) K(i+m, j+n) \tag{2.3}$$

In this work, we follow the convention proposed by I. Goodfellow, Y. Bengio and A. Courville [18] of using convolution and correlation interchangeably. If we distinguish correlation and convolution for special cases, it is mentioned accordingly.

For CNNs, the input can be an image of arbitrary size. Internally, the image is represented as a tensor. For a standard RGB image, each value in the corresponding 3D tensor, correlates to the intensity or colour of a pixel. A convolution applies a filter, which is called kernel, at each element of the tensor. The kernel size and depth can vary, depending on the designed network. The values of the kernel are called weights, which are adapted by the learning algorithm. Figure 2.3 shows a simple convolution with kernel size 3×3. Neurons are all elements in a CNN of the form: $\sigma(\sum_j w_{ij} x_j + b_i)$. w_{ij} is the weight between neuron i and input j, x_j is the corresponding input element and b_i is a bias term. Furthermore, input and output elements are called neurons too. Activation functions as $\sigma(.)$ are explained in Section 2.2. Because multiple values are convolved into the output neuron, this neuron contains information of its receptive field. All neurons which are used to calculate a new neuron are contained in the receptive field. Therefore, the new calculated values of the activation map (or feature map) do not correspond to intensity or colour values any more. For example, if an edge exists in the input image, the neuron corresponding to the edge in the activation map is activated.

Sparse Connectivity

As stated by I. Goodfellow, Y. Bengio and A. Courville [18], traditional neural network layers use a dense connectivity. Every input neuron is connected to each following output

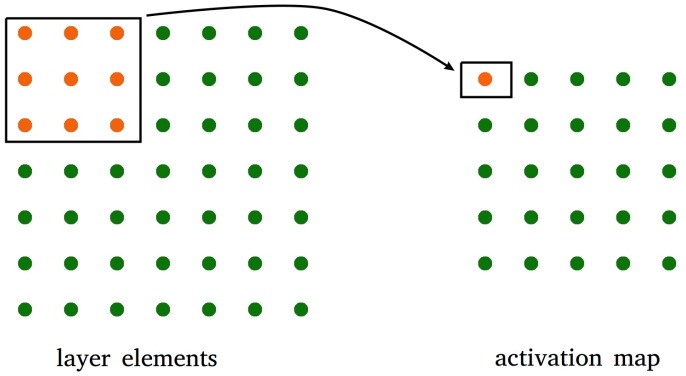


Figure 2.3: The computation of a convolution on an input image with stride 1 and no padding. A 3×3 filter is applied on a 7×7 image. The output is an activation map of size 5×5. Orange layer elements are the pixels which are currently used for calculations. The green layer elements are visited later. The receptive field of the orange elements in the activation map is shown as black rectangle in the left image. In this example, the receptive field is composed of nine layer elements. The filter is slid over the whole input image to create the activation map.

neuron, therefore each output is affected by every input neuron. In contrast, CNNs have a sparse connectivity. On this account, the filter size is smaller than the input image size. A greyscale image of size 512×512, has 262.144 weights, if a dense connectivity is used, for each activation map of size 1×1. If a sparse connectivity is applied, a filter size of 3×3, uses only 9 weights for the resulting activation map of size 510×510, when no padding is applied. Padding is explained in detail in Section 2.1. Small filters are used to decrease the amount of (floating-point) operations. For example, a 2×1 filter uses $511 \times 512 \times 3 = 784.896$ operations. (For each output pixel, two multiplications and one addition is needed).The same transformation with a dense matrix multiplication needs $512 \times 512 \times 511 \times 512$ operations. As a result, far fewer operations are needed to create the same activation map in a sparse connectivity system.

Padding

When a convolution is applied over an image, pixel information at the border is missing, depending on the filter size. Therefore, the output image size is reduced, by $\lfloor filtersize \div 2 \rfloor \times 2$ at each axis. A method to handle border pixels is padding, which adds estimated values at the sides. The following examples show three different versions, zero padding, replicate padding and mirror padding. All examples are explained for a 2-D image. It can be extended easily to multiple dimensions.
Zero padding adds rows and columns of zero values at the four borders of the matrix. Instead of zeros, replicate padding adds a copy of the adjoining borders to be able to calculate the convolution for each neuron. Mirror padding reflects the pixels at the edge

of the border. All of these methods introduce an error at the borders, but contain the shape and border pixel information after an convolution is applied. The benefit of the consistent matrix size and the retaining of border pixels outweighs the introduced errors. Figure 2.4 shows an example of padding applied on an one-dimensional array.
When no padding is used, the size of the resulting feature map is reduced, depending on the used kernel size. Information at the boarder is lost, however, no border errors are introduced after. In Figure 2.3, no padding is used, therefore the input matrix of size 7×7 gets reduced to an output matrix of size 5×5.

0 0 0 | 1 2 3 4 5 6 7 | 0 0 0

(a) zero padding

1 1 1 | 1 2 3 4 5 6 7 | 7 7 7

(b) replicate padding

4 3 2 | 1 2 3 4 5 6 7 | 6 5 4

(c) mirror padding

Figure 2.4: Three different types of padding shown on an one-dimensional array. Zero padding, replicate padding and mirror padding.

Stride

By default, the filter matrix is convolved over every layer element of the input matrix. With a custom set stride, one can skip a set amount of layer elements. For example, a stride of one does not skip a layer element. The receptive field is shifted over every pixel. A stride of two skips every second pixel in both directions. A stride of 3 skips 2 pixels at a time in both directions. It is important to note, that the row and column size of the output matrix has to be an integer value. No fractions are allowed. A benefit of using a stride is to remove the overlap of the receptive fields. The receptive fields overlap less, the higher the stride parameter is set. However, the resolution of the matrix gets reduced. In Figure 2.3, a stride of one is used, with no padding. In Figure 2.5, stride is set to two. Instead of shifting the receptive field over every pixel, only every second pixel is used.

Dilation

Dilation is similar to stride, but instead of skipping pixels from the input matrix, the filter kernel introduces spaces between its filter values. Figure 2.6 shows an example of a matrix with no padding and no stride, but with a dilation set to two. A 3×3 filter is used and only the orange middle point and blue side values are used for the filter calculation. Unlike the matrix reduction received by setting a stride, dilation does not reduce the output size if padding is used. Furthermore, dilation filters support exponentially expanding receptive fields, where no resolution or coverage is lost [20].

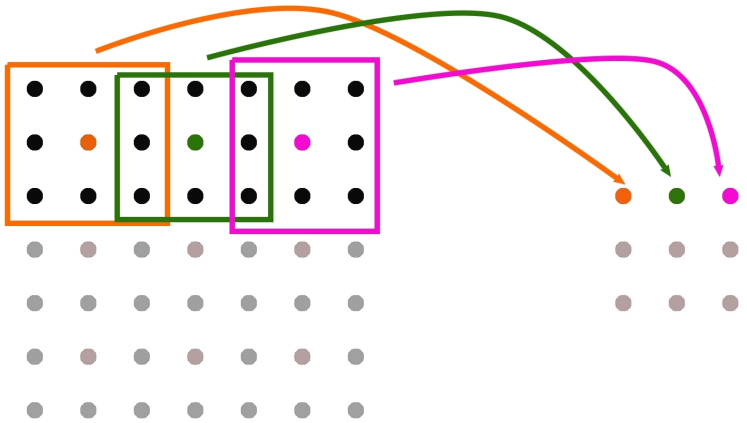


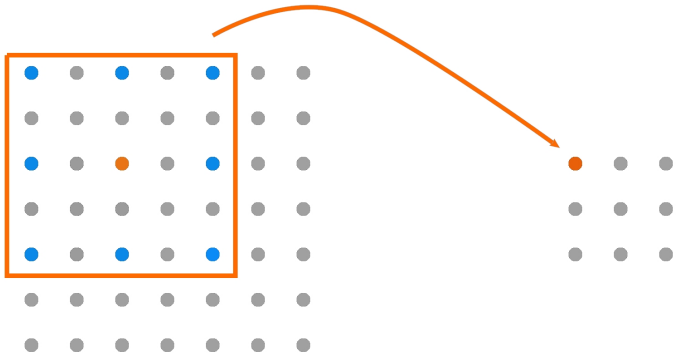
Figure 2.5: The computation of a convolution, with stride two and no padding, on an input image. A 3×3 filter is applied on a 7×7 matrix. The output is an activation map of size 3×3. The first row from the activation map on the right matrix, contains the corresponding convolutions for the orange, green and pink receptive fields from the left matrix.

2.2 Network Parts

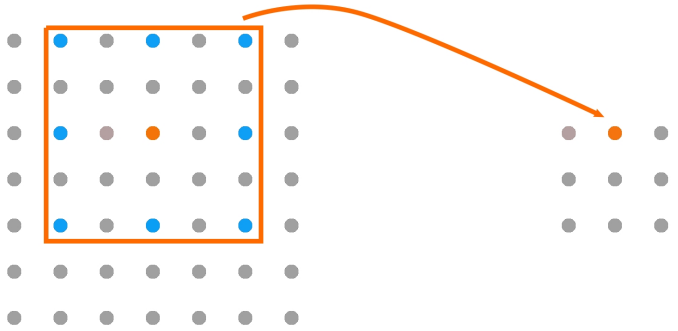
CNNs can be separated into three parts. Specifically, an input layer, one or more hidden layers and an output layer. The input layer is the first layer in the network. It is used to pass the data to the first hidden layer. The key concept of neural networks are hidden layers, which are used between input and output layers. These hidden layers have trainable parameters. Each neural network can be represented by a function. Because the convolution is a linear operation, multiple stacked convolutions can collapse to a single convolution operation. As a result, activation functions are needed to approximate non-linear relations. In the next section, different types of hidden layers and functions are explained in detail. Besides hidden layers like, convolution, pooling, and transposed convolutions, different activation functions are explained. For simplification, a greyscale image is used as the input to the network.

Convolutional Layer

The name CNN derives from the mathematical term of a convolution. As previously mentioned in Section 2.1, the convolution can be seen as a weighted average, where the weighting is implied by a different function. For CNNs, each convolutional layer needs user defined parameters. The channels of the input to the layer, the number of output channels, the kernel/filter size, the stride value, the padding type, the dilation value and whether a bias term should be added or not.
The input to the convolutional layer is a tensor with a predefined size of $n \times m \times d$. The number of rows is called n. m corresponds to the number of columns and d is the



(a) dilation filter over the first pixel



(b) dilation filter over the second pixel

Figure 2.6: Dilation is applied on the image matrix with a value of 2 and a filter size of 3×3. No padding and no stride is added. The filter uses only the orange and blue pixels in the left matrix to calculate the orange value in the right matrix. One can imagine the dilation as a sparse filter of size 5×5, where all values are zero, except for the blue and orange pixels.

amount of channels (depth of the tensor). A kernel of size $k_1 \times k_2 \times d$ is convolved with the input tensor, with $k_1 \ll n$ and $k_2 \ll m$. One kernel is used to create one activation map. In a convolutional layer, the amount of output channels e, correspond to the number of different kernels which are applied to all the input channels. Therefore, a total of $k_1 \times k_2 \times d \times e$ weights are used, without the bias term. The bias term would add additional e weights [21].
A special case is the convolution with a $1 \times 1 \times k$ filter. This convolution can be used to reduce the depth of a tensor, and compress multiple feature maps into one. This technique can be used at the end of the network to generate a segmentation for each pixel. Another benefit is shown in Squeezenet by F. N. Iandola et al. [22]. They replace all 3×3 convolutions with 1×1 convolutions. As a result, their Network uses $50\times$ fewer parameters than AlexNet [23], and the model size can be compressed to less than 0.5

MB, which is 510× smaller than AlexNet, with equivalent accuracy.

A convolution is a linear function. Therefore, to introduce non-linearity to a CNN, non-linear activation functions σ, like the rectified linear unit (ReLU), or the classical sigmoid function $\sigma = \frac{e^x}{1+e^{-x}}$, are needed. Without non-linear activation functions, multiple convolution layers can be reduced to a single convolutional layer.

Pooling Layer

The pooling layer is used to introduce an invariance to object motion and deformations. Because of the decrease of the tensor size, the computational complexity is reduced too. There are different types of pooling layers, which can be used to accomplish a tensor size reduction. For example, average pooling or max pooling. Figure 2.7 shows max pooling and average pooling applied on the same input tensor. In this layer, the depth of the output tensor is equal to the input tensor. Only the width and height of the input matrix is reduced to create the output tensor. The position of the high activation (the highest value in tensor) is not important, as long as the value is inside a pooling region. It is important to note, that this layer has no learnable weights.

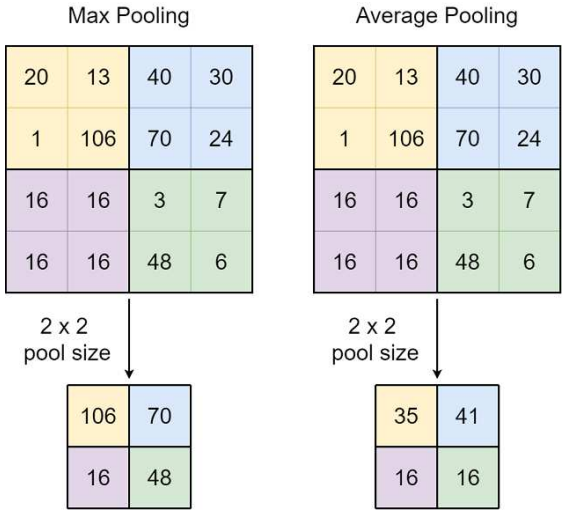


Figure 2.7: Max pooling with a 2×2 pool size is applied on the left tensor. The highest value in each 2×2 block is propagated to the output. The size on each axis is halved, resulting in a reduction of 75% of the information. On the right tensor, an average pooling operation with a 2×2 pool size is applied. The average value of each 2×2 block is propagated. Both types of pooling operations generate drastically different outputs.

Transposed Convolution Layer

A transposed convolution can be seen as a reverse convolution layer. A different name is fractionally strided convolution or deconvolution as mentioned by V. Dumoulin and F. Visin [21]. Therefore, it can be used to up-sample an input with learnable weights. In

computer vision, there are various techniques to up-sample an image. For example: nearest neighbour interpolation, bi-linear interpolation or bi-cubic interpolation [24]. With these interpolation techniques, the network would have no learnable up-sampling parameters. The transposed convolution layer takes the same parameters as the convolution layer, which is defined in Section 2.2. A transposed convolution can always be emulated with a convolution, by adding rows and columns of zeros to the input tensor. More efficient implementations of the transposed convolution exist, where the operation can be seen as a gradient of a convolution with respect to the input, as mentioned by V. Dumoulin and F. Visin [21]. Figure 2.8 shows a convolution followed by a transposed convolution, which is used to up-sample the input. Zero padding is added at the top 2×2 tensor (blue), to create the 4×4 tensor(green). It is important to note, that the transposed convolution is not an inverse convolution operation. The initial input values for the convolution can not be recreated. However, the size of the initial feature map can be restored.

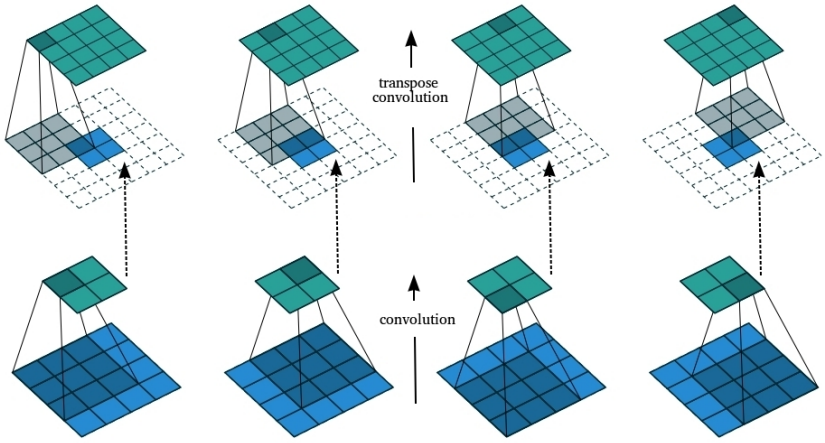


Figure 2.8: A 3×3 filter is convolved over the bottom blue 4×4 tensor, to create a feature map of size 2×2. To up-sample the resulting 2×2 tensor, a 3×3 filter is convolved over the top blue 2×2 tensor, which is padded by a 2×2 zero border and a stride of one. This transpose convolution creates a 4×4 feature map. The transposed convolution is not an inverse convolution operation, as the initial input values can not be recreated. Image reproduced from [21].

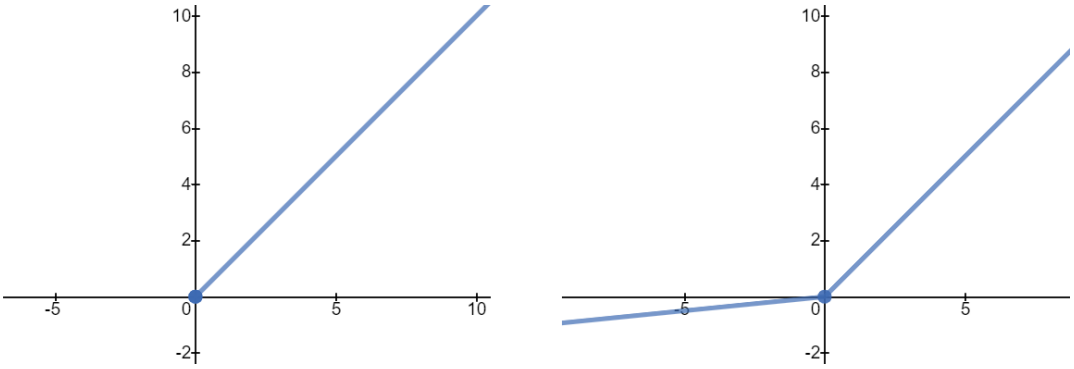
Dropout

The dropout layer is only used during the training process and is deactivated afterwards. With a set probability p, elements of the input to the layer are replaced with zero. On every forward call, the chosen elements are randomized again. As a result, different neurons need to adapt to the changes in the input. The network gets less sensitive to specific neuron weights. This method helps to reduce overfitting and prevents co-adaptations, as mentioned by G. E. Hinton et al. [25].

Activation Function

To learn complex functions, convolutions are stacked on top of each other in neural networks. A combination of linear functions is a linear function too. Therefore, a stack of convolutions without an activation function, can be reduced to a single convolutional layer. The introduction of non-linearity with activation functions allows the network to have multiple convolution layers, which do not collapse to one.
The rectified linear unit (ReLU) is a simple, fast and often used activation function [26]. The function prevents negative values in the activation map. Equation 2.4 shows the mathematical term of the activation function, where x is the tensor result from a given layer. Negative values are mapped to zero, while positive values do not change. Figure 2.9a shows the curve of the ReLU function. Besides the popularity of the ReLU activation function in machine learning, one major drawback exists, called the dying ReLU problem. A neuron can get inactive, therefore the output of the neuron is always zero for every input it gets. The neuron can never recover, because the gradient does not change for negative values. A solution for this problem exists, called the leaky ReLU. Figure 2.9b shows a plot of the leaky ReLU function. For a negative input x, the value is multiplied by a small constant factor, for example $0.01 \times x$. Non negative values are handled the same way as with the ReLU function. The gradient for negative values exists, as a result the backpropagation can update the weights of the neuron.

$$\sigma_{ReLU(x)} = max(0, x) \tag{2.4}$$



(a) ReLU (b) Leaky ReLU

Figure 2.9: For the ReLU, all negative input values are mapped to zero. As a consequence, neurons can become inactive, because of the resulting zero gradient. To tackle this problem, the leaky ReLU multiplies negative input values with a small constant factor. Therefore, the possibility to recover the weight still exists.

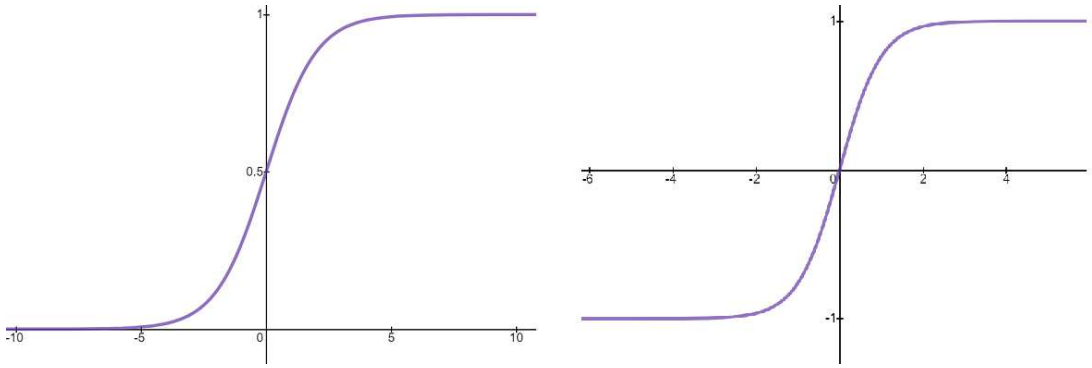
The sigmoid function is defined in Equation 2.5 and shown in Figure 2.10a. The input is compressed between the values $[0, 1]$. Therefore, large positive numbers are mapped

to one, while large negative numbers are mapped to zero. A problem of the sigmoid function is the vanishing gradient problem. The gradient at the edges is almost zero. During backpropagation, gradients close to zero are multiplied together, vanishing the gradient in early layers. The weights are not updated effectively any more. Deep neural networks with many hidden layers need to take care of this problem.

$$\sigma_{Sigmoid(x)} = \frac{1}{1 + e^{-x}} \tag{2.5}$$

The tanh function is defined in Equation 2.6 and shown in Figure 2.10b. The function can be seen as a scaled sigmoid function. Numbers are compressed in the range of $[-1, 1]$, with a zero-centred output. The gradient is stronger for tanh than it is for a sigmoid activation, therefore it is often preferred over the sigmoid function. Nevertheless, the vanishing gradient problem still exists with the tanh function.

$$\sigma_{Tanh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.6}$$



(a) Sigmoid (b) Tanh

Figure 2.10: The sigmoid function shifts an input value between zero and one. In contrast to that, the tanh function shifts inputs between -1 and 1.

The softmax layer is used to generate a probability matrix for the output layer. The probability to belong to each target class is assigned to every layer element, therefore each element is scaled between 0 and 1. For example, foreground and background pixel affiliation can be seen as two different classes in image segmentation. The sum of the class probabilities for each layer element adds up to one. Equation 2.7 shows the corresponding softmax function, where k is the amount of used classes -1. x_i is a layer element, belonging to class i.

$$\sigma_{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{n=0}^{k} e^{x_n}}, \text{for } i = 0, 1, 2, ..., k \tag{2.7}$$

Loss Function

In supervised machine learning, it is necessary to compare the actual result (called ground truth), with the predicted output [18]. A loss function is needed to measure an error between the predicted values and the ground truth labels. In Equation 2.8, the predicted label of an input image x_i, with the according weights W, is compared with the ground truth label y_i. To calculate the average loss over all examples, the sum of the losses for each example is divided by the amount of examples N. Therefore, the performance of the network can be connected to the loss function. For accurate predictions, the calculated error needs to be minimized.

$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i) \tag{2.8}$$

In neural networks, backpropagation computes the gradient in the hidden layers. The weights of each layer are modified with an optimization function. Figure 2.11 shows the correlation of the loss function and the optimization. Briefly summarized, the gradient of the loss function, with respect to the weights and biases of the layers, is calculated. Until a minimum of the loss function is reached, the weights are modified in the opposite direction of the gradient of each layer. In this work, the stochastic gradient descent is used as an optimization function.

In the implementation of the U-Net architecture by O. Ronneberger, P. Fischer and T. Brox [10] the Cross Entropy Loss is used. To follow the idea of the cross entropy loss, a short explanation of the entropy and the Kullback-Leibler Divergence is given.

Equation 2.9 shows the calculation of the entropy. The entropy H is defined as the sum of the probability P of each label i, times the information gain $-\log_2(P(i))$ of the label. The entropy measures the average information content one can expect, if one of the labels occur. For example, if a coin is tossed and both head and tail have a probability of 50%, an entropy of 1 will occur. 1 bit is needed to transport the information. If both sides are head, the entropy will be 0, because no new information is needed to predict the outcome. It will always be head.

$$H(P) = -\sum_i P(i) \log_2(P(i)) \tag{2.9}$$

The Kullback-Leibler Divergence (also called relative entropy), measures how big the variance between two probability distributions is. Equation 2.10 shows how the Kullback-Leibler Divergence (KLD) is calculated. The difference between KLD and the entropy, lies in the additional term of $Q(i)$ in the logarithm. If both distributions are identical,

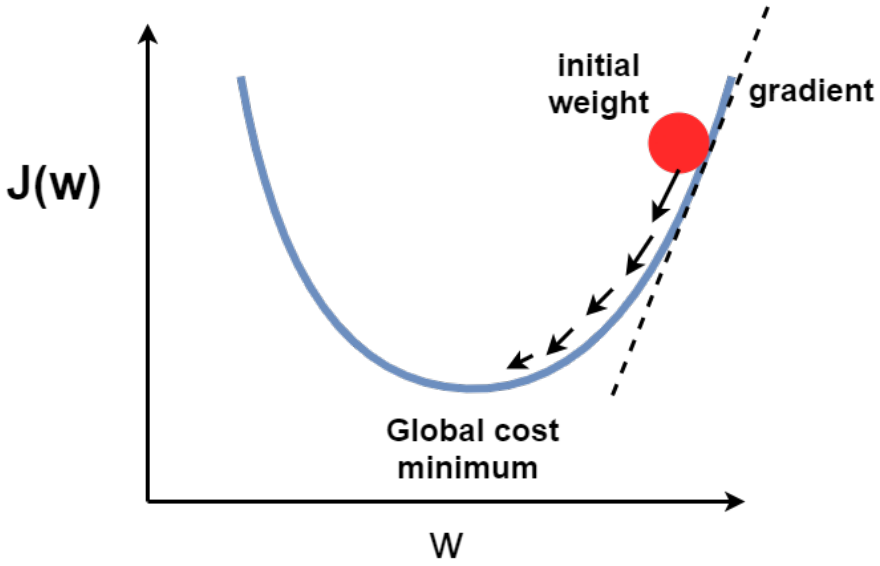


Figure 2.11: The gradient of a given loss function L with weights W is calculated. By modifying the weights in the opposite direction of the calculated gradient, a minimum can be found.

the logarithm term evaluates to zero, hence the function result is zero.

$$D_{KL}(P \parallel Q) = \sum_i P(i) \log_2(\frac{P(i)}{Q(i)}) \tag{2.10}$$

Instead of measuring the average information content one can expect, the KLD measures the additional content which is needed, if the probability distribution Q is used. If both distributions are equal, the function can be zero. Therefore, if the predicted output of a network is seen as a probability, one can use the KLD as a loss function for the neural network.

When using the cross entropy loss in a neural network, two probability distributions are needed. First, each predicted label is seen as a probability to belong to the right class. Therefore, the neural network creates a probability distribution. The other probability distribution is defined by the ground truth labels. The cross entropy measures the average information content one can expect, if the probability distribution Q is used, instead of P. The entropy and the KLD can be used to define the cross entropy, shown in Equation 2.11. The entropy of the ground truth data is added to the KLD.

$$H(P, Q) = D_{KL}(P \parallel Q) + H(P) \tag{2.11}$$

Inserting Equation 2.9 and Equation 2.10 into Equation 2.11 results in Equation 2.12.

$$H(P,Q) = \sum_i P(i)\log_2(\frac{P(i)}{Q(i)}) - \sum_i P(i)\log_2(P(i)) \tag{2.12}$$

With the logarithm quotient rule, Equation 2.12 can be split into three sums, as shown in Equation 2.13.

$$H(P,Q) = \sum_i P(i)\log_2(P(i)) - \sum_i P(i)\log_2(P(i)) - \sum_i P(i)\log_2(Q(i)) \tag{2.13}$$

Equation 2.14 shows the final term of the cross entropy for discrete distributions. $P(i)$ corresponds to the probability gathered by the ground truth data. $Q(i)$ is the probability generated by the neural network.

$$H(P,Q) = -\sum_i P(i)\log_2(Q(i)) \tag{2.14}$$

Equation 2.15 shows the calculation of the cross entropy for a single neuron.

$$\begin{aligned} \text{Class A prediction} = 30\% \quad & \text{Class A ground truth} = 0\% \\ \text{Class B prediction} = 70\% \quad & \text{Class B ground truth} = 100\% \\ H = -(0 \cdot \log(0.3) + 1 \cdot \log(0.7)) &= 0.154 \end{aligned} \tag{2.15}$$

Both the cross entropy and the KLD can be used as a loss function for neural networks. While KLD can be zero, the cross entropy will be equal to the entropy of the distribution, if both distributions are equivalent. For gradient optimizations, both methods generate the same end result in the network and can be equally used.

2.3 Chapter Summary

In this chapter, the progression of CNNs in the last years, which beat human classification, is discussed. After that, the core concept of a convolution, with the needed parameters and terms, is explained. These terms and definitions are needed, to fully grasp the properties of modern CNNs. The most common layers to construct a CNN are explained thereafter. Besides convolution, pooling and dropout layers, activation and loss functions are discussed.

CHAPTER 3

Related Work

In recent years, CNNs occupy the top places at image processing challenges. For example, since 2012, the winner of every ImageNet challenge [7] uses a deep neural network architecture. The first winning deep neural network, which used a fast GPU implementation of a CNN called AlexNet proposed by A. Krizhevsky, I, Sutskever and G. E. Hinton, [23], won with a top-5 test set error rate of 16.4%. The architecture is shown in Figure 3.1. Some layers are split, to train the network on 2 separate GPUs simultaneously. In comparison, humans score a top-5 error of 5%. In 2013, M. D. Zeiler and R. Fergus created ZFNet [27], which decreased the error rate to 11.7%. Figure 3.2 displays the tuned AlexNet, which is called ZFNet. Instead of using 11×11 filter size like AlexNet, ZFNet uses 7×7 filter sizes in the first layer. Additionally, M. D. Zeiler and R. Fergus [27] introduce a deconvnet layer for visualization. This new layer is used to visualize the feature activations of the convnet. Instead of mapping pixels to features, like in the convnet, the deconvnet maps features to pixels. Therefore, the deconvnet has no learnable parameters and is attached to an already trained convnet. The idea of deconvnet layers is used by Ronneberger O. Ronneberger, P. Fischer and T. Brox in their CNN called U-Net [10]. They extend the deconvnet layers with learnable parameters and let the network learn a class label for each pixel.

While VGG Net by K. Simonyan and A. Zisserman [28] (7.3%) and GoogleNet by C. Szegedy et al. [29] (6.7%) failed to beat human classification in 2014, ResNet by K. He et al. [15] (3.6%) surpassed human classification for the first time, one year later. The simple architecture and novel approach of VGG Net, with only 19 layers (in comparison to 152 layers of ResNet), is still used today. Instead of using a large filter, two consecutive small filters are applied to generate the same receptive field. For example, two 3×3 filters have the same receptive field as one 5×5 filter. A major benefit of this method is the reduction of learnable parameters. Instead of 25 weights, only 18 weights are needed.

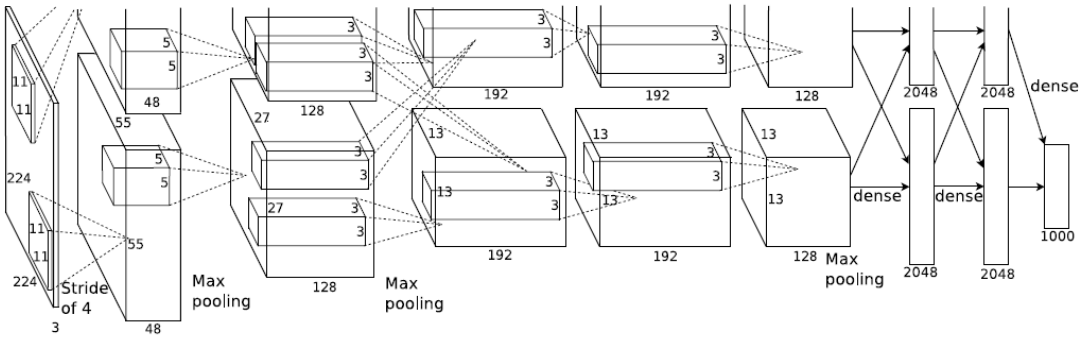


Figure 3.1: AlexNet 8 layer architecture. The architecture allows the network to be trained on 2 separate GPUs. A communication between GPUs is established only at certain layers. Image from [23].

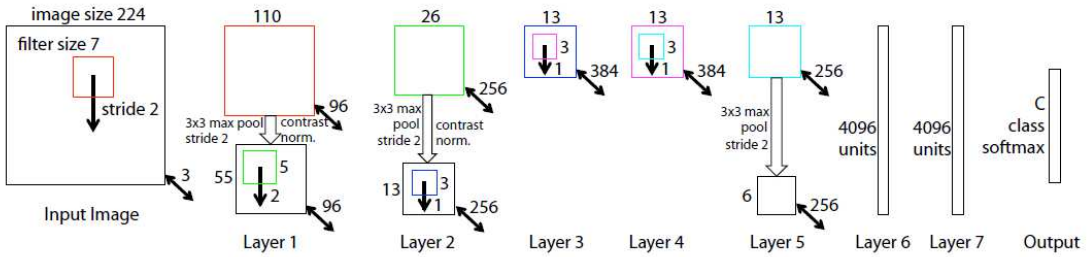


Figure 3.2: ZFNet 8 layer architecture. It uses an AlexNet structure, with tweaked hyper-parameters. Image from [27].

Since the success of neural networks in image classification, as mentioned before, the approach of CNNs has spread to different areas in segmentation too. The progress in recent years stems from powerful baseline systems. The baseline of Fast R-CNN by R. Girshick [30] contributes to the development of Faster R-CNN by S. Ren et al. [31] and Mask R-CNN by K. He et al. [32], which are dominant in object detection. The baseline of the Fully Convolutional Network (FCN) by J. Long, E. Shelhamer and T. Darrell [33] is found in encoder-decoder architectures like the U-Net [10] network. Te goal of semantic segmentation is to assign a class label to each pixel. It is not important to distinguish objects of the same class, in contrast to instance segmentation. The focus in instance segmentation is to detect all objects and still label every instance separately. Instance segmentation can be seen as an adapted semantic segmentation. In this work, the focus lies on a semantic segmentation.

3.1 Semantic Segmentation

In 2015, J. Long, E. Shelhamer and T. Darrell [33] developed a network named fully convolutional network (FCN). The basis of their work is a reinterpretation of networks which are used for image classification. To create a fully convolutional network, they strip the classification network of its final layer and replace all fully connected layers with convolutions. As a result, the new network can process inputs of arbitrary size and create classification maps as output. Typically, networks use two convolutions followed by a pooling layer multiple times, to downsample the input [28]. After the last convolution, the image is upsampled with a high stride, to produce an output segmentation map. Multiple variations are proposed to combine different convolution layers. Depending on the implemented variation, the output image can contain coarser or fine-grained information.

H. Chen et al. [34] create a network, which is inspired by the work of J. Long, E. Shelhamer and T. Darrelll [33]. Before the first two max pooling layers, two convolutions are used. After the second max-pooling, the structure is adapted. Three convolutions are followed by a max-pooling layer, which again is followed by three convolutions. After each pooling operation, the channel size is doubled in the next convolution. Therefore, the network consists of four levels. In the last layer of the second, third and fourth level, a transpose convolution is used to upsample the image to its original size with two feature maps, followed by a 3×3 and a 1×1 convolution. All six feature maps are summed together. A softmax classification layer is used to generate a probability map. In this network, the up-sampling path is necessary to reconstruct fine details of the image.

For the task of biomedical image segmentation, O. Ronneberger, P. Fischer and T. Brox [10] created a network called U-Net. It contains two different paths. The contracting path, which is well-known from image classification, contains four levels. Each level consists of two convolution layers of size 3×3, each followed by a rectified linear unit (ReLU). To down-sample the image, a 2×2 max pooling layer is used, with a stride set to 2. The number of features is doubled at each down-sampling step. The bottom layer follows the same structure as the previous layers, but instead of using a down-sampling, an up-sampling operation is used, which halves the feature channels. A 2×2 transposed convolution layer, with stride 2, handles the up-sampling, followed by a ReLU. Instead of using an up-sampling layer, which does not have learnable weights like the deconvnet in ZFNet [27], transpose convolutions are introduced. Therefore, the up-sampling operations is trainable. In the expansive path, a copy of the corresponding image of the contracting path is added to the feature map. To reduce the combined size of the channels, the first convolution halves the amount of features again. The final layer adds a 1×1 convolution, which maps the features to the number of defined classes. In contrast to H. Chen et al. [34], skip connections are introduced. Furthermore, the amount of feature maps is increased drastically. For biomedical images, it is highlighted, that excessive data augmentation is needed to train the network, if only few annotated training samples

are available. U-Net can be seen as a combination of the FCN [33] and ZFNet [27] architecture. As a result, the U-Net architecture achieves a more precise segmentation with the decoder path, than the FCN architecture. Additional advantages of the U-Net are the fast training times and the ability to work with datasets containing few images.

T. M. Quan, D. G. C. Hildebrand and W. Jeong [35] used a U-Net design with added residual blocks called FusionNet. Between each two convolutions in each layer, a residual block, which contains 3 convolutions and one skip connection is added. Furthermore, instead of concatenating the skip connections with feature maps from the same level, a pixel-wise addition is used to merge the maps together. Further, the dataset is rotated in 90° and flipped in both directions to generate seven additional versions of each image. Each epoch, a random field deformation is applied, to avoid overfitting.

J. Chen et al. [14] introduced two key network parts in their architecture. First, a kU-Net is explained. k is the amount of U-Net modules which are nested. For the case of $k = 2$, U-Net-2 is the first network in the stack. It applies the known U-Net structure on a downsampled input. U-Net-1 works on the original image. The output of U-Net-2 is fused in U-Net-1 after the first down-sampling level. The behaviour of human experts should be simulated, who first look at the whole picture to understand context, and zoom in to label the boundaries after. The second network component consists a recurrent neural network (RNN) which consists of Bi-Directional Convolutional Long Short Term Memory (BDC-LSTM) units. The name is derived from an adaptation of the CLSTM by S. Xingjian et al. [36]. CLSTM expands the LSTM [37] by replacing the vector multiplication with an convolution operation, therefore allowing images as input. BDC-LSTM is a stacked CLSTM layer, where the contextual information in two opposite directions is fused. Additionally, multiple BDC-LSTM layers can be stacked by taking the output of one layer as input to the next. Six different BDC-LSTM layers are combined with standard layers in between in a deep architecture. A 5×5 convolution is following the first, third, fourth and sixth BDC-LSTM layer. Max-pooling is applied after the second and a deconvolution after the fifth BDC-LSTM layer. At the final convolution, an additional 1×1 convolution is used to generate the output. While the first network captures context from 2D slices, the second network is used to capture 3D context of the given 2D slices. Both network components, kU-Net and RNN, can be trained separately or in combination.

S. Chopra et al. [38] presented a Siamese architecture to train a similarity metric for face verification. The main architecture contains two identical convolutional networks, which share the same weight. The used energy functions measures the difference between the output of the two identical networks. Therefore, if an image of the same person is fed to both networks, the output vectors should be close to each other, resulting in a low energy function. If both networks get a different person as input, a high energy is the result. The loss functions needs to be designed in such a way, that the energy function is decreased or increased, if a pair of faces match.

In May 2012, the first international challenge on 2D segmentation of neuronal processes in EM images was launched [4]. The goal of the challenge is to create an accurate boundary map to the given greyscale electron microscopy (EM) images and to attract researchers to the field of connectomics.
For segmentation, the main characteristics of cells need to be handled properly. Cells touch and occlude each other, heaps of cells occur and flowing transitions between cell states and cell functions exist. Furthermore, EM images have low contrast and can contain noise. The ground truth of the training images is created by a human expert. Therefore, the ground truth itself can contain errors in comparison to the biological reality. To reduce the human error, two different specialists separately draw the boundary maps for the test set, with different software tools. The final labels are created as a consensus between the two boundary maps. Disagreements (for example a split or merge error) are manually corrected to guarantee the 3D object continuity.
Participants are given a training data set, which contains a set of 30 sequential sections from a serial section Transmission Electron Microscopy (ssTEM) data set of the Drosophila first instar larva ventral nerve cord (VNC). The corresponding ground truth boundary maps of the 30 images is made available too. The boundary maps contain binary labels. A white pixel label (with value 1) corresponds to a pixel inside a cell, while a black pixel label (with value 0) corresponds to a boundary pixel.
Each greyscale image from the training and test set has a size of 512×512 . The 30 test images are made available without ground truth and are used as evaluation for the ISBI challenge [4].
To measure the segmentation accuracy, three different metrics were used. Minimum splits and mergers warping error, foreground restricted rand error and pixel error. After evaluating the results of the ISBI 2012 challenge, the authors proposed new measures for segmentation accuracy. Currently, the metrics used for the leader boards are Foreground-restricted Rand Scoring and Information Theoretic Scoring.

In 2016, the Densely Annotated Video Segmentation (DAVIS) first started [39], which created the first dataset specific for video object segmentation. The goal of the authors was to create a dataset, which combines methods related on image segmentation and object recognition. For this task, fifty professionally annotated high-resolution Full HD videos are made available. Each frame is annotated professionally at pixel-level. Furthermore, each video is annotated with a major problem of the sequence, namely occlusions, fast-motion, non-linear deformations or motion-blur. As a result, problems with different aspects of the videos can be easily identified. Each two to four second long video sequence, contains a single object, or spatially connected objects. Therefore, the focus lies on detecting the object with the significant motion. State-of-the-art algorithms using unsupervised, semi-supervised and supervised approaches are evaluated and analysed. Region similarity, contour accuracy and temporal stability are used as metrics for the evaluation.

3.2 Chapter Summary

In this chapter, the success of CNNs in the ImageNet challenge for image classification is explained. After that, the focus is shifted to semantic segmentation. The FCN [33] is explained in detail, which is used as baseline to produce a wide array of similar networks. Several networks, which use parts of FCN as a baseline are explained in detail. In Chapter 5, these networks are used to compare our adaptation of the U-Net architecture called SiamU-Net, with the ISBI 2012 challenge results. Then, the ISBI 2012 challenge and the EM dataset is introduced, which is still open for participants. Finally, the DAVIS 2016 challenge and the containing dataset is discussed.

CHAPTER 4

Methodology

In biomedical image processing, only little datasets are receivable or available [10]. Therefore, the architecture of neural networks differs, depending on the task it should be applied to. O. Ronneberger, P. Fischer and T. Brox [10] describe the U-Net architecture, which is specifically developed for biomedical image segmentation. Key aspects of the network are the fast training, the use of data augmentation to enlarge the used datasets and the concatenation of higher resolution features in the up-sampled path, to enable precise localization. As mentioned by O. Ronneberger, P. Fischer and T. Brox [10], elastic image transformations are a key concept, when working with very few annotated images, like in the ISBI 2012 challenge. We propose a novel method called Elastic Gradient Transformation (EGT), which introduces image dependent semi randomness, based on the image gradient. Additionally, to expand the U-Net network, we propose a structure to learn spatio-temporal information. The new network is called SiamU-Net. As a design choice, we do not use transfer learning. We avoid the dependency on already trained models, and make our results comprehensible and reproducible.

4.1 Network Architecture

Parts of the U-Net architecture can be seen in many different networks as stated in Chapter 3. In 2018, an adaptation of the U-Net architecture won the Kaggle 2018 Databowl challenge for nucleus detection [12]. Despite the success of the U-Net, it lacks the ability to learn spatio-temporal image information. On little sequential image datasets, like this ISBI 2012 dataset with 30 images, the use of the temporal image information can have an important impact on the segmentation result. In this thesis, we want to resolve the issue of the U-Net, and exploit spatio-temporal image information with the help of a second encoding path. We call the resulting architecture SiamU-Net.
In the following, a short introduction to the structure of the base U-Net is given. Figure 4.1 shows an illustration of the U-Net architecture. It comprises four contracting levels,

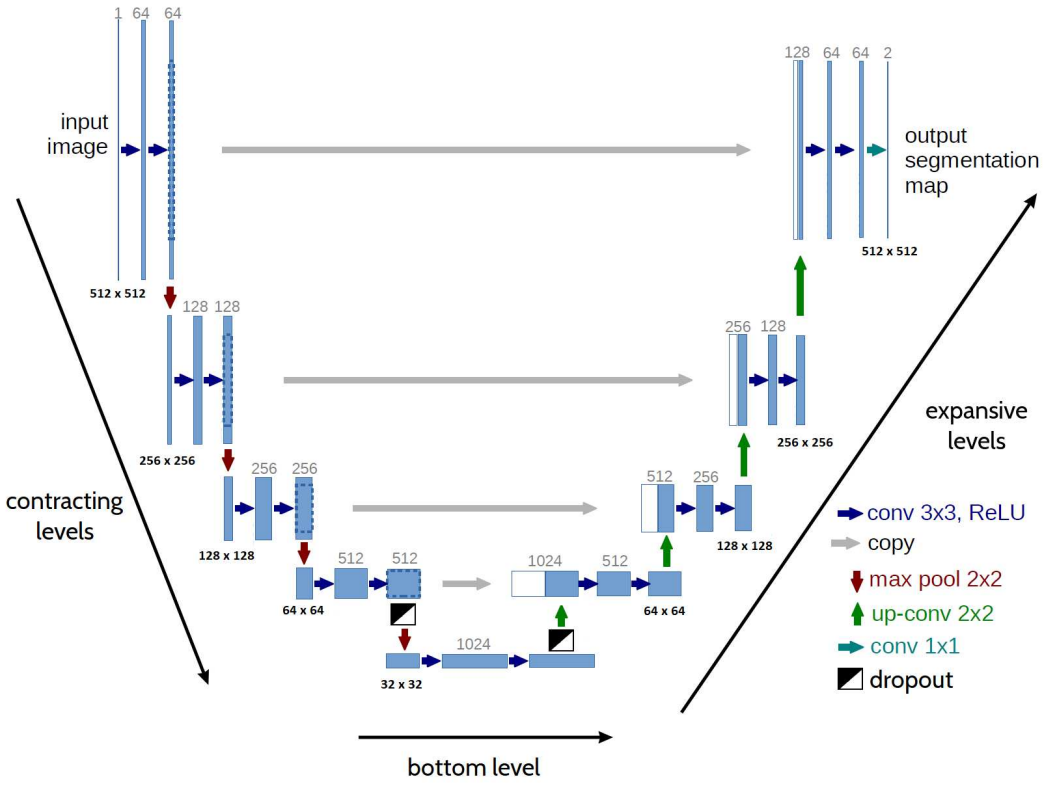


Figure 4.1: Original U-Net architecture proposed by O. Ronneberger, P. Fischer and T. Brox [10]. In comparison to Ronneberger et al. this U-Net implementation uses padding to prevent a tiling of the input image. Image adapted from [10]

.

one bottom level and four expansive levels. Every level of the contracting path consists of two convolution layers with a kernel size of 3×3, each followed by a rectified linear unit (ReLU), and a a max-pooling layer with a kernel size of 2×2 and a stride of 2. The idea behind stacked convolutions is derived from K. Simonyan and A. Zisserman [28]. Instead of only using a large filter, two small filters are used in a row, to create the same receptive field, as the large filter would have. Furthermore, this method generates a more discriminative function and the amount of learnable parameters is reduced. Before each down-sampling step, the output feature maps of the last convolution is additionally saved for later use. The number of feature channels is doubled after each down-sampling step in the first following convolution. In comparison to the original implementation of the U-Net [10], the proposed implementation uses reflection padding at each convolution layer, to create an output of the same size as the input.
There is a connection between the input image size and the number of used feature maps.

For a 512×512 input image, where 64 feature maps are calculated in the first convolution with a size of 512×512, the first convolution in the bottom layer calculates 1024 feature maps with an output size of 32×32. The number of feature maps in the last convolution in the bottom layer corresponds with the number of neurons in a single feature map of size 32×32. Therefore, in the bottom layer, for each feature, one feature map exists. To avoid excessive cropping, the size of the feature maps at each down-sampling step has to be even, to generate a matching up-sampling. For uneven input data, zero padding can be added to the data.
The bottom layer follows the same structure as in the contracting path, but instead of using a down-sampling layer, an up-sampling operation is used, which halves the number of feature channels. A 2×2 transposed convolution layer, with a stride of two, handles the up-sampling, followed by a ReLU. In addition, a dropout layer with 50% probability is added before the last max-pooling layer and before the first up-sampling layer. In the expansive path, before each first convolution, the saved feature maps from the corresponding contracting level are concatenated with the current feature maps, doubling the number of feature maps. Now, the first convolution halves the amount of feature maps again, as seen in Figure 4.1 on the expansive path. If no padding is used, the corresponding feature maps of the contracting layer have to be cropped to match the size of the feature maps of the expansive path. This skip connections combine low level, fine-grained feature maps from the expansive path with the semantic, coarse-grained feature maps from the contraction path.
After the final up-sampling, two convolutions are followed by a final convolution layer, which uses a kernel size of 1×1, to map all features to the number of defined classes. In total, the network contains 19 convolutions, four transposed convolutions, four max pooling layers and two dropout layers.

SiamU-Net

In the ISBI 2012 dataset, a spatio-temporal relation between the images exists. To improve the semantic segmentation, we propose an extension to the U-Net architecture. Instead of a single greyscale input image x_{t+1} at time $t+1$, the network needs to process an additional input x_t to calculate the segmentation map of the image x_{t+1}. The contracting path is mirrored for the additional input image x_{t+1} in a separate encoding path, without sharing weights. Therefore, eight additional convolutions layers, and four additional max-pooling layers are added to the network. The two encoders are fused back together in the latent space. Therefore, each encoding path generates 512 feature maps after the last pooling layer. Both feature maps are concatenated to form a tensor of 1024 feature maps. Before the first up-sampling step, two convolution layers are used to create a dependence between both encoding paths. Instead of concatenating low-level or mid-level features of the first three contracting levels, we want to associate high-level features of both images with each other. The reason for this approach is based on our observation of how human experts label datasets. To label biomedical data, first the whole image is looked at to capture context, as mentioned by J. Chen et al.

[14]. After that, the image is zoomed in for precise labelling. Another observation is that consecutive images change radically locally due to the small sampling size. As a result, we concatenate the features with the largest receptive field in our network. After the mentioned concatenation, the baseline U-Net architecture is followed again. In the expansive path, only skip connections from the contracting path of image x_{t+1} are used to calculate the segmentation map of input image x_{t+1}. Figure 4.2 shows the architecture of the SiamU-Net. The red rectangle shows the contracting path, which is added to the U-Net architecture to exploit spatio-temporal image information from the input image x_t.

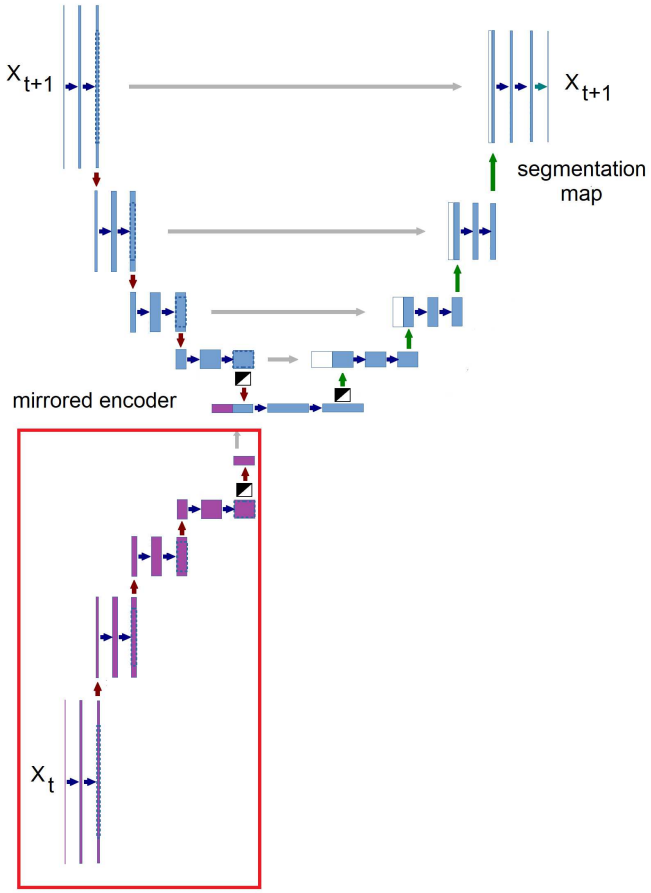


Figure 4.2: SiamU-Net architecture proposed in this thesis. The additional branch is marked with a red rectangle. In comparison to the U-Net by O. Ronneberger, P. Fischer and T. Brox [10] an additional mirrored encoder path is used, to exploit spatio-temporal information of image x_t. Image adapted from [10]

.

The idea to use an additional encoding path without sharing weights is influenced by the

kU-Net architecture by J. Chen et al. [14]. Instead of using multiple U-Nets behind each other, the proposed solution generates a split in the encoding path, resulting in a side by side path.

In our networks, the weights are initialized as mentioned by O. Ronneberger, P. Fischer and T. Brox [10]. The weights are drawn from a Gaussian distribution, with a standard deviation of $\sqrt{\frac{2}{K*N}}$. N corresponds to the number of incoming channels in the convolution layer and K is equal to the sum of elements in the kernel. For a 3×3 convolution with 128 incoming channels $K * N$ is equal to $9 * 128 = 1152$.

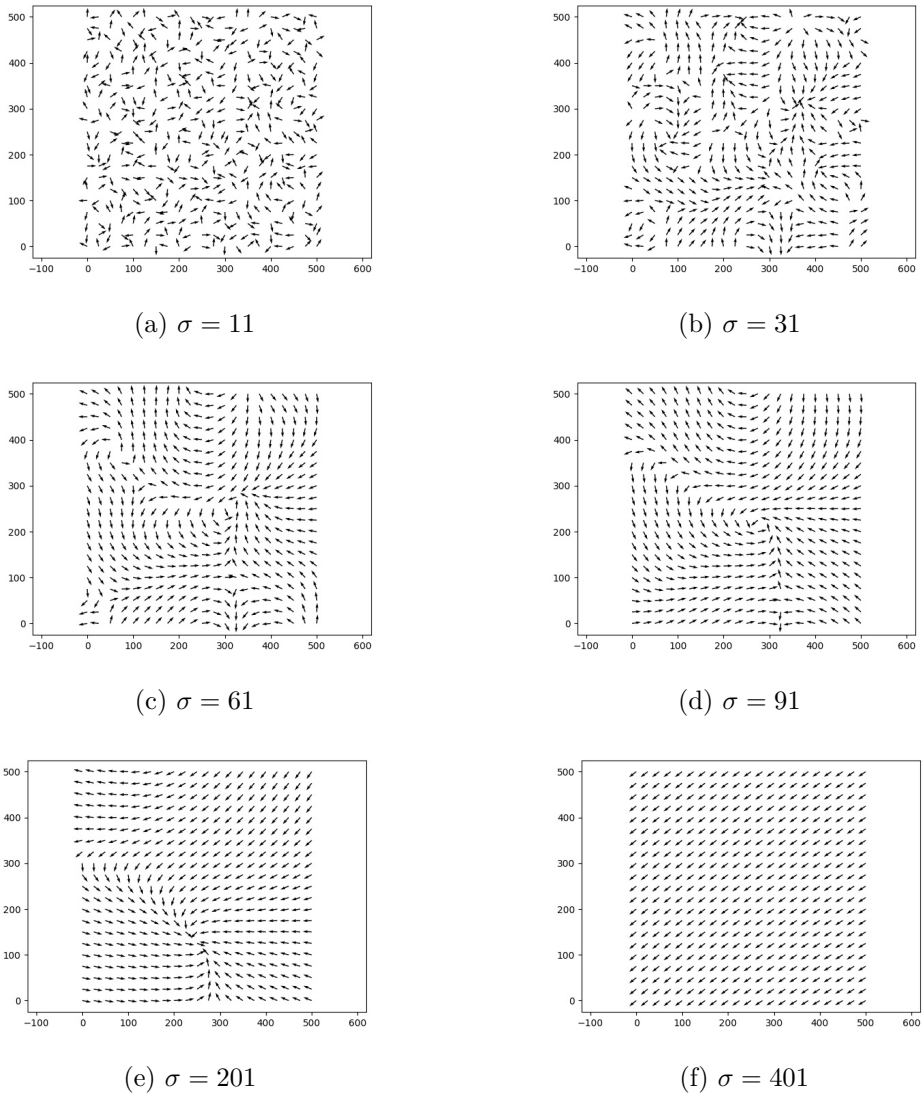
4.2 Elastic Image Transformation

For the task of biomedical image segmentation, publicly available datasets contain only a limited amount of annotated images [10]. Due to the high costs and time-consuming task of biomedical image annotation by experts, elastic image deformations are used to increase the size of datasets artificially. O. Ronneberger, P. Fischer and T. Brox highlight elastic random transformations as a key concept for the success of the U-Net [10]. Besides two common methods called random normalized and random scaled transformation, we propose a novel method called elastic gradient transformation, which introduces image dependent semi-randomness based on the image gradient. In the following, the steps to calculate elastic image transformation are explained in detail.

Random Normalized Transformation

First, we need to create a random field for the x and y direction with the same size as the input image. Each field contains uniform distributed random numbers between -1 and 1. To both random fields, a Gaussian filter is applied. The filter size is calculated with respect to σ as shown in Equation 4.1 . For large values of σ, random values are close to zero in both field directions. The random fields are combined to create a displacement vector field. After normalizing each displacement vector to a norm of 1, σ can be used to manipulate the field structure. Small values of σ create random displacement fields. Choosing large values for σ let the displacement field look constant. Intermediate values of σ can create elastic looking deformations. Figure 4.3 shows vector fields with different σ values. To control the intensity of the displacement, a scaling value α is introduced, which is multiplied with the scaled displacement vector field [40].

$$FilterSize = 2 \cdot round(3 \cdot \sigma) + 1 \tag{4.1}$$



(a) $\sigma = 11$ (b) $\sigma = 31$

(c) $\sigma = 61$ (d) $\sigma = 91$

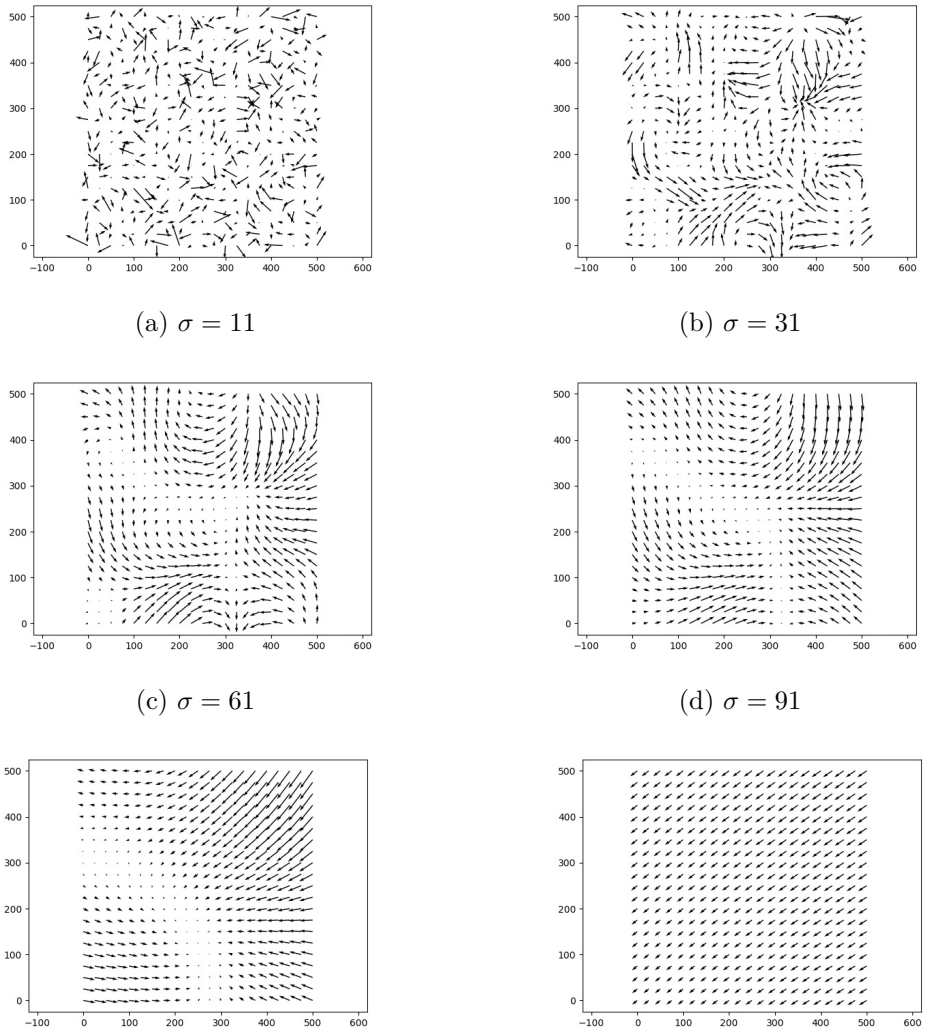
(e) $\sigma = 201$ (f) $\sigma = 401$

Figure 4.3: Random Normalized Transformation (RNT) applied to an image of size 512×512 with changing σ values. Every 25th vector is shown. Low values of σ create random fields, while high values of σ generate constant fields. Medium values create elastic looking displacement fields. For scaling, the field is multiplied with a scaling value α, after normalizing each displacement vector to length 1.

Random Scaled Transformation

Instead of normalizing each displacement vector to a length of one, like mentioned in the normalized transformation, the displacement field is multiplied with a scaling value α directly. The effect can be seen in Figure 4.4. Values, close to zero, do not impact the

displacement, while long vectors have a high impact on the resulting transformation. To decrease the impact of outliers, instead of using a uniform distribution, the random values are generated from a Gaussian distribution. An advantage of the scaled transformation method is the ability to simulate smooth waves, which avoids hard edges in images.



(a) $\sigma = 11$ (b) $\sigma = 31$

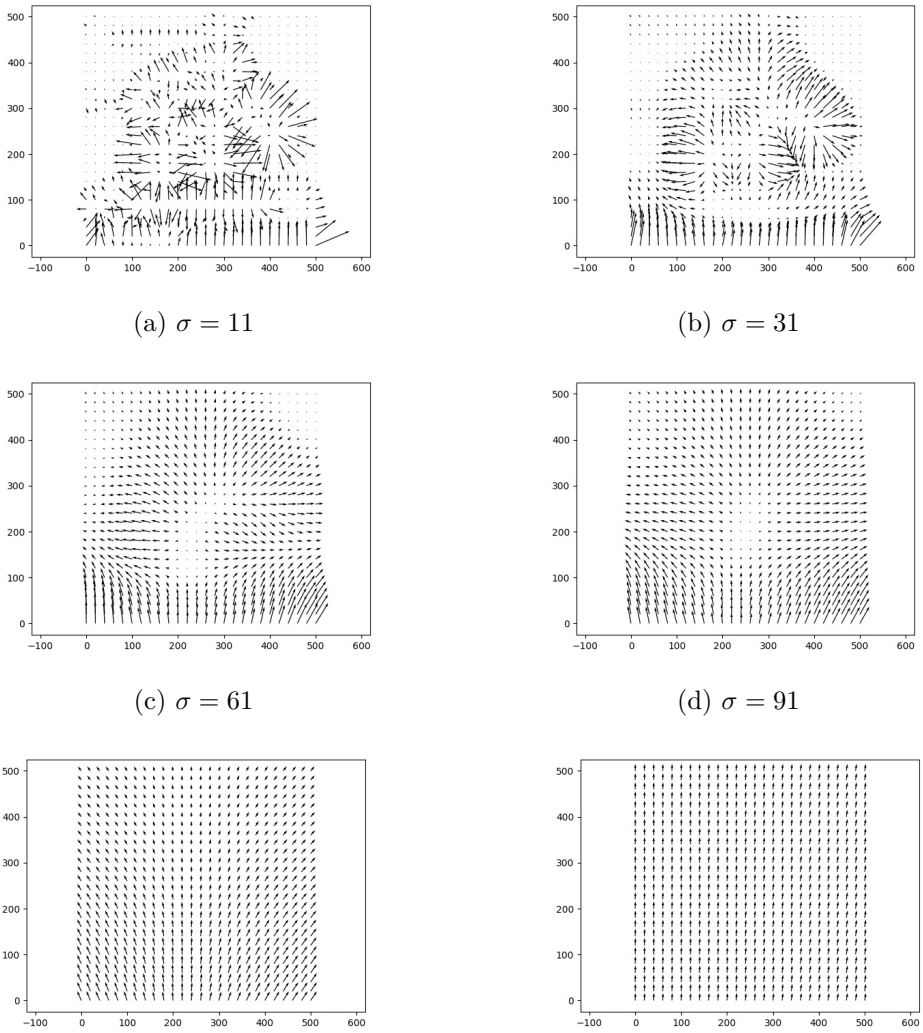
(c) $\sigma = 61$ (d) $\sigma = 91$

(e) $\sigma = 201$ (f) $\sigma = 401$

Figure 4.4: Random Scaled Transformation(RST) applied to an image of size 512×512 with changing σ values. Every 25th vector is shown. Low values of σ create random fields, while high values of σ generate constant fields. Medium values create elastic looking displacement fields. For scaling, the field is multiplied with a scaling value α. No vector normalization is applied.

Elastic Gradient Transformation

To guide an elastic transformation, we propose a novel method called Elastic Gradient Transformation (EGT). To show noticeable and comprehensible effects of this method, the image of a microscope, as seen in Chapter 2, is used.



(a) $\sigma = 11$ (b) $\sigma = 31$

(c) $\sigma = 61$ (d) $\sigma = 91$

(e) $\sigma = 201$ (f) $\sigma = 401$

Figure 4.5: Elastic Gradient Transformation (EGT) applied to an image of size 512×512 with changing σ values. Every 25th vector is shown. High values of σ generate constant fields. Medium values create elastic looking displacement fields, which depend on the structure of the image. To introduce randomness, the field is multiplied with two random scaling factors α and β. No vector normalization is applied.

As a first step, we calculate the gradient dx and dy of the image. The gradient matrix values are scaled to be between -1 and 1. Next, a Gaussian filter is applied to the gradient fields. This step is needed to reduce image noise. Furthermore, a threshold range for the Gaussian filter needs to be set. Hard edges of the gradient need to be avoided. For images from the ISBI challenge, a value between 70 and 90 worked best. To emulate randomness, two scaling factors, which are randomly chosen between two thresholds, are needed. Scaling factor α is multiplied with the Gaussian filtered dx gradient field, while scaling factor β is multiplied with the Gaussian filtered dy gradient field. No universal applicable threshold values exist. The correct values for the thresholds need to be tailored to the underlying image data. Figure 4.5 shows the elastic gradient fields with $\alpha = \beta$ and no vector normalization. The effect of different Gaussian filter values can be seen in Figure 4.5. In this example, the threshold value of α and β is equal. When changing the value of the scaling factors and the Gaussian filter randomly, the direction and intensity of the vectors can be manipulated.

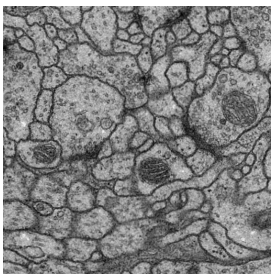
4.3 Datasets

Two datasets are used to evaluate the performance of the proposed SiamU-Net, in comparison to an U-Net architecture. For the ISBI 2012 challenge, the dataset contains a set of sequential sections from a ssTEM. The goal is to find an accurate boundary map, to distinguish the membrane from non-membrane elements at pixel-level. The second dataset from the DAVIS 2016 challenge contains fifty professionally annotated videos, which last between two and four seconds. The first frame of each video is given, and a segmentation of the following images is needed.
In the following, the needed pre-processing steps to adapt the datasets to our network and the evaluation metrics are highlighted.

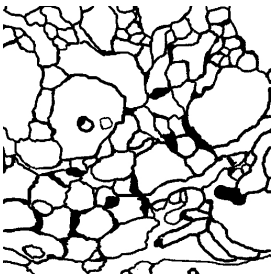
ISBI 2012

For this challenge, three tif files are present. Challenge, ground truth and train files. Each file contains 30 images of size 512×512. It is to note that, no separate validation images are present. Therefore, the participant has to pick the validation images from the trainings data on his own choosing. The TrakEM2 plug-in of the FIJI framework [41] was used for the annotation of the ground truth. More detailed information about the image data which shows the Drosophila Brain is found in the work of A. Cardona et al. [42].
The ground truth contains two classes. White pixels have a label value of 255 and correspond to non-membrane elements. Black pixels are labelled with a 0, and have 100% membrane certainty. Figure 4.6 shows two example images from the training set with the corresponding ground truth.
We wrote a custom python script, which helps us to separate the images from the single tif file. To match temporal corresponding images during training for our SiamU-Net, images are numbered from 0 to 29 in temporal sequence. We randomly shuffle the images for training, while still matching corresponding images for the input to our network. For our

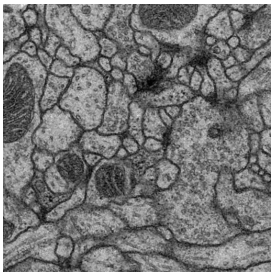
implementation, every third image is used for validation. The SiamU-Net calculates the segmentation of image x_{t+1}, therefore all validation images can be used in the additional encoder path as input x_t. In addition, every image without a predecessor is part of the validation set (like the first image of the sequence).
To save processing time, data augmentation can be applied as a preprocessing step. In our setup, rotation, flipping and elastic image transformation is applied at runtime. In each training iteration, the same data augmentation is applied to both of the input images. This helps the network to maintain the spatio-temporal image information of two consecutive images.



(a) train image 1



(b) ground truth 1



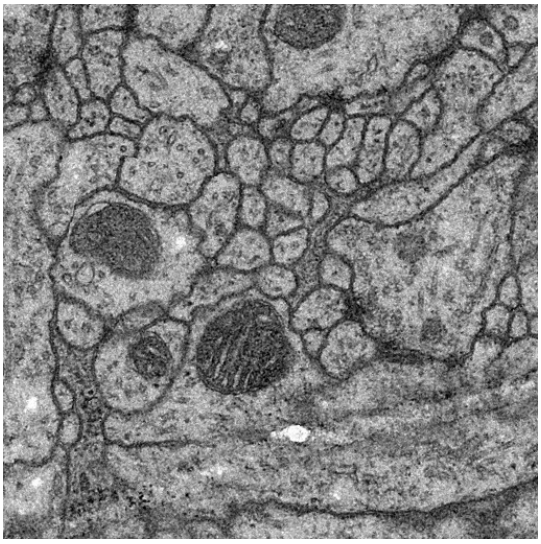
(c) train image 9

(d) ground truth 9

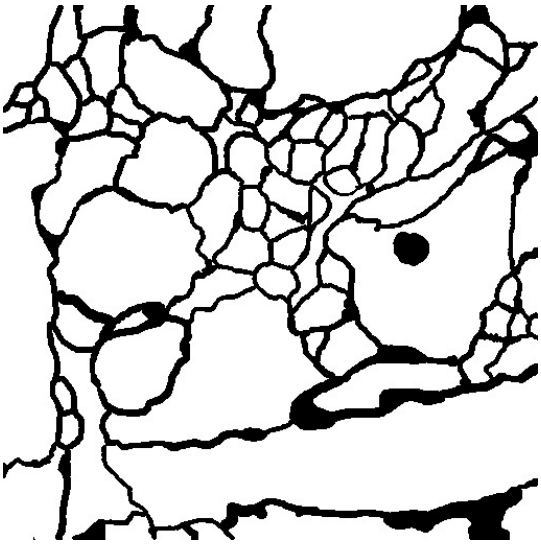
Figure 4.6: Two images with corresponding ground truth from the ISBI 2012 challenge dataset [4]. White pixels in the ground truth belong to segmented objects, while black pixels correspond mostly to membranes. The ground truth is generated by two experts, who independently segmented the data, with two different software programs. The final ground truth was created by a consensus of both expert segmented boundary maps. It is stated that the ground truth itself contains errors, relative to the underlying biological reality. Image reproduced from [4]

.

Some images of the dataset contain visible errors. However, we do not use any preprocessing to handle those. Figure 4.7 shows an image with multiple white stains and the corresponding ground truth image. White stains, as can be seen in the bottom and left part of the Figure 4.7a, are not processed in a special way by out network. It uses the unaltered images from the official available dataset.



(a) Validation image



(b) Ground truth

Figure 4.7: One image from the validation set with the corresponding ground truth. These images can contain errors. White stains, as can be seen in the bottom and left side of image (a) are present in multiple images of the trainings dataset.

DAVIS 2016

The dataset is structured into three folders, Annotations, JPEGImages and ImageSets. Annotations contains a folder for each video, with the ground truth images of each frame in png format. JPEGImages shares the same structure, but contains RGB images in a

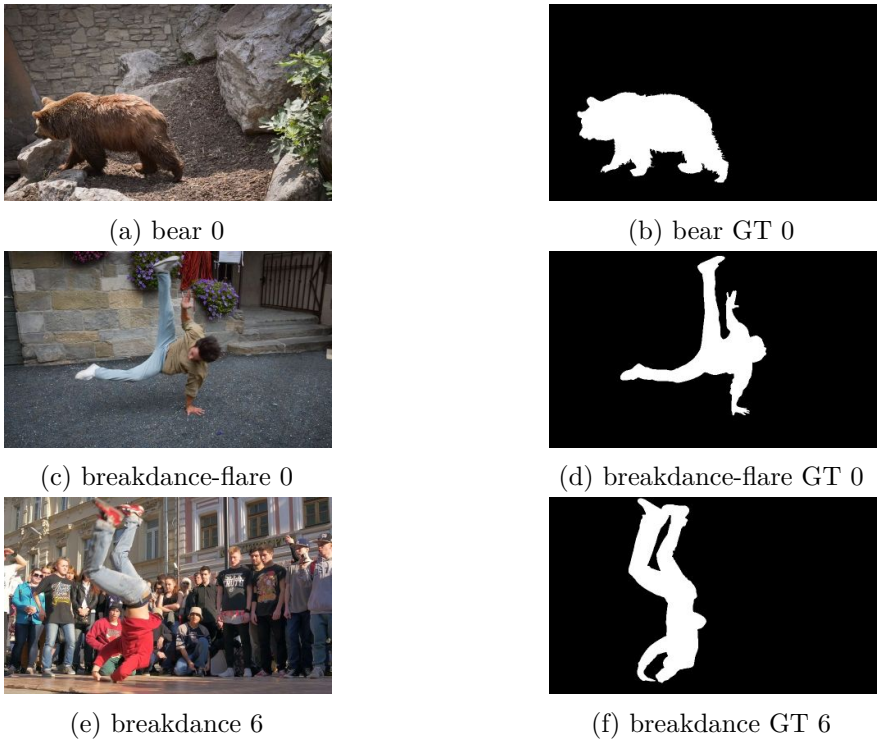
jpg file format. A partition into train and validation, is already predefined in a txt file in ImageSets. Each line of the txt file contains the location of an RGB image frame and the corresponding ground truth in sequential order for the used videos. Image bear/00077.png in Annotations contains an alpha layer, which is in a different format than the rest of the ground truth images. The alpha channel is removed as a pre-processing step. Instead of working with the 1080p images, a cropped version of the 480p images is used to decrease the trainings time and avoid image cropping. The final size of an image is 400×240, and is converted to grayscale. For real life images, as contained in the DAVIS 2016 dataset, elastic image transformation can not be used to enlarge the image count. An elastic transformation will change the shape and structure of an object. Contracting or expanding an image at random positions introduces errors, which reduce the segmentation quality. Furthermore, the perspective of the images is important. Videos are taken from popular streaming sites, and therefore do not contain mirrored or upside down videos. On this account, only horizontal flipping is applied to the training videos. The data for training contains 2079 images, from 30 videos, while the validation set contains 1376 images from 20 videos.
Three different images from the challenge are shown in Figure 4.8. The goal of the DAVIS 2016 challenge is to segment the object in significant motion, separating it from the background. Even if multiple objects of the same type exist, for example other people in the background, only one objects must be detected.

For the DAVIS 2016 challenge, a semi-supervised approach is used. Therefore, the first ground truth image of each video is available to the network while training. There are two ways to incorporate the semi-supervised approach. The first approach would move the first image and ground truth from each validation video to the trainings data. In this thesis, a different approach is used. Before inference, we apply another training to our network, which only trains on the second ground truth image of the corresponding validation video. This helps the network to specialize on the video. This step is necessary due to the structure of our SiamU-Net. A predecessor of each image is needed to generate a segmentation. However, the SiamU-Net still only is allowed to see one image with corresponding ground truth from the validation. Therefore, our model after training needs an additional training initialization for each validation video.

Challenge Metrics

The main goal of the ISBI 2012 challenge [4] is to find an accurate boundary map. A simple measure is the pixel error. This method only classifies, if a given pixel is correctly detected as a boundary pixel. The inability to detect merge errors, is a drawback of this method, as mentioned by I. Arganda-Carreras et al. [4]. If a single pixel is missing between a border, only a small pixel error occurs, but a merge error between two cells occurs.

In connectomics, defining a meaningful scoring for boundary maps is a non-trivial task. In



(a) bear 0 (b) bear GT 0

(c) breakdance-flare 0 (d) breakdance-flare GT 0

(e) breakdance 6 (f) breakdance GT 6

Figure 4.8: Three images with the corresponding ground truth (GT) from the DAVIS 2016 challenge dataset [43]. White pixels in the GT belong to segmented objects, while black pixels correspond to the background. The GT is manually labelled by experts at pixel-level.

real world applications, experts use boundary detection algorithms as a base segmentation and correct the missing or wrong parts themselves. The manual labour needed to correct the segmentation should impact the used metric. Computing such a metric automatically is difficult. The tools used by the experts and the experts themselves have an impact on quality of the segmentation. Therefore, in the ISBI 2012 challenge an approximation of such a metric is proposed. Split errors and merge errors can be detected by a metric. Both errors need human effort to be corrected. If one neuron is incorrectly split into two neurons or two neurons are merged into one, an expert needs to correct the boundary map manually. Therefore, a new metric is introduced in the challenge to accommodate the special needs in biomedical imaging, which is described in the following.

One can transform a boundary map into a segmentation by finding the connected components. First, a rand split score is defined in Equation 4.2. S is the predicted segmentation and T is the segmentation of the ground truth. p_{ij} can be defined as the probability that a randomly chosen pixel is part of segment i in S and segment j in T. $t_j = \sum_j p_{ij}$ is the probability of a randomly chosen pixel to belong to segment j in T. The numerator in Equation 4.2 , is the probability, that two randomly chosen pixels belong to the same

segment in both S and T. The denominator is the probability that two pixels belong to the same segment in T. The whole equation can be seen as the probability of two randomly chosen pixels belonging to the same segment in S, with the condition that they belong to the same segment in T. A high rand split score, corresponds to fewer split errors [4].

$$V_{split}^{Rand} = \frac{\sum_{ij} p_{ij}^2}{\sum_k t_k^2} \tag{4.2}$$

The rand merge score is defined in Equation 4.3. $S_i = \sum_i p_{ij}$ is the probability of a randomly chosen pixel to belong to segment i in S. The Equation 4.3 can be described by the probability of two randomly chosen pixels belonging to the same segment in T, with the condition that they belong to the same segment in S. Fewer merge errors increase the merge score [4].

$$V_{merge}^{Rand} = \frac{\sum_{ij} p_{ij}^2}{\sum_k s_k^2} \tag{4.3}$$

With the weighted harmonic mean, both scores can be joined, to include both split and merge errors. If $\alpha = 0.5$, split and merge errors are weighted equally [4]. The official ranking metric for the competition is Foreground-restricted Rand Scoring and seen in Equation 4.4. Borders between cells are often described differently by humans and algorithms. Therefore, border pixels in the ground truth boundary map are excluded in the calculation of the scores.

$$V_{\alpha}^{Rand} = \frac{\sum_{ij} p_{ij}^2}{\alpha \sum_k s_k^2 + (1-\alpha) \sum_k t_k^2} \tag{4.4}$$

A script to evaluate the training set is published at the website of the ISBI 2012 challenge [44]. The script evaluates a given ground truth and the resulting output segmentation map of the network.

Information Theoretic Scoring is an alternative scoring system which was added later to the competition. The ranking depends solely on the Foreground-restricted rand scoring value. The Information Theoretic Scoring values are added for the sake of comparison. Both methods are explained in detail in the main publication about the competition, by I. Arganda-Carreras et al. [4].
The DAVIS 2016 challenge uses a different metric to test the segmentation accuracy. For a resulting segmentation mask M, we want to answer the question of how well it fits the given ground truth mask G. The Jaccard index $\mathcal{J}$ was first introduced in PASCAL VOC2008 [45]. The Jaccard index $\mathcal{J}$ is defined in Equation 4.5 as the intersection over union. The intersection of $|M \cap G|$ calculates the sum of all labels in M, which overlap with the ground truth G. The union $|M \cup G|$ produces the sum of all distinct labelled pixels of both maps. Therefore, the Jaccard index provides an information about

the similarity of the segmentation to the ground truth and is used in the DAVIS 2016 challenge.

$$\mathcal{J} = \frac{|M \cap G|}{|M \cup G|} \tag{4.5}$$

Network Evaluation Metrics

For both datasets, only two classes exist. Therefore, the output of our networks can be represented by a probability map. To generate a binary labelling out of it, a threshold is applied. All probabilities below the defined threshold are mapped to the label zero, while the rest is mapped to label one. The label one (white pixel) corresponds to class non-membrane or object. It represents the true output. The label zero (black pixel) belongs to the class membrane or background and represents the negative output.

When looking at the ground truth and corresponding network segmentation for the DAVIS 2016 dataset, it is easy for non experts to detect a workable solution. In connectomics, it is difficult to link the generated segmentation map with a corresponding image. Expert knowledge in connectomics would be needed to declare, if a segmentation map is useful or meaningless. Nevertheless, the following metrics can still be used to detect problems during the learning phase of the CNN. In the following, we use the term *positive label* for class label 1 , while a *negative label* is equal to label 0. TP, TN, FP and FN correspond to true positive, true negative, false positive and false negative.
In addition to the above mentioned challenge metrics, we are using precision, recall, F1 score, accuracy and the false-positive rate (fpr) to evaluate our network.
Equation 4.6 shows the calculation of *Precision*. It can be formulated as: of all the predicted positive labels, how many of them are actual positive labels.

$$Precision = \frac{TP}{TP + FP} \tag{4.6}$$

Recall calculates how many of the actual positive labels are predicted as positive labels, as seen in Equation 4.7.

$$Recall = \frac{TP}{TP + FN} \tag{4.7}$$

Equation 4.8 shows the metric of *Accuracy*. In contrast to recall and precision, which only compares positive labels, accuracy shows the percentage of correctly classified elements in the segmentation.

$$Accuracy = \frac{TP + FP}{TP + FP + TN + FN} \tag{4.8}$$

The *FalsePositiveRate* is needed to confirm a high *precision* value. It calculates how many of the actual negative labels are predicted as positive labels. This is important if

there is a high-class imbalance between positive and negative labels. It can be thought of as the probability of a false alert.

$$FalsePositiveRate = \frac{FP}{FP + TN} \tag{4.9}$$

$$F1 - Score = \frac{2TP}{2TP + FP + FN} = \frac{precision * recall}{precision + recall} \tag{4.10}$$

Finally, the *F1-Score* is shown in Equation 4.10. F1 score is the harmonic mean of precision and recall, and can be used to set both metrics in context. The importance of the *F1-score* results from the ability to work with unbalanced classes.

4.4 Implementation

In CNNs, millions of parameters need to be trained and processed in each training step. For a fast implementation in frameworks, it is necessary to use GPUs with large memory bandwidth. In recent work, up to 27,600 Nvidia V100 GPUs are used in distributed training [46]. In the area of CNNs for image segmentation, CPU clusters are superseded by the use of GPU clusters. Nvidia plays a leading role to advance CNNs with the development of Nvidia Cuda and the cuDNN library.

Hardware

A main concern for our hardware setup is research comprehensibility. It is important to generate results which are testable, for as many researchers as possible. Therefore, the proposed setup uses consumer grade hardware. As CPU, we use an Intel i7-7700K with four cores at 4.20 GHz. Our mainboard is equipped with 8 GB DDR4 random access memory in dual channel, a 128 GB SSD for fast access of applications, and a 1.5 TB HDD to save the network models and network outputs at different stages. To benefit from CUDA Toolkit 10 and The Nvidia CUDA Deep Neural Network library (cuDNN) implementations , we use a KFA2 GeForce GTX 1070 EXOC which has 8 GB GDDR5 memory. It is possible to accumulate gradients and generate huge batch sizes on low end hardware too. However, current implementations of batch normalization layers are not compatible with gradient accumulation in PyTorch.

Software and Frameworks

Ubuntu 18.04 LTS Linux distribution is chosen as operating system. It supports Nvidia drivers and all popular machine learning frameworks. Therefore, we use Nvidia libraries of CUDA Toolkit 10 and cuDNN v6 to profit from high-performance GPU acceleration. The original architecture of the U-Net by O. Ronneberger, P. Fischer and T. Brox [10] is implemented in Caffe [11]. The development of Caffe stopped, therefore we chose an alternative machine learning framework. We ported the U-Net architecture to PyTorch

1.0, which is a framework for machine learning with deep integration into Python. We use the python package manager conda with Python 3.7 to load and manage our dependencies for the PyTorch 1.0 Framework. All scripts to process data and to visualize output are written in Python as well.
To configure, organize and log our experiments, we use the powerful tool called Sacred by K. Greff et al. [47], which is developed by the Swiss AI Lab IDSIA. The power of sacred lies in the separation of parameters into a separate file. This configuration file can be injected into any function with the help of decorators, therefore generating a predefined place for all hyper parameters. Additionally, the command-line interface has the possibility to change the predefined parameters on each call. Sacred gives us the opportunity to log our experiments into an online mongoDB. Therefore, any change of the experiment hyper parameters and code is documented. All metric variables are defined in code and are tracked with Sacred. Besides the tracking of parameters, sacred offers to track the used configuration file, all used source files, console outputs and artefacts. Artefacts are user defined objects, which can be saved at specific times. For example, we can save the PyTorch model, every time a best loss is reached. Therefore, it is simple to organize and structure the experiments.

Implementation details

The implementation of our networks is made available on request and is tested with Ubuntu only. Our code contains the custom data loader for the ISBI 2012 and DAVIS 2016 datasets. Data augmentation is applied at runtime. Therefore, changes or additions of augmentations need to be set in the data loaders. The trained models of our networks are not made available. A single model with all weights and settings needs up to 300 to 400 MB for storage, therefore we omit to send them. Our data augmentation method is separated into an own python script, making it usable without a neural network.
All needed hyper parameters are pre-set. A change of the most used parameters is done through the config.json file. For the network, multiple different settings are available : batch size, learningrate, momentum, weight decay, epochs, class weight and classes can be changed in the config. Furthermore, we added the possibility to train on the cpu, set padding for all layers to true or false, resume from a given model, save the segmentation maps generated while training and changing the location of all saved items. The setting txtinfo maps the output from the console to a txt file. For inference, the setting evaluate needs to be set to true. All used evaluation metrics are found in the main file of the networks.
To calculate the official ISBI 2012 metrics for one train image, a script for ImageJ (Fiji) is available. We wrote a separate Python script, which imports ImageJ with the loaded evaluation script and executes it for all files in a specific path. These file paths are hardcoded and need to be adjusted depending on the used file structure.

4.5 Chapter Summary

In this chapter, the methodology of our work is presented. First, two similar methods to generate an elastic transformation are explained. To show the differences between both methods, the random fields are visualized with different parameters. Subsequently, our proposed gradient elastic transformation is presented, and the gradient transformation is shown on a test image. Afterwards, we describe the base U-Net architecture in detail. The proposed U-Net adaptation is called SiamU-Net. The SiamU-Net is able to exploit the spatio-temporal image information between two successive images. An improved segmentation quality is the result. After that, an overview of the datasets ISBI 2012 and DAVIS 2016 is given. The metrics used for both challenges is explained thereafter. Finally, a description is given about the used hardware to support research reproducibility. Furthermore, the used software and frameworks are described. We only use software which is available without any costs to support our claim of reproducibility. Important details about our implementation and project structure are described in short.

CHAPTER 5

Evaluation and Results

We evaluate the performance of our reimplementation of the U-Net and SiamU-Net to prove the following hypotheses: first, we show that our novel elastic gradient transformation (EGT) helps to generalize on small cell datasets, like the ISBI 2012 cell dataset. Second, we analyse the improvements of our novel EGT method on the U-Net architecture. Therefore, a comparison is made between two additional elastic image transformation methods. Third, we analyse the ability of the SiamU-Net architecture to use spatio-temporal image information. We show the improvements when using the additional information on two domain different datasets, namely the ISBI 2012 cell dataset, which contains sequential cell images, and the DAVIS 2016 dataset, which contains multiple different videos. Our evaluation shows, that for both datasets, the SiamU-Net generates a higher score than the U-Net architecture. On the ISBI 2012 challenge, the original U-Net architecture by O. Ronneberger, P. Fischer and T. Brox [10] is placed at rank 61, while our SiamU-Net places at rank 31 [1]. On the DAVIS 2016 dataset, the SiamU-Net achieves a 7.76 point higher Jaccard index than our U-Net implementation, which confirms the ability of the SiamU-Net to exploit temporal information.

The computational results presented have been achieved [in part] using the Vienna Scientific Cluster (VSC) [2].

5.1 ISBI 2012

The ISBI 2012 dataset contains 30 challenge images and 30 train images. The ground truth of the train images is made available at their website, while the ground truth for the 30 challenge images is kept secret. Therefore, to compare our neural network architecture

[1] http://brainiac2.mit.edu/isbi_challenge/leaders-board-new accessed: 26.6.2020
[2] https://vsc.ac.at/home/

locally, every third image of the training set is used for validation. To generate a leader board entry for the ISBI 2012 challenge, the segmentation map of the 30 challenge images needs to be converted to a single tif file and is then uploaded on the competition's website. For our networks, we use the cross entropy as a loss function. The stochastic gradient descent with a learning rate of 0.001, a momentum of 0.99, a batch size of 1 and class weights is used for training. To tackle class imbalance, we apply a weighting to the classes. Class cell is weighted with 1, while the membrane class is weighted with a 5 times higher value. These are experimental values, which are generated by comparing the amount of white pixels and black pixels in the ground truth. All evaluated networks are trained for 600 epochs, where each epoch contains 20 training images in a random order. In our architecture, replacing the ReLU activation function with sigmoid or tanh activation functions generated no significant benefit. Therefore, the ReLU activation is used after every convolution layer, and the results of networks trained with different activation functions are omitted.

Elastic Image Transformation

Data augmentation can be applied to cell images, because deformations are a common variation in tissues [10]. Realistic looking deformations can be approximated efficiently with the elastic image transformations described in Section 4.2.
O. Ronneberger, P. Fischer and T. Brox [10] use an elastic image transformation method with a random field to morph the cell images. We propose a novel elastic gradient transformation method, which is dependent on the cell structure. Instead of using a random field to generate an elastic transformation, the proposed method uses the image gradient to deform the cell structure, which leads to an improved generalization of the network.
To evaluate this hypothesis, we train four U-Net networks with different image transformations. The first network trains without any elastic transformation. Network two, three and four trains with the Random Normalized Transformation (RNT), the Random Scaled Transformation (RST) and the Elastic Gradient Transformation (EGT) respectively. The image transformation method is applied to 90% of the images at runtime. In our setup, the networks start to overfit before reaching 600 epochs. A sign of overfitting is the decreasing train loss, while the validation loss keeps increasing.

To evaluate the performance of the elastic image transformation, we train four U-Net networks, which only use an elastic image transformation method as data augmentation, and compare the results on the validation images. Table 5.1 shows the challenge metric (rand score) for the ten validation images of each trained network. The values are calculated with an official challenge script locally. Our U-Net architecture, with the EGT method, achieves the highest score in 6 out of 10 validation images. While our network with the EGT method produces a rand score greater than 0.970 points in eight images, the network with the RST method reaches a higher value in only seven images. The

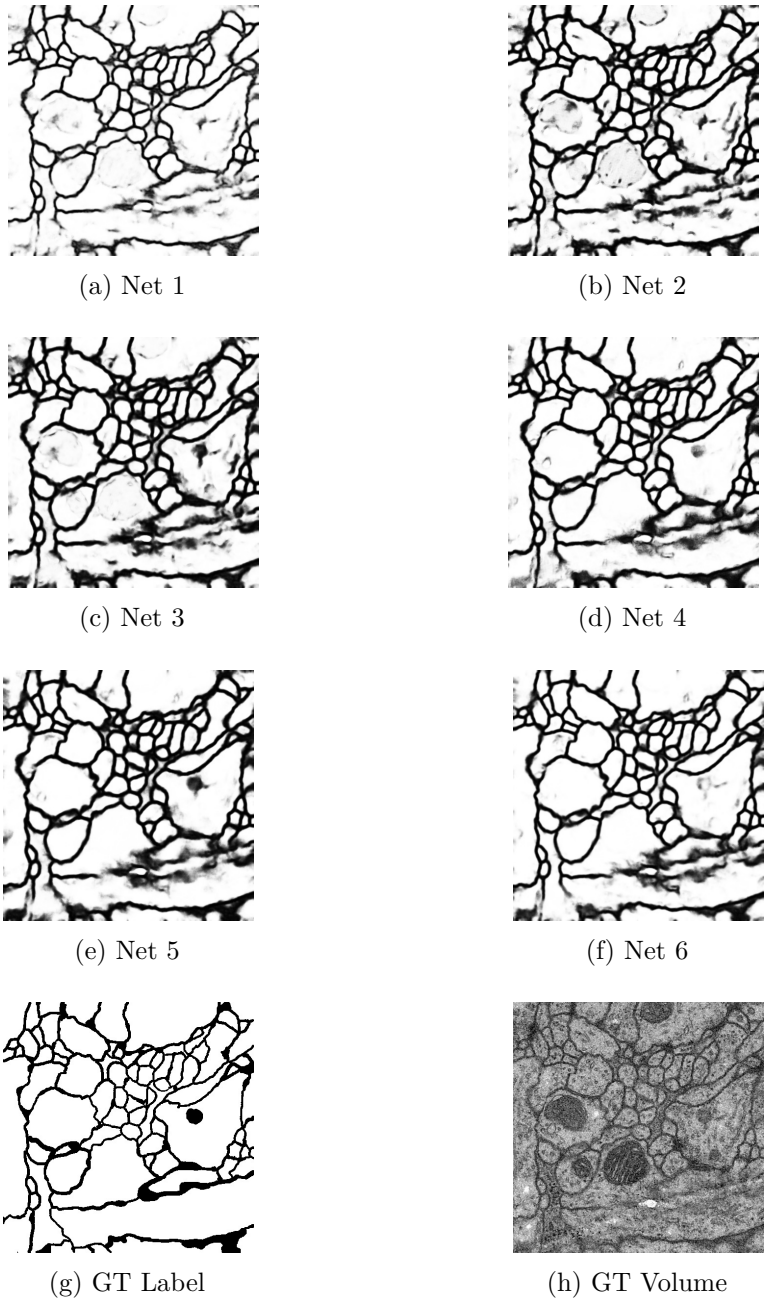
network with the RNT method falls behind the other networks, scoring above 0.970 in only six images. Despite using no data augmentation in one network, it still produces high rand scores. On Image 12, it scores the highest rand score, despite having no data augmentation at all. However, the result is not surprising. Due to the small size of the available dataset, the validation images need to be selected out of the 30 available train images. Therefore, the strong similarity of the train and validation cell images, and the sequential nature of the dataset has a remarkable impact on the validation results. Furthermore, pixel errors which do not introduce cell merges or splits, have no influence on the rand score.

	no augmentation	**RST**	**RNT**	**EGT**
Image 0	0.962	0.965	0.970	**0.971**
Image 3	0.966	0.968	0.968	**0.970**
Image 6	0.978	**0.983**	0.978	0.982
Image 9	0.985	0.985	0.984	**0.986**
Image 12	**0.994**	0.992	0.990	0.992
Image 15	0.992	0.994	0.993	**0.995**
Image 18	0.871	0.891	**0.901**	0.898
Image 21	0.967	**0.977**	0.974	0.952
Image 24	0.973	0.974	0.833	**0.975**
Image 27	0.974	0.971	0.957	**0.976**

Table 5.1: For evaluation, four different networks are trained. The resulting rand score is calculated with the official fiji script available from the ISBI 2012 challenge. In six out of ten images, the U-Net with EGT scores highest. RNT for Random Normalized Transformation. RST for Random Scaled Transformation. EGT for Elastic Gradient Transformation.

To validate the impact of the elastic image transformation methods on the ISBI 2012 challenge dataset we train another six U-Nets with different data augmentation settings. To generate competitive results, class weights, 10° rotations at random at every image and image flipping is applied. Elastic image transformation is again applied to 90% of the images at runtime. These basic augmentation settings do not differ between the networks, besides for Net 1, which utilizes no augmentation at all, and Net 2, which only uses class weights. Table 5.2 shows the best epoch, train loss, validation loss and the used data augmentation of the six networks. Net 1 and Net 2 apply no data augmentation at all. As a result, these networks start to overfit before 200 epochs. As can be seen in Table 5.2 column Best Epoch, elastic image transformations and rotations, delay the overfitting. Net 6 trains three times more epochs than Net 1 or Net 2, without overfitting. The low train and validation score of Net 1 can be explained by the lack of using class weights. As a result, the five times higher occurrence of white pixels than black pixels in the ground truth, impacts the loss function.
The best scoring network while training is Net 6, with our novel EGT method. The

lowest training loss, in combination with the lowest distance between the train and the validation loss, proves the positive impact of our EGT method on the network.



(a) Net 1 (b) Net 2

(c) Net 3 (d) Net 4

(e) Net 5 (f) Net 6

(g) GT Label (h) GT Volume

Figure 5.1: The images show the probability map of an image from the validation set for the six trained networks. In the last row, the ground truth (GT) and volume image is shown.

Out of the three elastic image transformation methods, EGT scores best, while RNT performs worst again.

	Best Epoch	Train Loss	Val Loss	Augmentation
Net 1	167	0.178	0.193	-
Net 2	148	0.203	0.231	W
Net 3	234	0.201	0.222	W, R, F
Net 4	527	0.223	0.203	W, R, F, RNT
Net 5	336	0.205	0.219	W, R, F, RST
Net 6	438	0.195	0.209	W, R, F, EGT

Table 5.2: Best epoch and best loss values for six U-Net networks with different data augmentations. After the best epoch is reached, each network starts to overfit. A sign for the overfitting is the decreasing train loss while the validation loss increases. Symbols for adjustments: W for class weights. R for image rotation. F for image flipping. RNT for Random Normalized Transformation. RST for Random Scaled Transformation. EGT for Elastic Gradient Transformation.

For all six networks, the resulting segmentation map of one validation image is shown in Figure 5.1. The ground truth and the corresponding volume image is seen in the last row. In Figure 5.1a, 5.1b and 5.1c, even without elastic image transformations, the heap of clusters in the middle is detected correctly. A drawback of not using elastic image transformation is an increase of pixels with a class probability between 40% to 60%. This results in the increase in cloudy areas seen in Net 1, Net 2 and Net 3. These areas are from the darker nucleus blobs in the validation image (Figure 5.1h), and impact the segmentation negatively. With the use of elastic image transformations, the impact of dark nucleus areas decreases. Despite the best score, Net 6 is unable to detect the black filled blob on the right side of the ground truth image. The impact of not detecting the dark blob is low, because the challenge metric does not factor in pixel errors. Wrong cell merges or splits have a higher impact on the rand score, than the detection of filled dark blobs.

Currently, 223 participants are registered at the leader board of the ISBI 2012 challenge [3]. Table 5.3 shows the achieved rank, the metric score, and the used augmentation method of the six networks. All networks are trained with the same hyper parameters. Net 1 uses no data augmentation at all. Net 2 uses class weights only. Net 3 uses class weights, rotations and image flipping. Net 4 to 6 use class weights, image rotation, image flipping and elastic image transformations. While applying rotation and elastic image transformation to the ground truth, non binary labels are produced. All labels below 127 are mapped to 0, while all labels equal or above 127 are mapped to 255. We generate the challenge segmentation images based on the model with the best validation loss. Furthermore, no post-processing is applied to our segmentation.
O. Ronneberger, P. Fischer and T. Brox [10] mention elastic image transformations as

[3] `http://brainiac2.mit.edu/isbi_challenge/leaders-board-new` accessed: 26.6.2020

a key concept in their network training. Our evaluation shows that even without any data augmentation as in Net 1, our implementation produces a rand score of 0.947. The height value can be explained by the competition metric. The metric does not factor pixel errors in. The main purpose of the challenge metric is to detect merge and split errors. Therefore, shrinking, expanding or even translating of boundaries between neurites has no impact on the metric score, as long as no new merge or split errors are introduced.
The U-Net architecture with our novel data augmentation called Elastic Gradient Transformation (EGT) scores the highest rand score out of the six trained networks and places at rank 37. The second placed Net 4 uses the RNT method, and is ranked 21 places behind Net 6, at place 58 [4]. The results confirm the importance of our cell tailored EGT method.
To configure our EGT method, we are using the following values: EGT is applied to 90% of the training images at runtime. The σ integer value is randomly chosen between 70 and 90. The scaling values for α and β are randomly chosen integer values between 2000 and 4000. In addition, each scaling value has a 50% chance to be negative.

	Rank	**Rand Score Thin**	**Augmentation**
Net 1	140	0.947	-
Net 2	110	0.962	W
Net 3	134	0.952	W, F
Net 4	58	0.973	W, R, F, RNT
Net 5	95	0.966	W, R, F, RST
Net 6	**37**	**0.976**	W, R, F, EGF

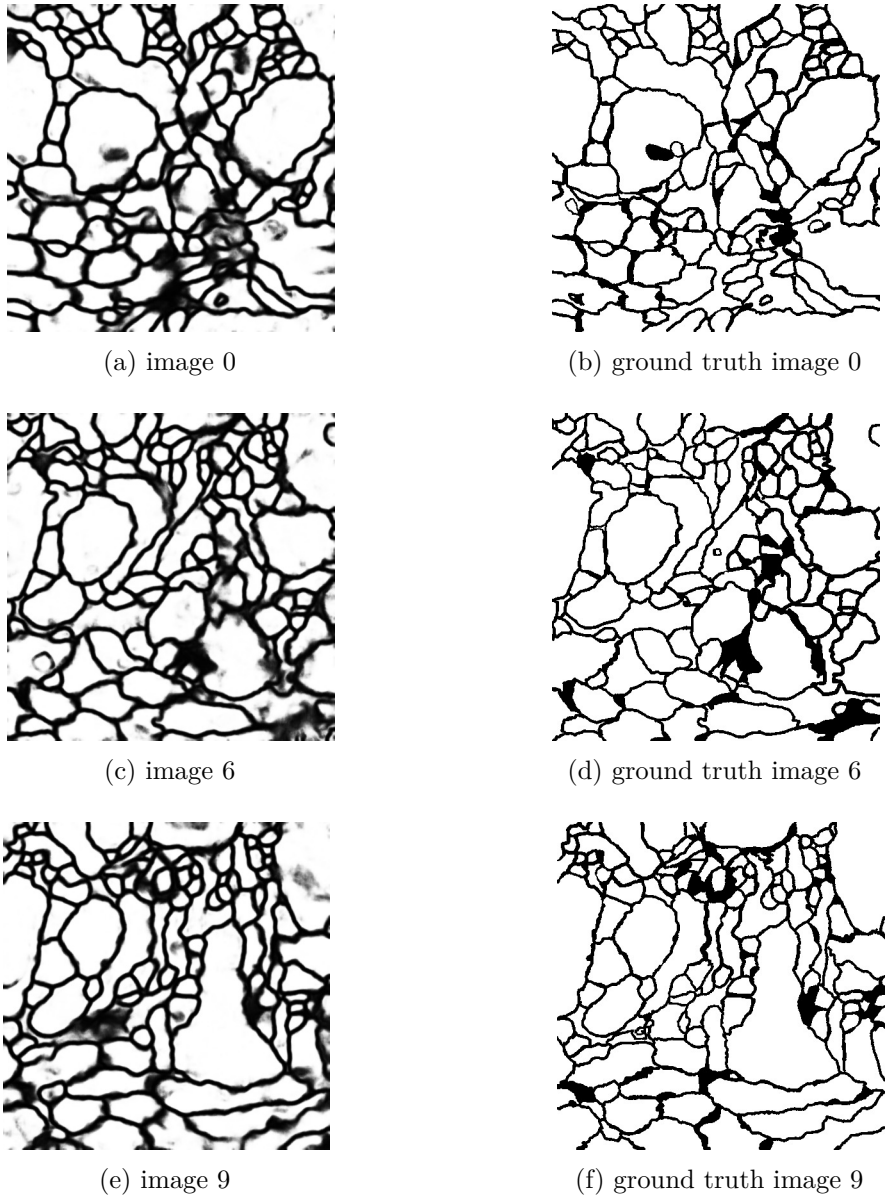
Table 5.3: Rand score for the tested neural networks with data augmentation. Symbols for data augmentation: W for class weights. R for image rotation. F for image flipping. RNT for Random Normalized Transformation. RST for Random Scaled Transformation. EGT for Elastic Gradient Transformation.

SiamU-Net

The proposed SiamU-Net implementation needs two sequential images as input. To calculate the segmentation map of image two, the sequential predecessor which is image one, is needed. Because we use every third image for validation, no predecessor would exists for several train images. We specify, that our SiamU-Net is allowed to use every validation image as a predecessor. In this process, the ground truth of the validation image is never seen by the network, but the temporal information of the validation image is still exploited. However, validation images are not used by the network in the firsst encoder path. No segmentation of a validation image is trained. The use of the validation volume image is restricted to being a predecessor input only.
Image 0 from the validation set has no predecessor. To handle such images, we chose

[4] http://brainiac2.mit.edu/isbi_challenge/leaders-board-new accessed: 26.6.2020

images without a predecessor to be part of the validation set. For such images, our SiamU-Net uses the same image on both encoder paths. This choice should replicate the challenge dataset, for which no predecessor for the first image is available either. Therefore, the impact of image 0 in the validation loss is a desired effect.
The SiamU-Net uses the same parameters as the previous trained U-Net with EGT. Rotations in random 10° steps are applied at every iteration. Therefore, the possibility to train with a non rotated image is $1 : 36$. Horizontal and vertical flipping is applied 50% of the time, while the EGT is applied for 90% of the images.



(a) image 0 (b) ground truth image 0

(c) image 6 (d) ground truth image 6

(e) image 9 (f) ground truth image 9

Figure 5.2: The segmentation map of the validation images is generated with SiamU-Net. These maps contain the class probability.

Our model for U-Net and SiamU-Net needs around 80 minutes to be trained end-to-end. O. Ronneberger, P. Fischer and T. Brox [10] achieve a training time of 10 hours for their implementation. The time difference can partially be explained by the different hardware and software used. Nevertheless, the ability to train our network end-to-end on consumer grade hardware in less than two hours is worth noting.

Figure 5.2 shows the probability maps for three validation images of our highest scoring SiamU-Net model. All three validation images have a rand score over 0.975 points. For image 0, which does not have a predecessor, the circle next to the dark blob on the mid left is not detected correctly. The bottom area of image 0 is detected as black blob, while there should be a white area. As previously mentioned, pixel errors do not have an impact on the rand score, as long as no wrong splits or merges are introduced. Therefore, the wrong black area in the bottom of image 0, does not introduce many splits or merge errors.

In Table 5.4, we show the final ISBI 2012 leader board scores of our implementation of U-Net and SiamU-Net, in comparison with the U-Net by O. Ronneberger, P. Fischer and T. Brox [10] and the kU-Net by J. Chen et al. [14].
With our U-Net implementation, and the use of our EGT, we reach a 0.0036 point higher rand score than the original U-Net [10]. This places our U-Net at rank 36, which results in a difference of 24 ranks in comparison to the original U-Net. Our U-Net implementation with the EGT method, without using temporal image information, achieves a 0.0009 point higher result than the computational intensive kU-Net+RNN architecture by J. Chen et al. [14], which is placed at rank 44. [5]
With the additional temporal image information, our proposed SiamU-Net architecture improves the result even further. The difference of the original U-Net and our SiamU-Net is 0.0045 points and 30 places. To the kU-Net+RNN the difference is 13 places. This proves our hypothesis, that the additional temporal image information from the second encoder path, increases the segmentation quality.

	Rank	**Rand Score Thin**
U-Net [10]	61	0.9727
kU-Net [14]	44	0.9753
our U-Net	37	0.9762
our SiamU-Net	31	0.9772

Table 5.4: leader board ranking of Networks which use a U-Net architecture and their score at the ISBI 2012 challenge [4]. Out of 223 participants, our SiamU-Net places at rank 31. Besides the original U-Net by O. Ronneberger, P. Fischer and T. Brox [10] placed at rank 61, we compare our implementation of U-Net and SiamU-Net to the kU-Net by J. Chen et al. [14] placed at rank 44. The challenge is still open for new contributions without any restrictions.

5.2 DAVIS 2016

To quantify the impact of temporal image information without elastic image transformation, we train our SiamU-Net on the video dataset DAVIS 2016. Both U-Net and SiamU-Net use the same hyper parameters. We reduce the size of each image from

[5] `http://brainiac2.mit.edu/isbi_challenge/leaders-board-new` accessed: 26.6.2020

854×480 to 400×240 pixels due to memory limitations. The goal of the DAVIS 2016 challenge is a binary image segmentation. The main object in focus is considered as foreground, while the rest of the image is considered as background. The dataset is described in detail in Section 4.3. O. Ronneberger, P. Fischer and T. Brox [10] developed U-Net for a biomedical use case in mind. The DAVIS 2016 challenge dataset is used to show the impact temporal image information can have on a U-Net architecture.
To ignore colour information, all images are converted to greyscale before training. As a result, the network is forced to learn object shapes and ignore object colour.
The Davis 2016 dataset with reduced image size can be trained end-to-end in 3 hours. The best results are achieved with a learning rate of 0.01, a momentum of 0.9 and no class weights. The batch size is set to four for both networks, and the last non-full batch is always discarded. Therefore, each epoch of the U-Net contains 519 batches of size four, which use about $5597MB$ of the GPU memory. The SiamU-Net uses two successive frames as input. With a batch size set to four, each epoch contains 259 batches for the SiamU-Net. $7641MB$ GPU memory is needed while training. At each training iteration, images are chosen at random without replacement from all available training images. The DAVIS 2016 challenge allows the first ground truth frame of each validation video to be used while training. As a design choice, we train our model without the first validation ground truth frame. However, the finished trained model needs to be initialized with the first frame of the validation video before inference. Therefore, we train the model multiple times with the first frame. For each video, four separate models are generated, where the first frame is trained, $10, 25, 50$ and 100 times. This helps the network to focus on the given object. Besides the hyper parameter adjustments and the change of the input image size, the network architectures for the ISBI 2012 and the DAVIS 2016 challenge are identical.
Table 5.5 shows the best scores of U-Net and SiamU-Net after training. Because no initialization on the first ground truth frame of each validation video is made, we compare the values on the best training loss. Furthermore, a hard-coded threshold is applied to generate a segmentation map. Output values above or equal 0.5 map to class foreground, while values below 0.5 map to class background.
The proposed SiamU-Net scores highest in five out of six metrics. However, the scores do not differ drastically. The higher recall from our U-Net is balanced out by the higher precision score of the SiamU-Net. This can be seen by the nearly equal F1 score. Furthermore, both networks have a high accuracy and a low false positive rate.

On unseen video data, the difference between both networks increases. For both networks, the best scores are achieved with retraining the model 50 times on each first validation video frame. Table 5.6 shows the mean Jaccard Index of all 20 validation videos for both networks. Our SiamU-Net scores a 7.76 point higher Jaccard Index than the U-Net implementation. Therefore, the importance of temporal image information on the segmentation quality increases significantly on the validation video data. SiamU-Net scores the highest Jaccard Index in thirteen videos, while U-Net scores highest in only seven videos. Looking at the scores of the videos where U-Net performs better,on can

	U-Net train	**SiamU-Net** train
loss	0.0634	0.0393
Jaccard Index	72.34	**72.83**
F1 score	0.8213	**0.8243**
recall	**0.8083**	0.7483
precision	0.8613	**0.9380**
false positive rate	0.0090	**0.0028**
accuracy	0.9772	**0.9809**

Table 5.5: The table shows six different metrics and the resulting loss of the best epoch while training. SiamU-Net has the better score in five out of six metrics.

see that for five out of seven vdeos, the difference to our SiamU-Net score is less than 4.5 points (see Table 5.6: *blackswan*, *cows*, *drift-chicane*, *libby* and *parkour*). On the other hand, when SiamU-Net performs best, only four out of thirteen videos are within a 4.5 range to the U-Net score (see Table 5.6: *car-shadow*, *dog*, *kite-surf* and *motocross-jump*). In the remaining nine videos where SiamU-Net scores highest, the gap between both networks is more than 9 points each time.
In the video *drift-chicane* both Networks perform poorly. The object in motion is a car, which changes the size drastically, from the beginning of the video to the end, due to perspective. Therefore, the initialization of the first ground truth frame has a negligible influence on the later prediction. In addition, the video contains smoke, which blurs the frames. Besides the video *drift-chicane*, on which both networks perform poorly, the lowest Jaccard Index for U-Net is 11.56 at the video *dance-twirl*, while SiamU-Net has a nearly three times higher score of 30.69 for the same video. For SiamU-Net, the lowest score is a Jaccard index of 26.36 points on the video *bmx-trees*, where the difference to the U-Net reaches 11.04 points. Looking at the lowest values for both networks, U-Net reached a Jaccard Index of below 30 in five videos (without the statistical outlier of video *drift-chicane*). The U-Net fails on videos with fast motions like video *breakdance*, *dance-twirl* and *scooter-black*. This results in a Jaccard index of below 18 points. In these videos, the object in motion moves around in the whole scene. The U-Net is not designed to use the available temporal image information. In comparison, the SiamU-Net scores a Jaccard Index of 40.53, 30.69 and 46.60 for *breakdance*, *dance-twirl* and *scooter-black*. The improvement on motion rich videos confirms our hypothesis, that the SiamU-Net learns temporal image information.
The two lowest scoring videos for SiamU-Net, besides *drift-chicane*, are *bmx-trees* with 26.36 points and *libby* with 27.20 points. In these two videos, the object in motion is sometimes occluded by trees and other objects. The U-Net architecture ignores temporal information, as a result, occlusion of the object in motion has no major impact. U-Net is scoring 37.40 points on *bmx-trees* and 31.83 points on *libby*. Occluding foreground objects do not impair the segmentation quality for the U-Net. For the SiamU-Net, an occlusion has a negative impact on the correct segmentation. The SiamU-Net predicts an object to move, based on the temporal information, while in the following frame it gets occluded.

Without an occlusion, the object position would be predicted correctly. As a result, the proposed SiamU-Net has a better performance on videos with high frame motion and no object occlusions, like the videos *breakdance*, *car-roundabout*, *dance-twirl*, *drift-straight*, *goat*, *horsejump-high*, *scotter-black* and *soapbox*. On videos with static motion, where the object in motion is not moving through the whole frame, both networks perform equally. This can be seen in the videos *blackswan*, *car-shadow*, *cows*, *kite-surf* and *parkour*.

Video	**U-Net**	**SiamU-Net**
blackswan	**80.84**	79.58
bmx-trees	**37.40**	26.36
breakdance	15.56	**40.53**
camel	65.68	**75.50**
car-roundabout	42.99	**61.72**
car-shadow	68.45	**70.77**
cows	**85.27**	81.24
dance-twirl	11.56	**30.69**
dog	71.73	**73.26**
drift-chicane	**3.87**	3.15
drift-straight	24.04	**35.83**
goat	50.57	**73.90**
horsejump-high	31.11	**59.84**
kite-surf	47.46	**47.81**
libby	**31.83**	27.20
motocross-jump	50.60	**54.12**
paragliding-launch	**63.05**	53.72
parkour	**34.57**	32.93
scooter-black	17.27	**46.60**
soapbox	29.36	**43.37**
Mean Jaccard Index	43.67	**51.43**

Table 5.6: Jaccard Index of each validation video for U-Net and SiamU-Net. The SiamU-Net reaches a higher Jaccard Index than the U-Net in 13 out of 20 videos, resulting in a 7.76

When looking at the segmentation quality, a good example is the video *camel*. Figure 5.3 shows the segmentation maps of the *camel* video frames 2, 15 and 60. The first row shows the original image. The second row shows the ground truth frames. The third row shows the segmentation maps for the U-Net model. The segmentation maps for SiamU-Net are shown in row four. The final row uses the segmentation maps from the challenge winning entry FCP by F. Perazzi et al. [48]. FCP is working with the full sized images, while our implementation use a reduced image size.

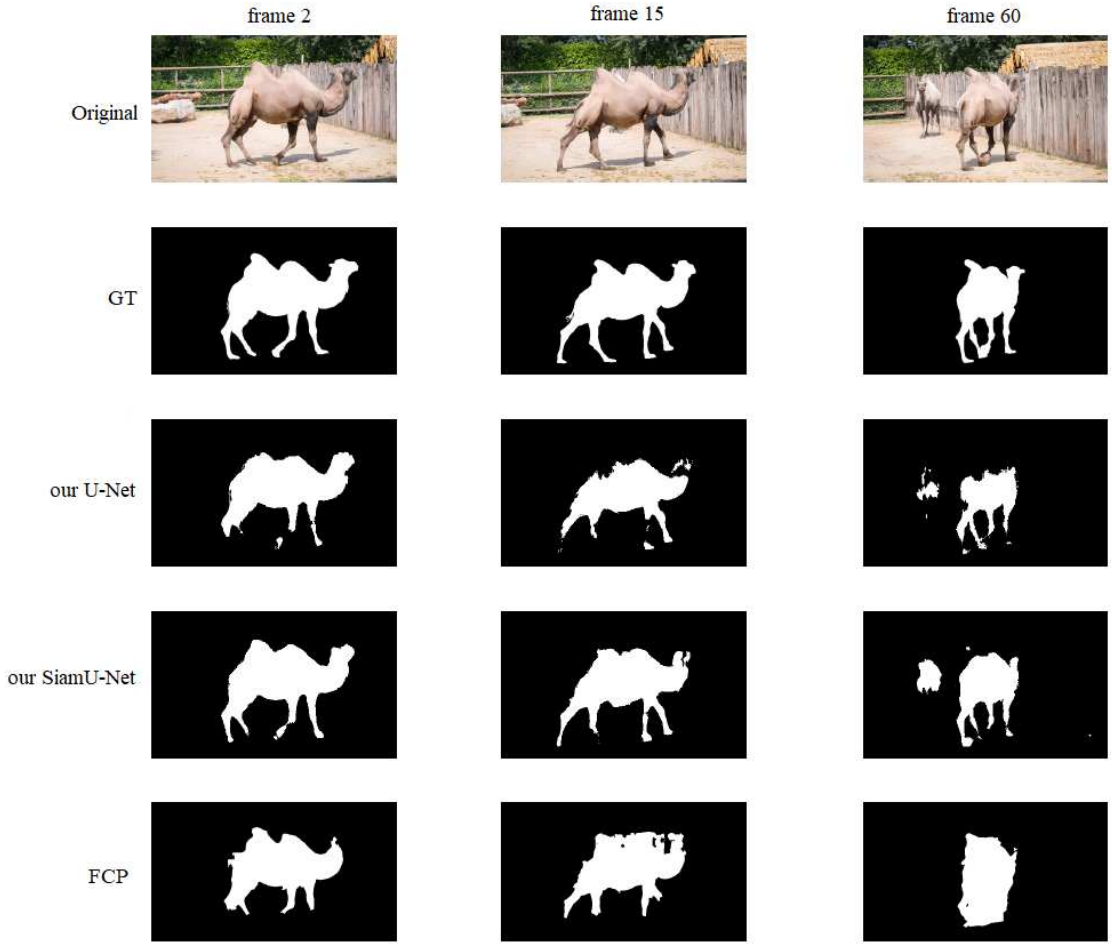


Figure 5.3: Qualitative results of the proposed SiamU-Net and U-Net. First row shows the camel frames 2, 15 and 60. Second row shows the corresponding ground truth. Third row shows the segmentation map of our U-Net implementation with a mean Jaccard Index score of 65.68 points. The forth row shows the segmentation map of our SiamU-Net, which reaches a mean score of 75.50. The last row shows the segmentation of the DAVIS 2016 winning entry FCP with a score of 73.4. U-Net and SiamU-Net use a reduced image size of 400×240 while training. FCP trains with 854×480 sized images.

While FCP [48] is able to correctly detect the single camel in focus, both U-Net and SiamU-Net detect the second visible camel in the left top, as seen in the last segmentation frame. However, FCP [48] detects a big blob in the last frame, where the shape of the camel is lost completely. U-Net and SiamU-Net both keep the basic form of the camel, where the four feet can be detected. The decreasing segmentation quality of the U-Net is noticeable in the last frame, where the segmentation of the camel shrinks. The SiamU-Net manages to use the temporal information to correctly segment the thin feet of the camel in early and late frames, while the FCP network fails to segment the thin feet even in

the first frame. Even though our network train on a reduced image size of 400 × 240, both camel humps can be detected in the shown segmentation maps of the SiamU-Net.

A possible higher score of the proposed SiamU-Net is prevented by the wrong detection of the second camel, which is not marked in the ground truth. While the SiamU-Net can maintain a camel silhouette throughout each frame, the FCP [48] segmentation transforms the shape of the camel to a blob. Despite, having frame 0 of the video as ground truth, FCP starts to chop the shape of the camel starting with frame 2. SiamU-Net is able to use the temporal information to maintain the shape of the camel in early and later frames.

The mean metrics for U-Net and SiamU-Net for the validation videos is shown in Table 5.7. SiamU-Net scores a 7.76 point higher Jaccard Index, than our U-Net implementation.

	U-Net	**SiamU-Net**
Jaccard Index	43.67	**51.43**
F1 score	0.5527	**0.6390**
recall	0.4778	**0.7103**
precision	**0.8199**	0.6544
false positive rate	**0.0087**	0.0361
accuracy	**0.9487**	0.9424

Table 5.7: Mean values of each metric for the whole validation video set.

The difference between the recall values between U-Net and SiamU-Net, corresponds to the proper use of the additional temporal image information. The SiamU-Net has a recall value of 0.7103 points, while the U-Net reaches 0.4778 points. Due to the use of temporal frame information, the SiamU-Net is able to detect less false negatives than the U-Net does. An explanation for the lower precision value of 0.65446 is the impact of occlusion on our segmentation for the SiamU-Net. Due to object occlusion, the temporal movement of the object does not correspond to a new predicted segmentation map. Furthermore, the segmentation of objects which are not in motion (as seen in Figure 5.3), influence the precision too. Therefore, more false positives get detected. The high precision value of 0.8199 from our U-Net, is tarnished by the low recall score of 0.4778. Due to no use of temporal information in the U-Net, more false negatives are detected, while false positives are avoided. The high precision and low recall of the U-Net can be seen in the F1 score, where a score of 0.5527 points is reached. The precision and recall for the SiamU-Net is more balanced, resulting in a F1 score of 0.6390. Both networks have a low false positive rate and a high accuracy. In summary, using the temporal frame information in our SiamU-Net improves the segmentation quality. The problems with occlusion are outweigh by the segmentation improvements in high motion videos.

Last, as seen in Table 5.8, both U-Net and SiamU-Net are behind the top scoring approach of the DAVIS 2016 challenge. As mentioned before, due to the development of the U-

Net architecture for biological image segmentation, a competitive score is not expected. Furthermore, the reduced image size due to memory limitations has an impact on the resulting score. To increase the Jaccard Index on the dataset, a further adaptations to the network layout is needed. However, the DAVIS 2016 dataset can be used to highlight the impact of the temporal image information in a U-Net architecture.

	U-Net	**SiamU-Net**	FCP
Jaccard Index	0.4367	0.5142	0.631

Table 5.8: Comparison of the Jaccard Index of the best semi-supervised approach FCP for the original DAVIS 2016 challenge and our implementations of U-Net and SiamU-Net.

5.3 Chapter Summary

This chapter introduces the evaluation and achieved results of our work. First, we compared two elastic image transformation methods to our novel elastic gradient transformation method. We used our implementation of the U-Net architecture and compared the reached score at the ISBI 2012 challenge. The proposed data augmentation method is tailored to cell images, where realistic looking deformations can be used to increase the dataset size. Results showed the advantages of using our elastic gradient transformations in comparison to random elastic image transformations.
In the second section, we compared our proposed SiamU-Net architecture with our U-Net implementation on the ISBI 2012 challenge. The addition of a mirrored encoder path, which exploits temporal image information, is a significant improvement to the network architecture.
The third section contains a comparison of the U-Net and SiamU-Net on a video segmentation dataset called DAVIS 2016. The evaluation confirmed the benefits of using temporal image information in a U-Net architecture.

CHAPTER 6

Conclusion

To summarize, we first explained the basic terminology needed for CNNs. Afterwards we described the needed layers to create the U-Net architecture. In Chapter 3, several neural networks are discussed, from which the U-Net derived. In addition, adaptation, like the kU-Net are explained. Following, the methodology of the thesis is presented. Besides an explanation of the base U-Net architecture, the proposed adaptation to exploit temporal image information called SiamU-Net is discussed. For the SiamU-Net, a second encoding path is introduced, which is fused back to the U-Net at the latent space. Afterwards, we introduce three elastic image transformation methods, which can be used with the ISBI 2012 dataset. Our novel Elastic Gradient Transformation method is tailored to cell datasets, and is used to generate plausible looking deformations. Furthermore, both the ISBI 2012 and the DAVIS 2016 dataset are explained. After that, the needed evaluation metrics for both challenges are discussed. Moreover, to support our claim of reproducibility, our hardware and software frameworks are explained.
In Chapter 5 we evaluated our proposed novel elastic gradient transformation method on the ISBI 2012 dataset. Results show that, our implementation of the U-Net with the novel EGT method place at rank 37, while the original U-Net by O. Ronneberger, P. Fischer and T. Brox [10] only place at rank 61. Our SiamU-Net with the EGT method increase placement to rank 31 [1]. Our best result is reached without any pre- or post-processing. Finally, we showed the importance of using temporal image information in our SiamU-Net, in comparison to a U-Net architecture. With the use of the temporal image information, SiamU-Net generated a 7.76 point higher Jaccard Index than the U-Net.

6.1 Discussion

Our contributions can be split into three parts. First, we create a novel elastic gradient transformation (EGT) method which is used to expand the small ISBI 2012 dataset. Our

[1] http://brainiac2.mit.edu/isbi_challenge/leaders-board-new accessed: 26.6.2020

results show that the choice of the elastic image transformation has a major impact on the segmentation quality. While our U-Net implementation with the RST method places at rank 95, using the RNT method achieves a better result than the U-Net implementation by O. Ronneberger, P. Fischer and T. Brox [10], placing at rank 58. Our results show that random elastic image transformations are unable to improve the result significantly. However, the EGT method, which uses the image gradient to create plausible looking cell deformations, reaches rank 37 at the challenge. This indicates that a data augmentation which is tailored to the used dataset should always be preferred.

Second, The highest achieved rank at the ISBI 2012 challenge is place 31 by our SiamU-Net. We showed that the addition of the second encoder path is used by the network to learn spatio-temporal information. Due to the small size of the dataset, the improvements are not as high as expected. Another possibility to improve the results would be the use of fewer validation images. Instead of discarding a third of our images for validation, we could inspect the train set, and only add simple cell images to the validation set. This would help the network to learn more difficult cell variations. Another possibility to improve the segmentation, is the introduction of a third class label and a corresponding class weight. Due to image rotation and image transformation, ambiguous cell labels exists, which are currently only mapped to 0 or 1, depending on a threshold.

Third, we use a domain different video dataset for training, which shows the ability of the SiamU-Net to extract temporal image information from the second encoder path. Our SiamU-Net achieves a 7.76 point higher mean Jaccard Index than our U-Net. This indicates the importance of the temporal frame information. The SiamU-Net reaches a recall value of 0.71035 points, while the U-Net only reaches 0.4778 points. Due to the additional temporal image information, the SiamU-Net is able to detect less false negatives. The temporal information helps to predict the position in the following frame. The lower precision score of 0.5644, in comparison to the recall, stems from the impact of occlusion and the wrong detection of objects in main focus. In comparison, our U-Net has a precision value of 0.8199 and a low recall score of 0.4778. A reason for the high precision is the low false positive rate of the U-Net. No temporal information is learned, which can predict the object position wrong. However, not all parts of the object are detected correctly. The more balanced precision and recall for the SiamU-Net can be seen in the F1 score. While SiamU-Net has a F1 score of 0.6390, the U-net reaches a F1 score of 0.5527. The advantages, in high motion videos, outweigh the drawbacks, on videos with occlusion, at the DAVIS 2016 dataset.

Our results show that a competitive result is not reached on the DAVIS 2016 dataset. Foreground objects are not differentiated by their object type. Therefore, no distinction is made between animals, humans or cars. The U-Net architecture achieves state-of-the-art results on datasets with similar looking objects. This leads us to believe that splitting the dataset into three clusters: animals, persons and cars, and training a separate U-Net on each of the datasets will have a positive impact on the Jaccard Index.

6.2 Future Work

A popular approach to improve the segmentation results, is the use of transfer learning. Using a different model as a backbone can have a positive impact on the segmentation results. As a drawback, comprehensibility of the data model gets lost. Therefore, a comparison of different models for using transfer learning would be beneficial. This could improve the results for the ISBI 2012 and the DAVIS 2016 dataset.

An exhaustive hyper parameter tuning would be a next step to boost performance. Besides the learning rate and momentum, a weight decay can be introduced. Furthermore, the layer parameters: filter size, dropout percentage and the use of batch normalization can be discussed. In addition, the convolution filter sizes can be adapted.

When performing elastic image transformation on the ISBI 2012 dataset, a third class can be introduced. All ground truth labels, which are ambiguous due to rotation and elastic transformation, can be assigned to an additional class, instead of applying a hard threshold. A separate weighting value for the new third class can improve the segmentation quality.

The idea to exploit temporal image information in a second encoder path can be applied to other architectures as well. The kU-Net [14] architecture is an example where the additional encoder path can be beneficial. However, the hardware requirements increase drastically for such a deep and wide network.

In addition to the SiamU-Net using a second encoding path, the decoding path can be expanded too. Therefore, the network could learn the segmentation of 2 images at the same time. Currently, the memory size is a limiting factor. A separation of the network at the latent space would allow a split of network parts to multiple GPUs. The feasibility of such a step needs to be evaluated.

Finally, the use of pre- and post-processing methods have a strong impact on the segmentation quality. For both datasets, especially the DAVIS 2016 dataset, calculating a better segmentation threshold for each image, and pruning small connected components, can increase the segmentation quality.

Bibliography

[1] C. Chennubhotla, L. P. Clarke, A. Fedorov, D. Foran, G. Harris, E. Helton, R. Nordstrom, F. Prior, D. Rubin, J. H. Saltz, et al. An assessment of imaging informatics for precision medicine in cancer. *Yearbook of medical informatics*, 26(01):110–119, 2017.

[2] G. Kumar and P. K. Bhatia. A detailed review of feature extraction in image processing systems. In *Advanced Computing & Communication Technologies (ACCT), 2014 Fourth International Conference on*, pages 5–12. IEEE, 2014.

[3] G. Dougherty. Image analysis in medical imaging: recent advances in selected examples. *Biomedical imaging and intervention journal*, 6(3):e32, 2010.

[4] I. Arganda-Carreras, S. C. Turaga, D. R. Berger, D. Cireşan, A. Giusti, L. M. Gambardella, J. Schmidhuber, D. Laptev, S. Dwivedi, J. M. Buhmann, et al. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in neuroanatomy*, 9:142, 2015.

[5] N. Pinto, D. D. Cox, and J. J. DiCarlo. Why is real-world visual object recognition hard? *PLoS computational biology*, 4(1):e27, 2008.

[6] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[8] James Gathany. *image depicts a Disease Control and Prevention intern, working with a transmission electron microscope.* CDC/ Cynthia Goldsmith, 2011. `https://phil.cdc.gov/Details.aspx?pid=13471`.

[9] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in Neural Information Processing Systems 25*, pages 2843–2851. Curran Associates, Inc., 2012.

[10] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[11] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[12] 2018 data science bowl. `https://www.kaggle.com/c/data-science-bowl-2018`. Accessed: 2019-10-1.

[13] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[14] J. Chen, L. Yang, Y. Zhang, M. Alber, and D. Z. Chen. Combining fully convolutional and recurrent neural networks for 3d biomedical image segmentation. In *Advances in neural information processing systems*, pages 3036–3044, 2016.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[19] Markus Spiske. *Mikroskop*. pixabay.com, May 2015. `https://pixabay.com/de/photos/mikroskop-arzt-praxis-untersuchung-772297/`.

[20] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015.

[21] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[22] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[24] K. S. Reddy and K. R. L. Reddy. Enlargement of image based upon interpolation techniques. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(12):4631, 2013.

[25] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[26] D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[27] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[30] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[31] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[32] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[33] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[34] H. Chen, X. J. Qi, J. Z. Cheng, and Ph. A. Heng. Deep contextual networks for neuronal structure segmentation. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[35] T. M. Quan, D. G. C. Hildebrand, and W. Jeong. Fusionnet: A deep fully residual convolutional neural network for image segmentation in connectomics. *arXiv preprint arXiv:1612.05360*, 2016.

[36] S. Xingjian, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[37] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[38] S. Chopra, R. Hadsell, Y. LeCun, et al. Learning a similarity metric discriminatively, with application to face verification. In *CVPR (1)*, pages 539–546, 2005.

[39] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–732, 2016.

[40] P. Y. Simard, D. Steinkraus, J. C. Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, volume 3, pages 958–962, 2003.

[41] A. Cardona, S. Saalfeld, J. Schindelin, I. Arganda-Carreras, S. Preibisch, M. Longair, P. Tomancak, V. Hartenstein, and R. Douglas. Trakem2 software for neural circuit reconstruction. *PloS one*, 7(6):e38011, 2012.

[42] A. Cardona, S. Saalfeld, S. Preibisch, B. Schmid, A. Cheng, J. Pulokas, P. Tomancak, and V. Hartenstein. An integrated micro-and macroarchitectural analysis of the drosophila brain by computer-assisted serial section electron microscopy. *PLoS biology*, 8(10):e1000502, 2010.

[43] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016.

[44] ISBI 2012 challenge evaluation script. `http://brainiac2.mit.edu/isbi_challenge/evaluation`. Accessed: 2020-04-30.

[45] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[46] N. Laanait, J. Romero, J. Yin, M. T. Young, S. Treichler, V. Starchenko, A. Borisevich, A. Sergeev, and M. Matheson. Exascale deep learning for scientific inverse problems. *arXiv preprint arXiv:1909.11150*, 2019.

[47] K. Greff, A. Klein, M. Chovanec, F. Hutter, and J. Schmidhuber. The Sacred Infrastructure for Computational Research. In Katy Huff, David Lippa, Dillon Niederhut, and M Pacer, editors, *Proceedings of the 16th Python in Science Conference*, pages 49 – 56, 2017.

[48] F. Perazzi, O. Wang, M. Gross, and A. Sorkine-Hornung. Fully connected object proposals for video segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 3227–3234, 2015.