Medical Image Analysis 56 (2019) 68–79

Contents lists available at ScienceDirect

Medical Image Analysis

journal homepage: www.elsevier.com/locate/media

GRUU-Net: Integrated convolutional and gated recurrent neural network for cell segmentation

T. Wollmann[a,*], M. Gunkel[b], I. Chung[c], H. Erfle[b], K. Rippe[c], K. Rohr[a]

[a] Biomedical Computer Vision Group, BioQuant, IPMB, Heidelberg University and DKFZ, Im Neuenheimer Feld 267, Heidelberg, Germany
[b] High-Content Analysis of the Cell (HiCell) and Advanced Biological Screening Facility, BioQuant, Heidelberg University, Germany
[c] Division of Chromatin Networks, DKFZ and BioQuant, Heidelberg, Germany

ARTICLE INFO

Article history:
Received 29 August 2018
Revised 9 February 2019
Accepted 17 April 2019
Available online 31 May 2019

Keywords:
Microscopy
Segmentation
Deep learning
Convolutional neural network
Gated Recurrent Unit

ABSTRACT

Cell segmentation in microscopy images is a common and challenging task. In recent years, deep neural networks achieved remarkable improvements in the field of computer vision. The dominant paradigm in segmentation is using convolutional neural networks, less common are recurrent neural networks. In this work, we propose a new deep learning method for cell segmentation, which integrates convolutional neural networks and gated recurrent neural networks over multiple image scales to exploit the strength of both types of networks. To increase the robustness of the training and improve segmentation, we introduce a novel focal loss function. We also present a distributed scheme for optimized training of the integrated neural network. We applied our proposed method to challenging data of glioblastoma cell nuclei and performed a quantitative comparison with state-of-the-art methods. Insights on how our extensions affect training and inference are also provided. Moreover, we benchmarked our method using a wide spectrum of all 22 real microscopy datasets of the Cell Tracking Challenge.

© 2019 Elsevier B.V. All rights reserved.

1. Introduction

Segmentation of prominent structures such as cells in microscopy images is a frequent and important task. In particular, features computed from cell nucleus and cytoplasm segmentations are used to determine phenotypes in quantitative microscopy. Automated quantitative microscopy drives modern biology experiments generating big data, while manual analysis is too labor intensive or error prone. In addition, quantitative microscopy has the potential to reduce the time for diagnostic pathology and improve the quality in clinical routine.

Although many different types of methods for segmentation exist, in recent years, deep learning methods dominate the field of computer vision. Deep learning has been successfully used for cell segmentation in microscopy images (e.g., Ronneberger et al., 2015; Akram et al., 2017; Sadanandan et al., 2017; Yi et al., 2018). Typically, hourglass-shaped Convolutional Neural Networks (CNNs) such as the U-Net or Deconvolution Network (Noh et al., 2015) are used, which aggregate features at multiple image scales. In contrast, Recurrent Neural Networks (RNNs) iteratively refine the segmentation result by exploiting the recurrent structure and mimic Conditional Random Fields (CRFs) or Level Sets (Zheng et al., 2015; Le et al., 2017). Often, RNN approaches are used in a subsequent step to refine segmentation results from an hourglass-shaped CNN (Chen et al., 2018). Segmentation using multi-scale feature aggregation by CNNs and iterative refinement performed by RNNs have distinct strengths and weaknesses. For CNNs it has been shown that they are very effective in capturing hierarchical patterns and extracting abstract features (Lin et al., 2017a). However, a drawback of standard CNNs is that they handle each pixel as a separate classification task and do not explicitly include global priors like shape. In contrast, RNNs iteratively minimize global energies. Multiple weak predictions are combined and the final prediction is iteratively improved using global priors like shape. Therefore, RNNs are robust to local errors and require less parameters than CNNs. However, current RNN-based approaches for segmentation (Zheng et al., 2015; Le et al., 2017) incorporate features only at a single scale. Combining iterative refinement with multi-scale feature aggregation and exploiting their strengths could be beneficial. Recently, a CNN for segmentation of street scenes in video images was proposed, which uses a full-resolution feature path combined with hierarchical feature aggregation (Pohlen et al., 2016). However, iterative refinement of features is limited to summing up the extracted feature maps of each Full-Resolution Residual Unit (FRRU). Other approaches perform full-resolution feature extraction using dilated

* Corresponding author.
E-mail addresses: thomas.wollmann@bioquant.uni-heidelberg.de (T. Wollmann), k.rohr@dkfz.de (K. Rohr).

https://doi.org/10.1016/j.media.2019.04.011
1361-8415/© 2019 Elsevier B.V. All rights reserved.

convolutions (Yu and Koltun, 2015; Wollmann and Rohr, 2017). However, with these approaches undesirable "checkerboard" artifacts occur (Odena et al., 2016). In addition, (Yu and Koltun, 2015; Pohlen et al., 2016; Wollmann and Rohr, 2017) do not use an RNN for iterative refinement. Generally, deep neural networks tend to outperform shallow networks (Poggio et al., 2017), but due to non-linear activation functions and multiplications they suffer from gradient vanishing. In recent years, Deep Neural Network (DNN) architectures like ResNet (He et al., 2016) or DenseNet (Huang et al., 2017) have been proposed to improve the gradient flow. Residual Connections (Drozdzal et al., 2016) and Densely Connected blocks (Jégou et al., 2017) have been transferred from classification tasks to semantic segmentation.

Despite the effectiveness of deep learning methods dealing with large image datasets of natural scenes like ImageNet or MS COCO, it has been shown that training is feasible with relatively small datasets. Common approaches for training on small datasets are transfer learning, adversarial training, and data augmentation. For microscopy images, it has been shown that transfer learning is not very effective, since the properties of the images are quite different from natural images (Liu et al., 2017). Adversarial training improves the performance but does not incorporate domain knowledge, which can help to reduce overfitting (e.g., Arbelle and Raviv, 2018). In contrast, data augmentation (e.g., Ronneberger et al., 2015; Wollmann et al., 2018b; Wollmann et al., 2018a) is a computational efficient and effective method to increase the training data set size, incorporate domain knowledge, and prevent overfitting. However, data augmentation for real datasets poses a number of challenges. Enlarging the dataset has to be performed with care to improve and not harm the training. In particular, the used sampling strategy for the data can bias the network to a certain class or feature. On the other hand, performing transformations like elastic deformation can lead to degenerated objects. In addition, technical challenges arise, if data augmentation is performed with a huge amount of data. Heavy augmentation of datasets can quickly result in millions of images which exceed terabytes of data volume, and even simple operations are then computationally demanding. By naively transferring the generated images to the GPU memory for further processing, the capabilities of the GPU are generally not fully exploited. Therefore, smart techniques for efficient data streaming are required.

In this work, we introduce a novel deep neural network, which combines both paradigms of multi-scale feature aggregation of CNNs and iterative refinement of RNNs. Compared to previous approaches, in our method a convolutional and a recurrent neural network are integrated to aggregate features from different image scales. By employing Densely Connected blocks in the CNN part and a Gated Recurrent Unit (GRU) in the RNN part of our network, we keep the number of learnable parameters and feature tensors to a minimum. Since our network combines a GRU with a U-Net like network, we denote it as GRUU-Net. We propose a novel focal loss function for momentum-based optimizers, which enforces the network to learn separating touching objects. Also, we describe a framework for performing data augmentation for generating huge amounts of data. We describe pitfalls and solutions in data handling, sampling the dataset, and performing transformations of the data. We performed a quantitative comparison with state-of-the art methods using challenging real microscopy image data of DAPI stained cell nuclei in glioblastoma tissue. Insights into our novel loss function, the refinement process, and our data augmentation scheme are provided. In addition, we benchmarked our method using a wide spectrum of all 22 real 2D and 3D datasets of the Cell Tracking Challenge, and yielded superior or competitive results for most of the datasets.

2. Methods

We propose a novel DNN architecture for cell segmentation, which combines iterative refinement of feature maps by a Gated Recurrent Unit (GRU) (Cho et al., 2014) with multi-scale feature aggregation by a U-Net like CNN. Hence, we call this network GRUU-Net. The network is trained with a normalized pixel-wise focal cross-entropy loss to deal with class imbalance and enforce object separation. In addition, we perform heavy data augmentation by a distributed scheme. Below, we describe the architecture of the GRUU-Net and the training procedure.

2.1. Architecture of GRUU-Net

GRUU-Net has a fully convolutional network architecture as sketched in Fig. 1. The neural network unifies a recurrent processing stream with a pooling stream. Both streams are based

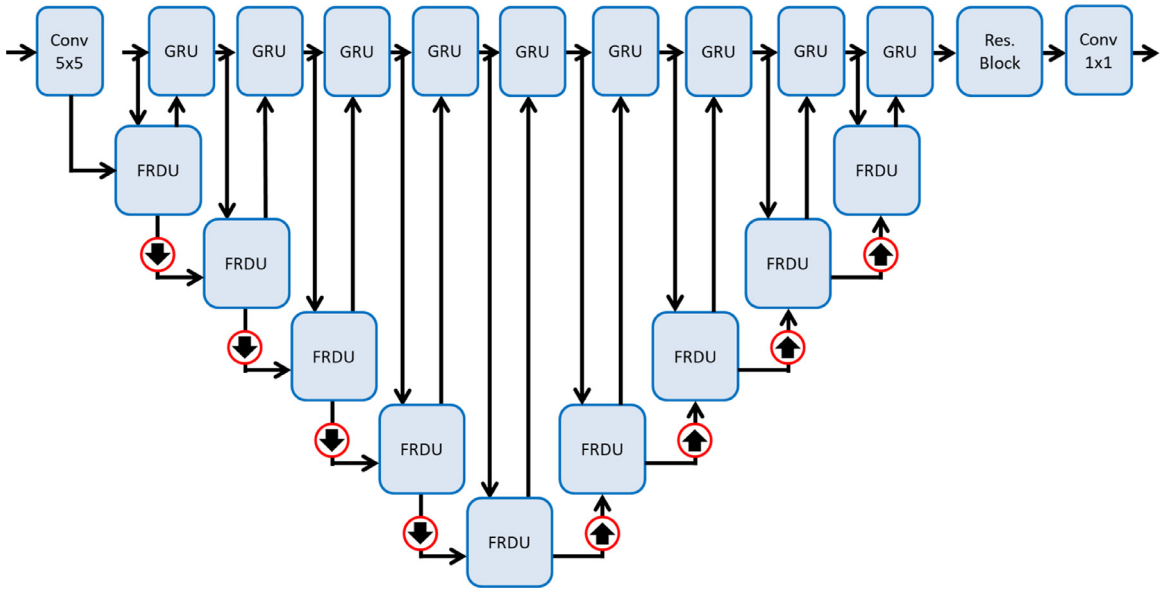

Fig. 1. GRUU-Net architecture. Red circles with an arrow pointing upward/downward denote unpooling/pooling. At each scale Full-Resolution Dense Units (FRDUs) extract features, which are aggregated by a Gated Recurrent Unit (GRU).

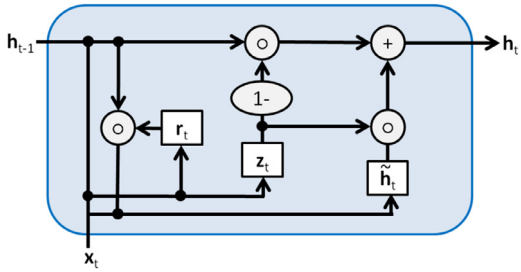

Fig. 2. Gated Recurrent Unit (GRU), where "∘" denotes the Hadamard product.

on a different paradigm. The recurrent stream iteratively refines features at full resolution. On the other hand, the pooling stream extracts high-level features within a large receptive field. The two streams are capable of exchanging information at each resolution level. To maximize the gradient flow we do not use a Feed-Forward Network (Simonyan and Zisserman, 2014), but a Residual Network (He et al., 2016), which is also a recurrent network. In Residual Networks the input $\mathbf{x}_{n-1} \in \mathbb{R}^{m\times n\times p}$ is added to the output $\mathbf{x}_n \in \mathbb{R}^{m\times n\times g}$ of a small subnetwork $\mathcal{F} \in \mathbb{R}^{m\times n\times g}$ with parameters $\mathbf{W}_n \in \mathbb{R}^{k\times k\times p\times g}$ to reduce gradient vanishing, where $m \times n$ is the spatial feature map size, p the number of input filters, g the number of output filters, and $k \times k$ the window size of the convolutional kernel:

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \mathcal{F}(\mathbf{x}_{n-1}; \mathbf{W}_n) \tag{1}$$

Adding the input to the output of the residual is referred to as skip connection. Carefully designed recurrent units are capable of using a residual as shown in Liao and Poggio (2016). Therefore, we kept the principle of residual connections throughout the network to maximize gradient flow.

Recurrent stream: The recurrent stream of our GRUU-Net performs iterative refinement of initially extracted features at full resolution. We use a GRU (Cho et al., 2014) and unfold it over all scales in both bottom up and top down paths of the pooling stream. A GRU (Fig. 2) computes a candidate state $\tilde{\mathbf{h}}_t \in \mathbb{R}^{m\times n\times p}$ from the previous state $\mathbf{h}_{t-1} \in \mathbb{R}^{m\times n\times p}$ and uses the update gate $\mathbf{z}_t \in \mathbb{R}^{m\times n\times p}$ to weight the previous state and the candidate state. Instead of a standard GRU, which operates in a fully-connected manner on a fixed image size, we use a convolutional version of a GRU (Ballas et al., 2015). Therefore, we replace all fully-connected layers within the standard GRU by 3×3 convolutions. First, the reset gate $\mathbf{r}_t \in \mathbb{R}^{m\times n\times p}$ and update gate $\mathbf{z}_t$ are calculated using the input $\mathbf{x}_t \in \mathbb{R}^{m\times n\times p}$ and the parameters $\mathbf{W}_r \in \mathbb{R}^{k\times k\times p\times g}$, $\mathbf{U}_r \in \mathbb{R}^{k\times k\times p\times g}$, $\mathbf{b}_r \in \mathbb{R}$, $\mathbf{W}_z \in \mathbb{R}^{k\times k\times p\times g}$, $\mathbf{U}_z \in \mathbb{R}^{k\times k\times p\times g}$, and $\mathbf{b}_z \in \mathbb{R}$:

$$\mathbf{r}_t = \sigma_g(\mathbf{W}_r * \mathbf{x}_t + \mathbf{U}_r * \mathbf{h}_{t-1} + \mathbf{b}_r) \tag{2}$$

$$\mathbf{z}_t = \sigma_g(\mathbf{W}_z * \mathbf{x}_t + \mathbf{U}_z * \mathbf{h}_{t-1} + \mathbf{b}_z) \tag{3}$$

$$\sigma_g(x) = \frac{e^x}{e^x + 1} \tag{4}$$

where the operator "$*$" denotes convolution. Then, the candidate state $\tilde{\mathbf{h}}_t$ is calculated using the parameters $\mathbf{W}_h \in \mathbb{R}^{k\times k\times p\times g}$, $\mathbf{U}_h \in \mathbb{R}^{k\times k\times p\times g}$, $\mathbf{b}_h \in \mathbb{R}$:

$$\tilde{\mathbf{h}}_t = \sigma_h(\mathbf{W}_h * \mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \circ \mathbf{h}_{t-1}) + \mathbf{b}_h) \tag{5}$$

$$\sigma_h(x) = \mathrm{LReLU}(x) = \begin{cases} x & x > 0 \\ 0.2x & otherwise \end{cases} \tag{6}$$

where the operator "$\circ$" denotes the Hadamard product. For the activation function σ_h, we used Leaky Rectifed Linear Units (LReLU) (Maas et al., 2013). Finally, the previous state $\mathbf{h}_{t-1}$ and the candidate state $\tilde{\mathbf{h}}_t$ are weighted to determine the new state $\mathbf{h}_t \in \mathbb{R}^{m\times n\times p}$.

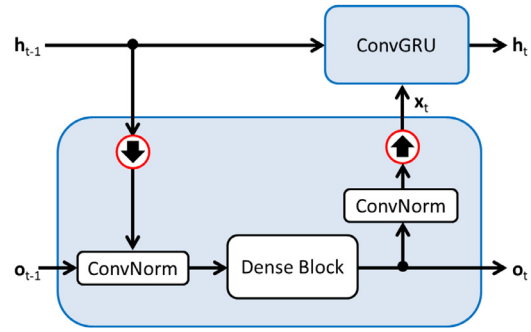

Fig. 3. Full-Resolution Dense Unit (FRDU).

$$\mathbf{h}_t = \mathbf{z}_t \circ \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \circ \tilde{\mathbf{h}}_t \tag{7}$$

Pooling stream: The pooling stream consists of pooling blocks, Full-Resolution Dense Units (FRDUs), and unpooling blocks. To increase the size of the receptive field and the number of feature maps of the network we include a bottom up path with max pooling blocks. To restore the original resolution and perform top down inference we construct a top down path. Within this path, we perform bilinear interpolation instead of transposed convolution as in the U-Net. In Lin et al. (2017b) it has been shown that both bottom up and top down paths for feature extraction are important for capturing the semantic information of an image. Both bottom up and top down paths alternatingly consist of pooling/unpooling and FRDU blocks.

FRDU blocks (Fig. 3) combine information from the recurrent stream with the pooling stream and feed back the results to both streams. Therefore, the FRDU is capable of integrating convolutional and gated recurrent neural networks. Thus, high resolution information can be stored in the recurrent stream and simultaneously high-level features can be extracted in the pooling stream at multiple resolutions.

To combine the feature maps from both streams $\mathbf{h}_{t-1}$ and $\mathbf{o}_{t-1} \in \mathbb{R}^{m\times n\times p}$, we use max pooling (arrow down) to map $\mathbf{h}_{t-1}$ to the resolution of $\mathbf{o}_{t-1}$ and concatenate both feature maps. Afterwards, we perform a batch normalized (BN) 1×1 convolution to create an embedding. We found that using bilinear interpolation instead of max pooling decreased the stability of the training. Features $\mathbf{o}_t \in \mathbb{R}^{m\times n\times p}$ at the current resolution are extracted by a Densely Connected block (Dense Block) (Huang et al., 2017) with k layers. In Densely Connected blocks each layer has access to all feature maps of the previous layers. Therefore, layer n receives the concatenated feature maps $[\mathbf{x}_0; \ldots; \mathbf{x}_{n-1}]$ as input:

$$\mathbf{x}_n = \mathcal{F}(\mathbf{x}_0; \ldots; \mathbf{x}_{n-1}; \mathbf{W}_n) \tag{8}$$

By including additional skip connections, the number of parameters can be significantly reduced, while increasing the depth of the network without harming gradient flow or performance. The input $\mathbf{x}_t$ of the GRU is extracted from $\mathbf{o}_t$ by performing a 1×1 convolution and nearest neighbor interpolation (arrow up) to the resolution of $\mathbf{h}$. Using bilinear interpolation yielded inferior results.

Details on the layer configuration of the GRUU-Net are provided in Table 1. In addition to the pooling and recurrent stream, we extract the initial feature maps by performing a 5×5 convolution in the first layer. It has been shown that early layers benefit from negative activations of the filters (Paszke et al., 2016; Wollmann et al., 2018a, b). To minimize the number of parameters, but keep the negative activations, we use Leaky Rectified Linear Units (LReLU) (Maas et al., 2013) (see (6)) for all non-linear layers with a leakage factor of 0.2. All filters are initialized using a scaled random normal distribution (He et al., 2015). We increase

Table 1
GRUU-Net layer configuration. The superscripts denote the filter size for the convolutions and the number of layers k in the Dense blocks of the FRDU. The subscripts represent the number of output feature maps.

	$\text{conv}_{32}^{5\times5}$+BN+LReLU		
Pooling Stream	$\text{FRDU}_{32}^{3\times3,k=3}$	$\text{GRU}_{32}^{3\times3}$	Recurrent Stream
	max pooling		
	$\text{FRDU}_{64}^{3\times3,k=3}$	$\text{GRU}_{32}^{3\times3}$	
	max pooling		
	$\text{FRDU}_{128}^{3\times3,k=6}$	$\text{GRU}_{32}^{3\times3}$	
	max pooling		
	$\text{FRDU}_{256}^{3\times3,k=12}$	$\text{GRU}_{32}^{3\times3}$	
	max pooling		
	$\text{FRDU}_{512}^{3\times3,k=12}$	$\text{GRU}_{32}^{3\times3}$	
	bilin. upsampling		
	$\text{FRDU}_{256}^{3\times3,k=12}$	$\text{GRU}_{32}^{3\times3}$	
	bilin. upsampling		
	$\text{FRDU}_{128}^{3\times3,k=6}$	$\text{GRU}_{32}^{3\times3}$	
	bilin. upsampling		
	$\text{FRDU}_{64}^{3\times3,k=3}$	$\text{GRU}_{32}^{3\times3}$	
	bilin. upsampling		
	$\text{FRDU}_{32}^{3\times3,k=3}$	$\text{GRU}_{32}^{3\times3}$	
	-	Residual $\text{Block}_{32}^{3\times3}$	
	-	$\text{conv}_{2}^{1\times1}$+BN	
	softmax		

the stability of the training by using reflection-padding instead of zero-padding. For computing the final prediction, we use a Residual Block and a 1×1 convolution for the output $\mathbf{x}\in\mathbb{R}^{m\times n\times g}$ of the recurrent stream followed by the softmax function to compute the pixel-wise foreground and background probability. Our network could be extended by using an additional class for object borders. To better focus on the improvements of our base network, we did not do this and also did not perform refinement with an additional CRF (e.g., Zheng et al., 2015).

2.2. Focal loss function

We train the network using an extension of the focal loss in Lin et al. (2017c), which was previously used for object detection in images of natural scenes using a stochastic gradient descent optimizer. The focal loss is an extension of the cross-entropy loss, which addresses very large class imbalance and performs implicit negative mining by enforcing a higher loss on uncertain predictions. In our application, especially background pixels separating cells are rare compared to inner and outer pixels of cells, and can be hardly learned via a traditional cross-entropy loss. Using the focal loss relieves designing weighting functions as done in Ronneberger et al. (2015) and naturally generalizes to many difficult applications. We extended the focal loss in Lin et al. (2017c) by introducing a normalization and adapting it to semantic segmentation using a momentum-based optimizer. The focal loss in (Lin et al., 2017c) is defined by:

$$\text{FL}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{bmng} \text{vec}(-\mathbf{w}_{\text{FL}}(\mathbf{x},\mathbf{y})\circ\mathbf{y}\circ\log(\mathbf{x}))_i \tag{9}$$

which is calculated pixel-wise over the vectorized (vec operator) predictions $\mathbf{x}\in\mathbb{R}^{b\times m\times n\times g}$ and ground truth $\mathbf{y}\in\mathbb{R}^{b\times m\times n\times g}$, and summed up over all pixels $m\times n$, the two classes $g=2$ (background, foreground), and the b samples within a batch, weighted using the weights:

$$\mathbf{w}_{\text{FL}}(\mathbf{x},\mathbf{y}) = \mathbf{y}\circ(\mathbf{1}-\mathbf{x})^{\gamma} \tag{10}$$

where $\mathbf{1}$ denotes a tensor of ones and γ the focusing parameter, and the cross-entropy:

$$\text{CE}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{bmng} \text{vec}(-\mathbf{y}\circ\log(\mathbf{x}))_i \tag{11}$$

Since scaling by $\mathbf{w}_{\text{FL}}$ is equivalent to changing the learning rate, the focal loss leads to an unequal learning rate over training batches. This can be seen when inserting the focal loss FL into the equation of a standard gradient step to compute a network weight $W^t\in\mathbb{R}$ in one layer, using the learning rate l, the network prediction $\mathbf{x}^{t-1}(W^{t-1})\in\mathbb{R}^{b\times m\times n\times g}$ at iteration $t-1$, and the corresponding ground truth $\mathbf{y}^{t-1}\in\mathbb{R}^{b\times m\times n\times g}$:

$$W^t \leftarrow W^{t-1} - l\,\nabla_{W^{t-1}}\left[\text{FL}(\mathbf{x}^{t-1},\mathbf{y}^{t-1})\right] \tag{12}$$

$$\begin{aligned}\text{FL}(\mathbf{x}^{t-1},\mathbf{y}^{t-1}) &= \sum_{i=1}^{bmng} \text{vec}(-\mathbf{w}_{\text{FL}}(\mathbf{x}^{t-1},\mathbf{y}^{t-1})\circ\mathbf{y}^{t-1}\circ\log(\mathbf{x}^{t-1}))_i \\ &= \sum_{i=1}^{bmng}\Big(-\text{Diag}(\text{vec}(\mathbf{w}_{\text{FL}}(\mathbf{x}^{t-1},\mathbf{y}^{t-1}))) \\ &\qquad \text{vec}(\mathbf{y}^{t-1}\circ\log(\mathbf{x}^{t-1}))\Big)_i\end{aligned} \tag{13}$$

where the diagonal matrix $\text{Diag}(\text{vec}(\mathbf{w}_{\text{FL}}(\mathbf{x}^{t-1},\mathbf{y}^{t-1})))$ performs an anisotropic scaling of the cross-entropy. Momentum-based optimizers like ADAM (Kingma and Ba, 2015) or AMSGrad (Reddi et al., 2018) use the loss to adjust the momentum and therefore the learning rate, which interferes with the scaling by the focal loss. Combining focal loss and momentum-based optimizers can therefore result in unstable training. To improve the stability during training, we suggest normalizing the weights $\mathbf{w}_{\text{FL}}$ to one within a batch using the sum of all weights. Normalization of the focal loss for each image independently was less stable. The same effect can be observed for the Dice loss (Milletari et al., 2016). Incorporating an additional class weight did not improve the results in our experiments. Thus, our proposed *normalized focal loss* $\mathcal{L}(\mathbf{x},\mathbf{y})$ is defined by:

$$\mathcal{L}(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{bmng}\text{vec}(-\mathbf{w}_{\text{FL}}(\mathbf{x},\mathbf{y})\circ\mathbf{y}\circ\log(\mathbf{x}))_i}{\sum_{i=1}^{bmng}\text{vec}(\mathbf{w}_{\text{FL}}(\mathbf{x},\mathbf{y}))_i} \tag{14}$$

In all experiments, we used $\gamma=2$ as in Lin et al. (2017c). By normalizing $\mathbf{w}_{\text{FL}}$ to one, the trace of $\text{Diag}(\text{vec}(\mathbf{w}_{\text{FL}}(\mathbf{x}^{t-1},\mathbf{y}^{t-1})))$ and thus the overall scaling remains the same in all iterations. We found that our normalized focal loss improved the stability significantly.

2.3. Training

We augmented the datasets to increase the variability of the training data without changing the semantic information. Since some data augmentation steps are computationally expensive, we developed a scheme for distributed data augmentation (Fig. 4).

Data augmentation is usually done on a single machine (e.g., Google Brain Team, 2019; Paszke et al., 2019; Microsoft Research, 2019). When performing computational demanding augmentation steps, the GPU can not be fully utilized. Instead, in our work we perform distributed data augmentation using multiple compute nodes, which has additional technical challenges (e.g., computation

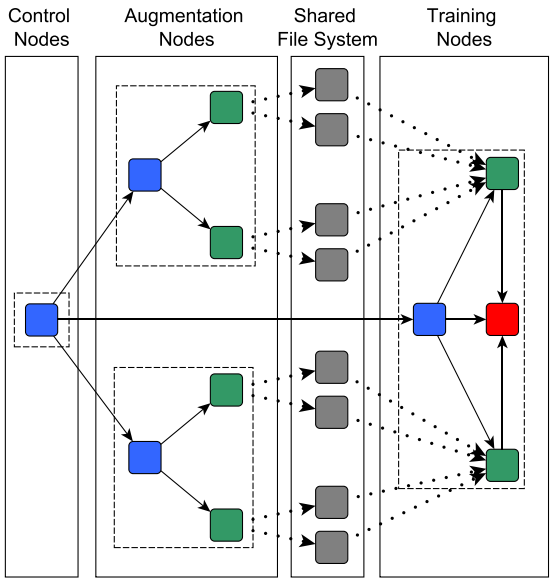

Fig. 4. Scheme for distributed data augmentation and training. Blue boxes are CPU management processes and green boxes CPU compute threads. Gray boxes represent preprocessed files and dotted lines indicate file access. Red boxes represent GPU computations. Dashed rectangles denote compute nodes connected by threads creation (solid lines) outlining the hierarchy tree of thread forks. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

resource management, job coordination, data transfer). A single control node coordinates the data augmentation nodes, which generate augmented training data, and the training nodes, which perform the actual training. Each data augmentation node starts several threads that generate training data chunks in a fast readable binary format (TFRecord). Files are transfered to the training nodes via a shared file system and read by multiple CPU reader threads. These readers constantly transfer the data to the GPU memory to prevent the GPU from being idle. We used separate augmentation nodes for generating training data and validation data. The nodes of the distributed system are connected by high throughput InfiniBand, data is stored on up-to-date SSDs, and the CPUs are fourth generation Intel Xeon CPUs. When using online multi-threaded data augmentation, we observed a mean GPU utilization of about 60%. With our distributed data augmentation scheme, we were able to increase the mean GPU utilization to 98%. We note that the performance of multi-threaded and multi-machine data augmentation strongly depends on the local hardware infrastructure. In our case, the used distributed system has a negligible IO overhead, which is beneficial for distributed data augmentation.

For training and validation, we sample N_e epochs from the original images and respective ground truth data randomly. $N_e - 1$ epochs are augmented using our distributed computing scheme. The last epoch is not augmented so that the network is fine tuned to the dataset. Instead of using whole images, we extract small crops with approximately the size of the largest object in the dataset. For the regions of interest (ROIs), we used the bounding box of the ground truth segmentations. During training, we sample image crops from the ROIs to achieve a balance between foreground and background samples. Each crop is augmented by rotation, flipping, brightness, zoom, and elastic deformation. Augmenting by zoom and elastic deformation pose special challenges in the case of microscopy images, since altering the object structure in the ground truth can wrongly change the semantics of the training data (e.g., cell splitting). We use a grid-based method to perform elastic deformation. In this method, displacement vectors of the grid anchor points are sampled from a normal distribution. The deformed image is then generated using bicubic interpolation. To prevent merging of objects with the same label, we assign an identity to each object in the ground truth and perform data augmentation. Afterwards, we use morphologic operations to ensure that previously separated objects are still separated by at least one pixel. We generate a one-hot encoding (vector of zeros except one value) for each pixel of the crop. Augmented crops exceeding the original image dimensions are filled up with reflection padding.

We train the network using the AMSGrad optimizer in Reddi et al. (2018). We used a batch size of two and an initial learning rate $l_{init} = 0.001$ as well as $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Each dataset is split into 50% for training, 25% for validation, and 25% for testing, and the network is trained using early stopping and cross-validation. Our model was implemented in Tensorflow (Abadi et al., 2016), and we used an Intel i7-6700K workstation with a NVIDIA GeForce GTX 1070 Ti GPU.

3. Experimental results

We applied our method to different types of datasets and performed a quantitative comparison with state-of-the-art methods. For quantifying the performance, we used the following measures, calculated as one score integrated over all test images:

Dice: The Sørensen-Dice coefficient measures the similarity of two sets **X** and **Y**, where |**X**| and |**Y**| are the cardinalities of the sets.

$$\mathrm{Dice}(\mathbf{X}, \mathbf{Y}) = \frac{2|\mathbf{X} \cap \mathbf{Y}|}{|\mathbf{X}| + |\mathbf{Y}|} \tag{15}$$

SEG: The object-wise Jaccard similarity index measures the Jaccard similarity index of two matching objects (Maška et al., 2014). An object in the two sets **X** and **Y** is matched if the overlap is more than 50%. Objects consisting of just one pixel are discarded.

$$\mathrm{Jaccard}(\mathbf{X}, \mathbf{Y}) = \frac{|\mathbf{X} \cap \mathbf{Y}|}{|\mathbf{X} \cup \mathbf{Y}|} \tag{16}$$

Hausdorff: The Hausdorff distance measures the maximum occurring Euclidean distance d between two sets **X** and **Y**.

$$\mathrm{Hausdorff}(\mathbf{X}, \mathbf{Y}) = \max\left(\sup_{x \in \mathbf{X}} \inf_{y \in \mathbf{Y}} d(x, y), \sup_{y \in \mathbf{Y}} \inf_{x \in \mathbf{X}} d(x, y)\right) \tag{17}$$

3.1. Ablation study for data augmentation method

We performed an ablation study to investigate the effectiveness of our data augmentation scheme. Therefore, we disabled different augmentation steps and evaluated the performance of our method. We used a challenging dataset consisting of 50 maximum intensity projection tissue images of glioblastoma cells (Baltissen et al., 2018). The images have a size of 2048×2048 pixel and a resolution of $0.12\,\mu\,\mathrm{m} \times 0.12\,\mu\,\mathrm{m}$, and were acquired using confocal spinning disc microscopy and show cell nuclei with fluorescently stained telomeres, centromers, PML proteins, and DNA. The dataset is challenging due to high image noise, strongly heterogeneous intensity variation, cell clustering and overlaps, high shape variation, and poor contour information. Two experts manually determined the ground truth by drawing contours using ImageJ for more than 250 cell nuclei. The dataset was split into 25 training, 5 validation, and 20 test images. We used our distributed computing scheme with different disabled augmentation steps to generate training datasets. These datasets were used for training of our GRUU-Net. To demonstrate the generalization ability of our data augmentation scheme for CNNs, we also used a standard U-Net (Ronneberger et al., 2015). Both networks were trained with early stopping by checking (every 100 iterations) whether a plateau

Table 2
Ablation study of our data augmentation method for the glioblastoma dataset using the U-Net and our GRUU-Net.

Experiment		1	2	3	4	5	6
Cropping			✓	✓	✓	✓	✓
Flipping/Rotation				✓	✓	✓	✓
Zoom					✓	✓	✓
Brightness						✓	✓
Deformation							✓
U-Net	Training Iteration	2000	2500	3500	5000	10,000	10,000
	SEG	0.629	0.695	0.804	0.784	0.798	**0.807**
	Dice	0.892	0.889	0.907	0.912	0.926	**0.932**
	Hausdorff	36.844	34.190	22.106	27.019	22.277	**15.489**
GRUU-Net	Training Iteration	7500	9000	10,000	12,000	10,000	10,000
	SEG	0.647	0.695	0.723	0.751	0.811	**0.840**
	Dice	0.909	0.917	0.922	0.914	0.923	**0.933**
	Hausdorff	56.091	71.864	60.918	52.457	20.436	**14.179**

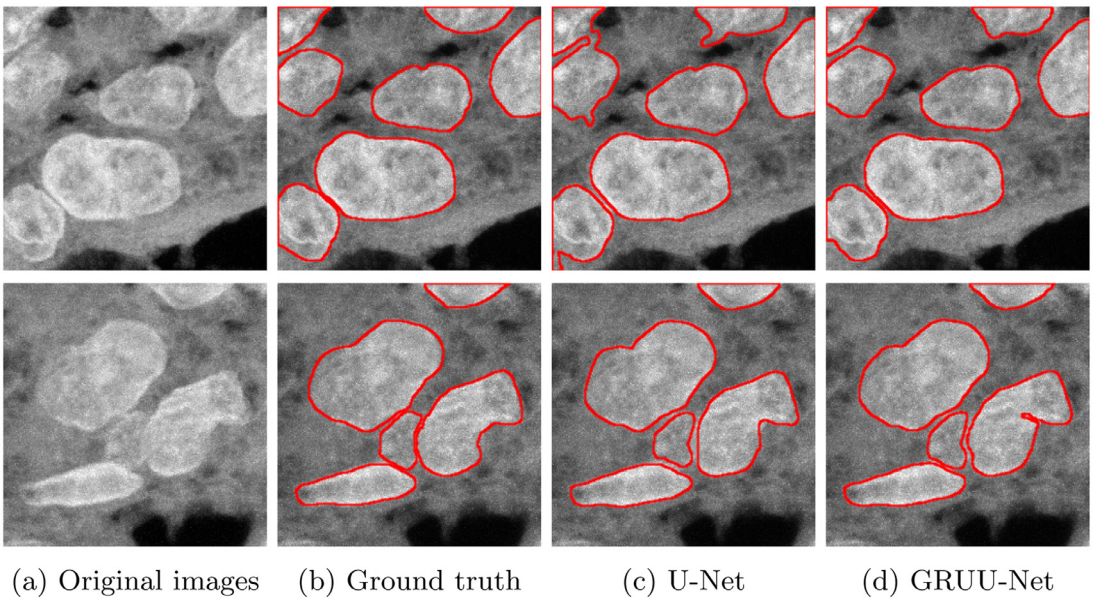

Fig. 5. Segmentation results of GRUU-Net, U-Net, and corresponding ground truth annotations for two example images of tissue microscopy images of glioblastoma cells (top, bottom).

is reached, and evaluated on the test images. Table 2 shows the experimental results. It can be observed that each augmentation step generally increases the performance. However, some augmentation steps such as zoom decrease the performance of some measures due to significantly increased variability of the dataset and therefore more difficult training. The GRUU-Net yields better results that the U-Net, but not for all ablated augmentation steps. The best result is obtained using all data augmentation steps (last column). Note that the maximum training iteration number increases with the number of augmentation steps. As expected, the number of iterations, before a plateau of the loss is reached (when using early stopping), increases with more data augmentation due to increased variability in the training dataset. With increasing variability in the training dataset the generalization abilities increase and the network gets less prone to overfitting. Samples images and segmentation results of our GRUU-Net compared to the U-Net using all augmentation steps are shown in Fig. 5. It can be seen that the GRUU-Net yields superior results and better separates the cell nuclei.

3.2. Evaluation of normalized focal loss

We investigated our GRUU-Net using the original focal loss (Lin et al., 2017c) and our normalized focal loss in 14 for the same glioblastoma dataset employed in Section 2 above. To demonstrate the generalization ability of our normalized focal loss, we also applied a U-Net with the original focal loss and our normalized focal loss. In addition, we performed a comparison with other methods including supervised and unsupervised machine learning methods. Below, we outline these methods.

Local thresholding (Bernsen, 1986): Gaussian filtering was performed with $\sigma = 4$ followed by Bernsen's thresholding method using a contrast threshold of 15.

Fast marching (Sethian, 1996): The fast marching algorithm is based on level sets and uses a deformable model. An image was first smoothed by a Gaussian filter ($\sigma = 4$) and intensity maxima were used as seed points for the deformable model.

K-means clustering (Arthur and Vassilvitskii, 2007): A Gaussian filter ($\sigma = 4$) was applied for smoothing and then the intensity values were clustered into two clusters. The manually selected foreground cluster was used as segmentation result.

Ilastik (Sommer et al., 2011): Ilastik uses a random forest classifier for pixel-wise segmentation. All provided features were used and the image scales were defined by σ={0.3, 0.7, 1.0, 1.6, 3.5, 5.0, 10.0}. The classifier was trained using 20 fully annotated images from the training set.

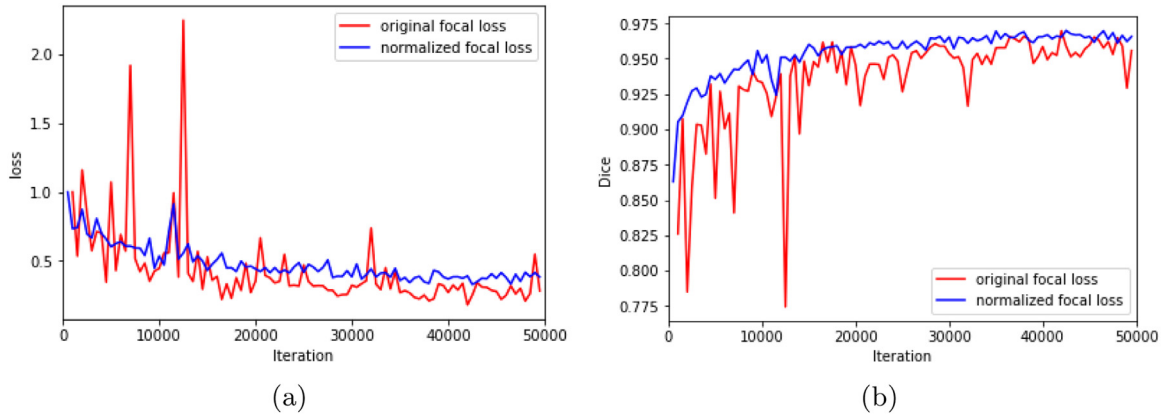

Fig. 6. (a) Original and normalized focal loss for the validation set during training. The values were normalized with respect to the maximum value. (b) Dice coefficient for the validation set during training for original and normalized focal loss.

Table 3
Comparison of methods for the glioblastoma dataset.

Method	SEG	Dice	Hausdorff
Local thresholding	0.480	0.881	42.558
Fast marching	0.491	0.905	36.678
K-means clustering	0.531	0.910	35.518
Ilastik	0.610	0.911	25.016
U-Net (Weighted CE loss)	0.770	0.925	18.024
U-Net (Non-Normalized FL)	0.553	0.865	61.278
U-Net (Normalized FL)	0.807	0.932	15.489
ASPP-Net (Weighted CE loss)	0.798	0.877	65.228
ASPP-Net (Non-Normalized FL)	0.708	0.844	69.299
ASPP-Net (Normalized FL)	0.833	0.911	23.351
GRUU-Net (Weighted CE loss)	0.772	0.930	18.020
GRUU-Net (Non-Normalized FL)	0.777	**0.933**	16.024
GRUU-Net	**0.840**	**0.933**	**14.179**

U-Net (Ronneberger et al., 2015): U-Net is a popular hourglass-shaped convolutional neural network for semantic segmentation. A multi-scale classifier is learned while preserving high resolution features through skip connections. Learning of difficult samples is enforced using a hand-crafted cross-entropy weight map computed by morphological operations. Training was performed using the same training data split and data augmentation as for our GRUU-Net.

ASPP-Net (Wollmann et al., 2018b): ASPP-Net is an hourglass-shaped convolutional neural network for semantic segmentation. Compared to the U-Net it incorporates an additional Atrous Spatial Pyramid Pooling (ASPP) block to achieve a larger receptive field than the U-Net.

From the results in Table 3 it can be seen that our GRUU-Net yields the best performance. It also turns out that using our normalized focal loss improves the performance for SEG of our GRUU-Net (0.840), ASPP-Net (0.833), and that of the U-Net (0.807), compared to using the Weighted CE loss by Ronneberger et al. (U-Net: 0.553, ASPP-Net: 0.798, GRUU-Net: 0.772) or the original Focal loss (U-Net: 0.770, ASPP-Net: 0.708, GRUU-Net: 0.777). Fig. 6 shows the convergence curves of the original and normalized focal loss during training. It can be seen that our normalized focal loss leads to more stable training than the original focal loss.

3.3. Visualization of iterative refinement of the GRUU-Net

To provide insight into the refinement process of our GRUU-Net, we investigated segmentation results at different iterations. As example image, we used a fluorescence microscopy image of rat mesenchymal stem cells (Fluo-C2DL-MSC) from the Cell Tracking Challenge (Maška et al., 2014; Ulman et al., 2017). The results at different iterations were obtained by applying the final residual block, convolution, and softmax function to the corresponding hidden state of the GRU (cf. Fig. 1).

The refined results as a function of the number of iterations are shown in Fig. 7. It can be observed that the segmentation is improved in each iteration. It can also be seen that in the contracting path of the GRUU-Net (iterations 1–4) the segmented region is continuously enlarged. In the expanding path (iterations 5–9) the segmented object is smoothed.

3.4. Method comparison for Cell Tracking Challenge data

We also evaluated the performance of our GRRU-Net using the Cell Tracking Challenge training data (Maška et al., 2014; Ulman et al., 2017). The challenges compared several cell segmentation and tracking methods (e.g., Harder et al., 2009; Esteves et al., 2012; Magnusson and Jaldén, 2012; Ronneberger et al., 2015). We applied our method to all available real 2D and 3D datasets, comprising 11 different categories of data, which represent a very wide spectrum of cell microscopy data (see Fig. 8). The datasets comprise different microscope modalities (fluorescence, differential interference contrast, phase-contrast) and cells (rat mesenchymal stem cells, mouse stem cells, lung cancer cells, human breast carcinoma cells, HeLa cells, U373 cells, pancreatic stem cells, C. elegans embryo, CHO nuclei). In Ulman et al. (2017) only one method, namely UP-PT was applied to all these 22 real data of the challenge. Each category of datasets consists of two videos. We trained our GRUU-Net using the fully labeled frames of one video and tested it on the other video. Thus, we used quite limited data for training. In Ulman et al. (2017), the measure SEG was employed to quantify the segmentation performance. To complement the results in Ulman et al. (2017), we also computed the mean Dice coefficient and the mean Hausdorff distance, if fully annotated images were available. Tables 4 and 5 show the results of our method for the 2D and 3D datasets, respectively. For the 2D datasets, we also provide results for different variants of our network (Weighted Cross-Entropy loss, Non-Normalized Focal loss, and our Normalized Focal loss). We also compared the results with the local adaptive thresholding approach HD-Har (Harder et al., 2009) and the U-Net (Ronneberger et al., 2015). Note that in Ulman et al. (2017), for the U-Net both videos of a dataset category were used for training and testing. In our work, for a fair comparison we performed training on one video and testing on the other video. In addition, we included results of other previous methods, which are briefly outlined below.

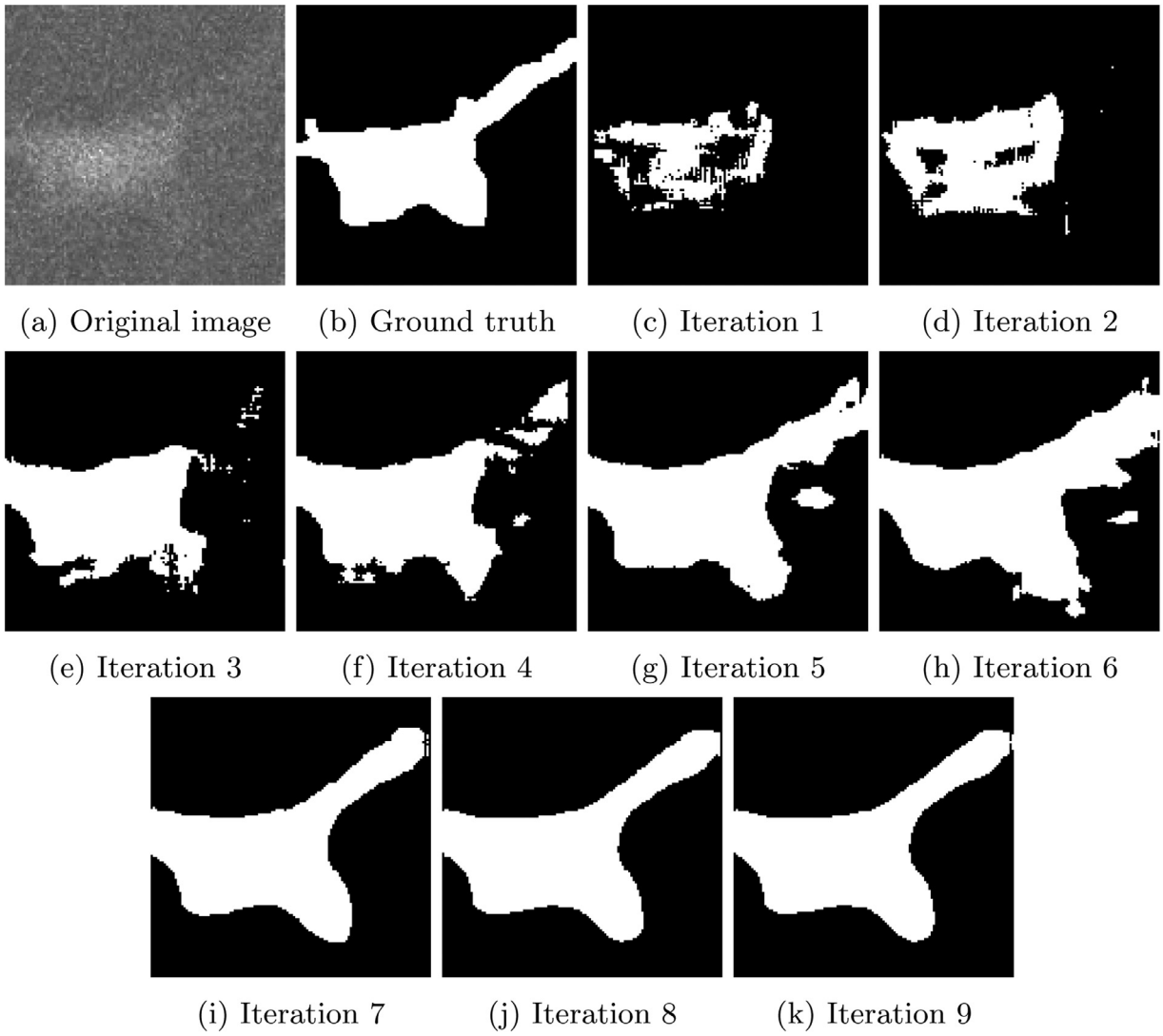

Fig. 7. (a) Original fluorescence microscopy image of rat mesenchymal stem cells (Fluo-C2DL-MSC) from the ISBI Cell Tracking Challenge, (b) corresponding ground truth, and (c)-(k) segmentation results of GRUU-Net for different iterations.

CPN (Akram et al., 2017): A U-Net is used for cell segmentation and a Faster R-CNN (Ren et al., 2015) for cell detection. The result of the Faster R-CNN is used by ROI pooling to crop features from the U-Net to improve cell splitting.

HD-Har (Harder et al., 2009): Local thresholding based on Otsu's method on a Gaussian filtered image is used after Gaussian filtering. A local threshold is computed if the intensity variance within an image patch is higher than a threshold, otherwise global Otsu thresholding is used.

Table 4
Results for the real 2D datasets of the Cell Tracking Challenge.

Dataset	Video	Method	SEG	Dice	Hausdorff
DIC-C2DH-HeLa	1	UP-PT	0.345		
		U-Net	0.327	0.880	52.404
		GRUU-Net (Weighted CE loss)	0.258	0.885	103.858
		GRUU-Net (Non-Normalized FL)	0.290	0.907	88.803
		GRUU-Net	**0.648**	**0.886**	**36.673**
	2	UP-PT	0.125		
		U-Net	0.219	0.853	63.463
		GRUU-Net (Weighted CE loss)	0.333	0.901	88.479
		GRUU-Net (Non-Normalized FL)	0.420	0.899	84.652
		GRUU-Net	**0.490**	**0.870**	**46.856**
Fluo-C2DL-MSC	1	UP-PT	0.382		
		HD-Har	**0.450**	0.593	109.631
		U-Net	0.408	**0.711**	**78.912**

(*continued on next page*)

Table 4 (*continued*)

Dataset	Video	Method	SEG	Dice	Hausdorff
		GRUU-Net (Weighted CE loss)	0.209	0.361	338.677
		GRUU-Net (Non-Normalized FL)	0.222	0.451	381.431
		GRUU-Net	0.329	0.620	84.126
	2	UP-PT	0.264		
		HD-Har	**0.598**	0.745	**101.842**
		U-Net	0.502	0.672	189.401
		GRUU-Net (Weighted CE loss)	0.535	0.793	290.017
		GRUU-Net (Non-Normalized FL)	0.543	0.792	293.935
		GRUU-Net	0.550	**0.772**	137.963
Fluo-N2DH-GOWT1	1	UP-PT	0.703		
		HD-Har	0.545	0.883	**6.833**
		CPN	0.851		
		CVXELL	0.821	0.637	
		U-Net	0.814	0.864	23.219
		GRUU-Net (Weighted CE loss)	0.854	0.939	100.644
		GRUU-Net (Non-Normalized FL)	0.866	0.946	99.016
		GRUU-Net	**0.888**	**0.901**	43.788
	2	UP-PT	0.798		
		HD-Har	0.898	0.925	**8.080**
		CPN	0.873		
		CVXELL	0.913	0.894	
		U-Net	0.832	0.826	21.995
		GRUU-Net (Weighted CE loss)	0.843	0.929	176.479
		GRUU-Net (Non-Normalized FL)	0.840	0.926	60.839
		GRUU-Net	**0.929**	**0.956**	11.776
Fluo-N2DH-HeLa	1	UP-PT	0.627		
		HD-Har	0.744	**0.887**	9.943
		CPN	**0.831**		
		BLOB	0.795		
		U-Net	0.775	0.875	**6.674**
		GRUU-Net (Weighted CE loss)	0.706	0.838	91.530
		GRUU-Net (Non-Normalized FL)	0.788	0.888	1.500
		GRUU-Net	0.749	0.858	7.145
	2	UP-PT	0.709		
		HD-Har	0.814	0.897	**6.651**
		CPN	**0.845**		
		BLOB	0.839		
		U-Net	0.798	0.892	7.581
		GRUU-Net (Weighted CE loss)	0.813	0.899	7.193
		GRUU-Net (Non-Normalized FL)	0.788	0.901	7.009
		GRUU-Net	0.809	**0.911**	7.341
PhC-C2DH-U373	1	UP-PT	0.356		
		CPN	0.734		
		GC-ME	0.875		
		U-Net	0.812	0.869	59.156
		GRUU-Net (Weighted CE loss)	0.926	0.930	57.507
		GRUU-Net (Non-Normalized FL)	0.922	0.941	53.957
		GRUU-Net	**0.938**	**0.942**	**47.463**
	2	UP-PT	0.359		
		CPN	0.738		
		GC-ME	0.757		
		U-Net	0.739	0.791	71.665
		GRUU-Net (Weighted CE loss)	0.787	0.859	75.490
		GRUU-Net (Non-Normalized FL)	0.796	0.874	42.402
		GRUU-Net	**0.814**	**0.889**	**34.513**
PhC-C2DL-PSC	1	UP-PT	0.514		
		HD-Har	0.464	**0.720**	**7.374**
		CPN	0.661		
		U-Net	0.347	0.663	8.141
		GRUU-Net (Weighted CE loss)	0.256	0.497	105.252
		GRUU-Net (Non-Normalized FL)	0.264	0.524	96.013
		GRUU-Net	**0.684**	0.711	9.142
	2	UP-PT	0.477		
		HD-Har	0.465	0.415	12.479
		CPN	**0.648**		
		U-Net	0.272	0.635	**8.520**
		GRUU-Net (Weighted CE loss)	0.311	0.121	39.735
		GRUU-Net (Non-Normalized FL)	0.329	0.598	100.996
		GRUU-Net	0.422	**0.686**	9.310

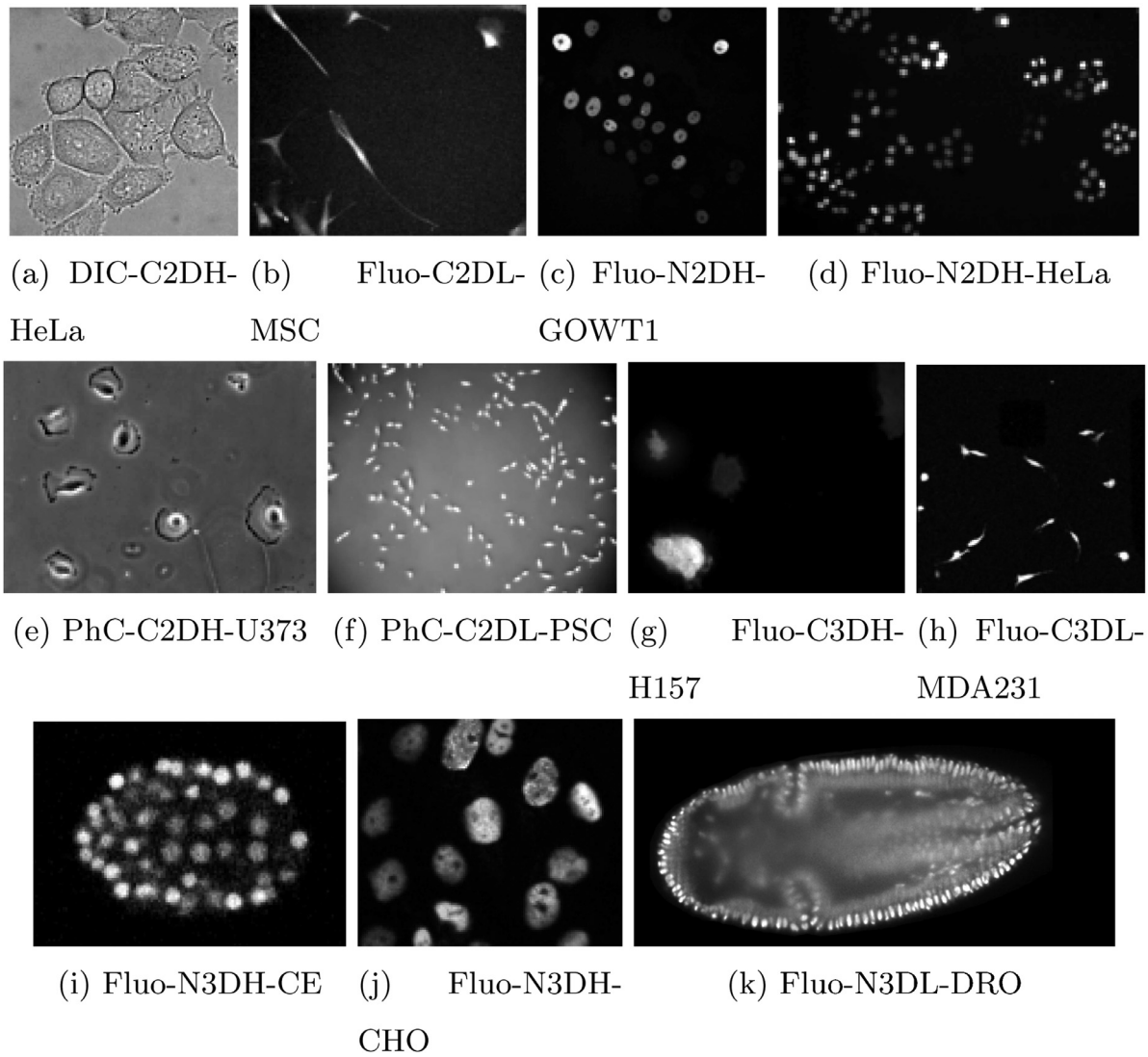

Fig. 8. Sample images showing the variability of image data in the Cell Tracking Challenge datasets (partially contrast-enhanced for better visibility).

CVXELL (Kostrykin et al., 2018): Ellipses are fitted to the regions of interest (ROIs) using a sequence of convex programs. The ROIs are determined using a blob detector and a modified Voronoi tessellation.

BLOB (Akram et al., 2016): Either graph-cuts or thresholding are used for initial segmentation. Generalized Laplacian of Gaussian (gLOG) filter banks and non-maxima suppression are employed to split cell clusters.

GC-ME (Bensch and Ronneberger, 2015): This method uses graph cuts with asymmetric boundary costs for cell segmentation.

UP-PT (Esteves et al., 2012): Non-maxima suppression is performed on the result of an LoG filter. The cell shape is determined using a local convergence filter.

From the results in Tables 4 and 5 it turns out that our method achieved the best performance for SEG for 13 out of 22 datasets, and was among the top two methods for 14 out of 22 datasets. For the Dice coefficient, our method was in 14 out of 20 datasets best, and among the top two methods for 17 datasets. We investigated whether GRUU-Net yields a statistically significant improvement for SEG and Dice compared to UP-PT and U-Net, which were applied to all datasets. A Shapiro–Wilk test revealed that the results for SEG and Dice do not follow a normal distribution. Therefore, a Wilcoxon signed-rank test was conducted with significance level of 5%. For the comparison of GRUU-Net with UP-PT we obtained $p < 0.001$ for SEG and Dice. GRUU-Net and U-Net yielded $p < 0.003$ for SEG and $p < 0.004$ for Dice. Thus, our method yields a statistically significant improvement over UP-PT and U-Net. Comparing the different variants of our network in Table 4, it turns out that the results are consistent with the results of the ablation study in Table 3. For some datasets, a relatively high Hausdorff distance was observed, which is an indication for missed objects (the Hausdorff distance was computed for the whole image). For some datasets (e.g., DIC-C2DH-HeLa, Fluo-C3DH-H157), it can be observed that U-Net overfitted much faster than our GRUU-Net, which is indicated by the maximum number of training iterations using early stopping (cf. Table 2). In addition, the cell appearance in the two videos for a dataset is quite different. Thus, the reason for the low performance is probably that the networks overfitted on the specific appearance of one video and did not generalize well to the other video. Partially, classical methods that do not use machine learning performed quite well. However, these methods were probably tuned based on all training and challenge data, which generally leads to overfitting. Since our method achieved

Table 5
Results for the real 3D datasets of the Cell Tracking Challenge.

Dataset	Video	Method	SEG	Dice	Hausdorff
Fluo-C3DH-H157	1	UP-PT	0.458		
		HD-Har	0.753	0.922	105.897
		U-Net	0.017	0.007	**21.664**
		GRUU-Net	**0.759**	**0.929**	29.216
	2	UP-PT	0.557		
		HD-Har	0.573	0.766	**36.825**
		U-Net	0.032	0.037	166.007
		GRUU-Net	**0.602**	**0.865**	55.383
Fluo-C3DL-MDA231	1	UP-PT	0.348		
		HD-Har	0.196	0.494	**59.969**
		U-Net	0.340	0.521	90.787
		GRUU-Net	**0.570**	**0.703**	75.506
	2	UP-PT	0.429		
		HD-Har	0.290	0.521	**5.663**
		U-Net	**0.516**	0.649	70.452
		GRUU-Net	0.503	**0.792**	12.657
Fluo-N3DH-CE	1	UP-PT	0.385		
		HD-Har	0.566	**0.772**	31.735
		U-Net	**0.627**	0.760	**17.844**
		GRUU-Net	0.598	0.716	19.485
	2	UP-PT	0.355		
		HD-Har	0.486	0.735	24.539
		U-Net	**0.636**	0.683	**20.256**
		GRUU-Net	**0.636**	**0.747**	34.010
Fluo-N3DH-CHO	1	UP-PT	0.625		
		HD-Har	**0.814**	**0.875**	**27.622**
		U-Net	0.579	0.661	29.887
		GRUU-Net	0.595	0.671	36.449
	2	UP-PT	0.682		
		HD-Har	**0.903**	**0.950**	**8.740**
		U-Net	0.746	0.815	19.961
		GRUU-Net	0.729	0.810	24.564
Fluo-N3DL-DRO	1	UP-PT	0.296		
		U-Net	0.423		
		GRUU-Net	**0.534**		
	2	UP-PT	0.205		
		U-Net	0.640		
		GRUU-Net	**0.709**		

the best results for SEG in most datasets, it can cope better with the high variability in the 2D and 3D datasets compared to previous methods. Recently, we participated in the Cell Segmentation Benchmark of the Cell Tracking Challenge at ISBI 2019 and our method achieved top-3 rankings in three categories.

4. Discussion and conclusion

We presented GRUU-Net, a new deep neural network which integrates convolutional neural networks and gated recurrent neural networks. Our method combines a convolutional GRU with a dense hourglass-shaped U-Net like CNN architecture for iterative, multi-scale feature aggregation and refinement. Our network has much less parameters (0.7 M) compared to a U-Net (1.9 M) and a Deconvolution Network (1.1 M). To increase the robustness of the training and improve segmentation, we introduced a novel normalized focal loss for momentum-based optimizers. Our focal loss did not only improve the segmentation result of our network but also the result of other deep neural networks such as the U-Net. The network was trained end-to-end from scratch using few example images. Compared to previous deep learning approaches, all layers in our model have access to features from all previous layers over a common memory at full resolution, which has the potential to improve the sharing of information and better gradient flow. Through learning a common feature representation over all scales and therefore introducing skip connections between all layers is expected to reduce overfitting when using only a limited number of training samples. We also presented a distributed scheme for data augmentation and optimized training of our GRUU-Net. A comprehensive evaluation of our method has been performed on challenging tissue microscopy images of glioblastoma nuclei. Our proposed method outperformed previous methods and we demonstrated the achieved improvements by the different introduced concepts. In addition, we benchmarked our method using a wide spectrum of all 22 real 2D and 3D microscopy image datasets from the Cell Tracking Challenge. Our method achieved superior or competitive results for the majority of the 22 datasets, although we trained our network using only a few example images, and did not employ hand-crafted weighting of the cross-entropy loss. Also, our network comprises only a reduced number of parameters. In addition, classical segmentation methods included in our evaluation, that do not rely on learning, were probably optimized directly on the target dataset which reduces the generalization ability. We applied our method to segmentation of objects in microscopy images, which has the potential to improve the results of subsequent tasks like object-wise classification, tracking, and clustering. In future work, we plan to apply our network to other real microscopy image data. In addition, we plan to use the concept of multi-scale feature aggregation and iterative refinement for object detection.

Declaration of Competing Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgments

Support of the BMBF within the projects CancerTelSys (e:Med, #01ZX1602) and de.NBI (HD-HuB, #031A537C) is gratefully acknowledged.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al., 2016. TensorFlow: a system for large-scale machine learning. In: Proc. OSDI, 16, pp. 265–283.
Akram, S.U., Kannala, J., Eklund, L., Heikkilä, J., 2016. Joint cell segmentation and tracking using cell proposals. In: Proc. ISBI. IEEE, pp. 920–924.
Akram, S.U., Kannala, J., Eklund, L., Heikkilä, J., 2017. Cell tracking via proposal generation and selection. arXiv: /1705.03386.
Arbelle, A., Raviv, T.R., 2018. Microscopy cell segmentation via adversarial neural networks. In: Proc. ISBI. IEEE, pp. 645–648.
Arthur, D., Vassilvitskii, S., 2007. k-means++: The advantages of careful seeding. In: Proc. ACM-SIAM. SIAM, pp. 1027–1035.
Ballas, N., Yao, L., Pal, C., Courville, A., 2015. Delving deeper into convolutional networks for learning video representations. arXiv: /1511.06432.
Baltissen, D., Wollmann, T., Gunkel, M., Chung, I., Erfle, H., Rippe, K., Rohr, K., 2018. Comparison of segmentation methods for tissue microscopy images of glioblastoma cells. In: Proc. ISBI. IEEE, pp. 770–778.
Bensch, R., Ronneberger, O., 2015. Cell segmentation and tracking in phase contrast images using graph cut with asymmetric boundary costs. In: Proc. ISBI. IEEE, pp. 1220–1223.
Bernsen, J., 1986. Dynamic thresholding of grey-level images. In: Proc. Int. Conf. on Pattern Recognition, pp. 1251–1255.
Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L., 2018. Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. TPAMI 40 (4), 834–848.
Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y., 2014. On the properties of neural machine translation: encoder-decoder approaches. arXiv: /1409.1259.
Drozdzal, M., Vorontsov, E., Chartrand, G., Kadoury, S., Pal, C., 2016. The importance of skip connections in biomedical image segmentation. In: Proc. MICCAI Workshop LABELS. Springer, pp. 179–187.
Esteves, T., Quelhas, P., Mendonça, A.M., Campilho, A., 2012. Gradient convergence filters and a phase congruency approach for in vivo cell nuclei detection. Mach. Vis Appl. 23 (4), 623–638.
Google Brain Team, 2019. Importing data - Tensorflow.
Harder, N., Mora-Bermúdez, F., Godinez, W.J., Wünsche, A., Eils, R., Ellenberg, J., Rohr, K., 2009. Automatic analysis of dividing cells in live cell movies to detect mitotic delays and correlate phenotypes in time. Genome Res. 19 (11), 2113–2124.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proc. ICCV, pp. 1026–1034.
He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proc. CVPR, pp. 770–778.
Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q., 2017. Densely connected convolutional networks. In: Proc. CVPR, 1. IEEE, pp. 2261–2269.
Jégou, S., Drozdzal, M., Vazquez, D., Romero, A., Bengio, Y., 2017. The one hundred layers tiramisu: fully convolutional densenets for semantic segmentation. In: Proc. CVPR. IEEE, pp. 1175–1183.
Kingma, D.P., Ba, L., 2015. ADAM: a method for stochastic optimization. In: Proc. ICLR.
Kostrykin, L., Schnörr, C., Rohr, K., 2018. Segmentation of cell nuclei using intensity-based model fitting and sequential convex programming. In: Proc. ISBI. IEEE, pp. 654–657.
Le, N., Quach, K.G., Luu, K., Savvides, M., Zhu, C., 2017. Reformulating level sets as deep recurrent neural network approach to semantic segmentation. arXiv: /1704.03593.
Liao, Q., Poggio, T., 2016. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. arXiv: /1604.03640.
Lin, H.W., Tegmark, M., Rolnick, D., 2017a. Why does deep and cheap learning work so well? J. Stat. Phys. (6) 1223–1247.
Lin, T.-Y., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J., 2017. Feature Pyramid Networks for object detection. In: Proc. CVPR, 1. IEEE, pp. 936–944.
Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017c. Focal loss for dense object detection. arXiv: /1708.02002.
Liu, Y., Gadepalli, K., Norouzi, M., Dahl, G.E., Kohlberger, T., Boyko, A., Venugopalan, S., Timofeev, A., Nelson, P.Q., Corrado, G.S., et al., 2017. Detecting cancer metastases on gigapixel pathology images. arXiv: /1703.02442.
Maas, A.L., Hannun, A.Y., Ng, A.Y., 2013. Rectifier nonlinearities improve neural network acoustic models. In: Proc. ICML, 30.
Magnusson, K.E., Jaldén, J., 2012. A batch algorithm using iterative application of the viterbi algorithm to track cells and construct cell lineages. In: Proc. ISBI. IEEE, pp. 382–385.
Maška, M., Ulman, V., Svoboda, D., Matula, P., Matula, P., Ederra, C., Urbiola, A., España, T., Venkatesan, S., Balak, D.M., et al., 2014. A benchmark for comparison of cell tracking algorithms. Bioinformatics 30 (11), 1609–1617.
Microsoft Research, 2019. CNTK 201: part B - image understanding.
Milletari, F., Navab, N., Ahmadi, S.-A., 2016. V-net: fully convolutional neural networks for volumetric medical image segmentation. In: Proc. 3DV. IEEE, pp. 565–571.
Noh, H., Hong, S., Han, B., 2015. Learning deconvolution network for semantic segmentation. In: Proc. ICCV. IEEE, pp. 1520–1528.
Odena, A., Dumoulin, V., Olah, C., 2016. Deconvolution and checkerboard artifacts. Distill 1 (10), e3.
Paszke, A., Chaurasia, A., Kim, S., Culurciello, E., 2016. ENet: a deep neural network architecture for real-time semantic segmentation. arXiv: /1606.02147.
Paszke, A., Gross, S., Chintala, S., Chanan, G., 2019. Data loading and processing tutorial.
Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., Liao, Q., 2017. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. IJAC 14 (5), 503–519.
Pohlen, T., Hermans, A., Mathias, M., Leibe, B., 2016. Full-Resolution Residual Networks for semantic segmentation in street scenes. arXiv: /1611.08323.
Reddi, S.J., Kale, S., Kumar, S., 2018. On the convergence of ADAM and beyond. In: Proc. ICLR.
Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster R-CNN: towards real-time object detection with region proposal networks. In: Proc. NIPS, pp. 91–99.
Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: convolutional networks for biomedical image segmentation. In: Proc. MICCAI. Springer, pp. 234–241.
Sadanandan, S.K., Ranefall, P., Le Guyader, S., Wählby, C., 2017. Automated training of deep convolutional neural networks for cell segmentation. Sci Rep 7 (1), 7860.
Sethian, J.A., 1996. A fast marching level set method for monotonically advancing fronts. Proc. Natl. Acad. Sci. USA 93 (4), 1591–1595.
Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv: /1409.1556.
Sommer, C., Straehle, C., Köthe, U., Hamprecht, F.A., 2011. Ilastik: interactive learning and segmentation toolkit. In: Proc. ISBI. IEEE, pp. 230–233.
Ulman, V., Maška, M., Magnusson, K.E., Ronneberger, O., Haubold, C., Harder, N., Matula, P., Matula, P., Svoboda, D., Radojevic, M., et al., 2017. An objective comparison of cell-tracking algorithms. Nat. Methods 14 (12), 1141–1152.
Wollmann, T., Eijkman, C.S., Rohr, K., 2018a. Adversarial domain adaptation to improve automatic breast cancer grading in lymph nodes. In: Proc. ISBI. IEEE.
Wollmann, T., Ivanova, J., Gunkel, M., Chung, I., Erfle, H., Rippe, K., Rohr, K., 2018b. Multi-channel deep transfer learning for nuclei segmentation in glioblastoma cell tissue images. In: Proc. BVM. Springer, pp. 316–321.
Wollmann, T., Rohr, K., 2017. Deep residual Hough voting for mitotic cell detection in histopathology images. In: Proc. ISBI. IEEE, pp. 341–344.
Yi, J., Wu, P., Hoeppner, D.J., Metaxas, D., 2018. Pixel-wise neural cell instance segmentation. In: Proc. ISBI. IEEE, pp. 373–377.
Yu, F., Koltun, V., 2015. Multi-scale context aggregation by dilated convolutions. arXiv: /1511.07122.
Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H., 2015. Conditional random fields as recurrent neural networks. In: Proc. ICCV, pp. 1529–1537.