Stable deep neural network architectures for mitochondria segmentation on electron microscopy volumes

Daniel Franco-Barranco[1,2], Arrate Muñoz-Barrutia[3,4], and Ignacio Arganda-Carreras[1,2,5]

[1] University of the Basque Country (UPV/EHU)
[2] Donostia International Physics Center (DIPC)
[3] Universidad Carlos III de Madrid
[4] Instituto de Investigación Sanitaria Gregorio Marañón
[5] Ikerbasque, Basque Foundation for Science
daniel.franco@dipc.org

Electron microscopy (EM) allows the identification of intracellular organelles such as mitochondria, providing insights for clinical and scientific studies. In recent years, a number of novel deep learning architectures have been published reporting superior performance, or even human-level accuracy, compared to previous approaches on public mitochondria segmentation datasets. Unfortunately, many of these publications do not make neither the code nor the full training details public to support the results obtained, leading to reproducibility issues and dubious model comparisons. For that reason, and following a recent code of best practices for reporting experimental results, we present an extensive study of the state-of-the-art deep learning architectures for the segmentation of mitochondria on EM volumes, and evaluate the impact in performance of different variations of 2D and 3D U-Net-like models for this task. To better understand the contribution of each component, a common set of pre- and post-processing operations has been implemented and tested with each approach. Moreover, an exhaustive sweep of hyperparameters values for all architectures have been performed and each configuration has been run multiple times to report the mean and standard deviation values of the evaluation metrics. Using this methodology, we found very stable architectures and hyperparameter configurations that consistently obtain state-of-the-art results in the well-known EPFL Hippocampus mitochondria segmentation dataset. Furthermore, we have benchmarked our proposed models on two other available datasets, Lucchi++ and Kasthuri++, where they outperform all previous works. The code derived from this research and its documentation are publicly available at https://github.com/danifranco/EM_Image_Segmentation.

1 Introduction

Recent imaging methods in electron microscopy (EM) allow scientists to identify subcellular organelles such as vesicles or mitochondria with nano-scale precision. In particular, mitochondria play an important role in some crucial functions in

the cell, such as energy production, signaling, differentiation, cell growth and death [52]. For that reason, the automated and accurate segmentation of mitochondria is specially relevant for basic research in neuroscience, but in clinical studies as well, since their number and morphology are related to severe diseases such as cancer [10,13,53], Parkinson disease [44] or Alzheimer disease [10].

In the past decade, advances in machine learning and computer vision, especially those based on deep learning, have helped scientists to automatically quantify the size and morphology of cells and organelles in microscopy images [41,37]. However, with an increasing number of deep learning-based bioimage segmentation publications every year, there is a lack of enough benchmarks for different image modalities and segmentation problems to compare state-of-the-art methods under the same conditions. Moreover, deep learning methods are usually too data-specialized, making it difficult to identify those approaches that perform well on datasets different from those they have been tested on [24]. On top of that, many of such approaches are published without their supporting code and image data, leading to major reproducibility and reliability problems. Such issues have not gone unnoticed and are increasingly in focus. They have become the main target even for recently proposed challenges[1] where the machine learning community aims at reproducing the computational experiments and verifying the empirical results already published at top venues.

As pointed out by recent works [3,24], while many publications insist on presenting architectural novelties, the overall performance of a network depends substantially on its corresponding pre-processing, training, inference and post-processing strategies. Even though such choices play a critical role in the final results, very often they tend to be omitted in the method descriptions and their comparisons with competing approaches.

Another issue inherent to the use of deep learning architectures (and frequently not discussed in publications) is the sometimes not negligible variability of the results produced by different executions of the same exact architecture and training configuration. Despite programmatically setting all initial random seeds, the non deterministic nature of the graphical processing units (GPUs) introduces variations from execution to execution, resulting on slightly different performances. This variability is usually not taken into account when presenting results, although it could be crucial to select models, training and inference strategies that repeatedly lead to stable results.

In the particular task of mitochondria segmentation, the *de facto* benchmark dataset adopted by the community is the EPFL Hippocampus dataset [36] (hereafter referred to as Lucchi dataset). Published in 2011, it contains two image volumes (training and test) of the same size, and their respective semantic segmentation labels are both public. As the reference in the field for a decade, many methods have been published proposing solutions for this dataset. Unfortunately, most of them suffer from the aforementioned problems, forcing other scientists to code their own versions of the published algorithms, many times

[1] https://paperswithcode.com/rc2020

knowing too few details about their original implementations, training and inference methodologies.

To address this deficiency in the field, we first re-implemented the top-performing deep learning architectures for the Lucchi dataset following the descriptions of their original publications. None of them led directly to their claimed results. After our own modifications, an extensive hyperparameter search and multiple runs of the same configuration, some of these methods occasionally achieved such results. Second, we compared the performance of state-of-the-art biomedical semantic segmentation architectures in the same dataset, evaluated under the same training and inference framework. In particular, we focused on the stability of the resulting metric values after several executions of the same configuration and scrutinized the impact of different popular post-processing and output reconstruction methods. Finally, based on our findings, we propose light encoder-decoder architectures that consistently lead to robust state-of-the-art results in the Lucchi dataset as well as in other public mitochondria segmentation datasets.

In brief, our main contributions are as follows:

1. We performed a thorough study on the reproducibility and stability of the top-performing deep learning segmentation methods published for the Lucchi dataset, exposing major issues to achieve their claimed results in a consistent manner.
2. We made a comprehensive comparison of the performance of the most popular deep learning architectures for biomedical segmentation using the Lucchi dataset, and show their stability under the same training and post-processing conditions.
3. We propose different variations of light-weight encoder-decoder architectures, together with a training/inference workflow, that lead to stable and robust results across mitochondria segmentation datasets.

The rest of this paper is structured as follows. In Section 2, we review the state of the art in biomedical semantic segmentation, with a special focus on mitochondria segmentation. In Section 3, we introduce our proposed architectures and the different post-processing and test-time evaluation methods. In Section 4, we introduce the datasets and evaluation metrics employed. We also show the results that support our findings together with an ablation study that unveils the contribution of every component of our proposed solution. Finally, in Section 5, the conclusions of this work are presented.

2 Related work

In the last decade, deep learning approaches have become dominant in the field of computer vision and its most common target applications [14,40] including semantic segmentation for biomedical image analysis [17,29]. In particular, the semantic segmentation problem aims at linking each pixel in an image to a class

label, producing an output of the same size as the input image. The first steps towards resolving this problem using deep learning were taken by means of fully convolution networks (FCNs) [31]. More specifically, fully connected layers were replaced by convolutional layers in some classic networks such as AlexNet [28], VGG [49] or GoogLeNet [50] and information from intermediate layers was fused to upsample the feature maps encoded by the network, finally producing a pixel-wise classification. This idea of *encoding* the image through a convolutional neural network (CNN), outputing a vector feature map (also called *bottleneck*), and recovering its original spatial shape in a *decoding* path was further extended in subsequent works [42,45,39,25,2,6].

A major breakthrough was the U-Net [45], that extended the encoding and decoding idea by making an upsampling path with up-convolutions after the bottleneck to recover the original image size. In addition, the authors proposed skip connections between the contracting and the expanding path, allowing the upsampling path to recover fine-grained details. The U-Net is the baseline of numerous approaches due to its success on multiple biomedical applications. For instance, U-Net-based methods have been proposed to segment, among others, polyps in colonoscopy videos [56], liver in abdominal computed tomography (CT) scans [56], pancreas in abdominal CT scans [48], brain magnetic resonance imaging scans [46], neurites [1,16] and synapses in brain EM images [4], dermoscopy images [23] or retina blood vessels in diabetic retinopathy images [57,26].

In the specific case of mitochondria segmentation, early works attempting to segment the Lucchi dataset [36] leveraged traditional image processing and machine learning techniques [34,33,32,35]. In their last two works, Lucchi *et al.* proposed alternative methodologies to segment mitochondria on their own dataset explicitly modeling mitochondria membranes [32,35]. From that last work, Casser *et al.* [5] inferred a Jaccard index or intersection over the union (IoU) lower bound value of 0.895 in the test set. The IoU is a common way of measuring the overlapping area between the ground truth and the produced segmentation with values that range from 0 to 1, where 0 means no overlap at all and 1 represents a perfect match (see Section 4.2).

More modern approaches made use of deep learning architectures to segment the Lucchi dataset. For instance, Oztel *et al.* [43] trained a CNN with four convolutional layers to classify 32×32 pixel patches extracted from the training data into two classes: mitochondria and background. After that, they fed the network with the full test images to simulate a sliding window process and applied three consecutive post-processing methods to improve the segmentation results: 1) spurious detection to remove small false blobs, 2) marker-controlled watershed transform [38] for border refinement, and 3) a median filtering to smooth labels along the z-axis. This way, they reported an IoU value of 0.907 in the test set, which is the highest reported value to date. Liu *et al.* [30] used instead a modified Mask Region-based CNN (Mask RCNN [18]) to detect and segment mitochondria. As post-processing methods they performed: 1) a morphological opening operation to eliminate small regions and smooth large ones, 2) a multi-layer fusion operation to exploit 3D mitochondria information, and 3)

a size-based filtering to remove tiny segmented objects that have an IoU score below a given threshold. As a result, they reported an IoU value of 0.849 in the test set. Cheng *et al.* [7] applied both a 2D and a 3D version of an asymmetric U-Net-like network. They introduced the *stochastic downsampling* method, that produces augmentation at feature-level in an operation they named *feature level augmentation*. More specifically, on that downsampling layer, they subdivided the image in fixed square regions and picked random rows and columns inside them to select the pixels/voxels that will constitute the downsampled output. Moreover, they implemented factorized convolutions [51] instead of classical ones to drastically reduce the number of network parameters. As best result, they reported an IoU value of 0.889 in the test set using their 3D network. Xiao *et al.* in [55] employed a variant of a 3D U-Net model with residual blocks. Additionally, in the decoder of the network they included two auxiliary outputs to address the vanishing gradient issue. Their final output is the result of the ensemble prediction of the 16 possible 3D variations (using flips and axis rotations) per each 3D subvolume. They reported an IoU value of 0.900 in the test set. Finally, in a more recent work, Casser *et al.* [5] presented a light version of a 2D U-Net aiming to achieve real-time segmentation. They reduced the number of network parameters to 2M and reported an IoU value of 0.890 applying median *Z-filtering* as post-processing method as well.

3 Methodology

Although an increasing number of methods are published every year proposing architectural modifications of a basic U-Net to perform biomedical segmentation, it is usually unclear if their claimed superiority is only due to an incomplete optimization of the basic network for the task at hand [24,3]. We hypothesize that, on top of answering that question, a full optimization can also lead to lightweight models that constantly produce stable and robust results across different datasets. To prove it, we explored basic U-Net configurations together with popular architectural tweaks such as residual connections [20] or attention gates [48]. Additionally, to disentangle the impact of each training choice, all configurations are run several times and their results are shown in the context of different post-processing and output reconstruction methods.

3.1 Proposed networks

Building upon the state of the art, we have explored different lightweight U-Net-like architectures in 2D and 3D. The general scheme for all architectures is represented in Figure 1, where our basic and Attention U-Net models use convolutional blocks as processing blocks (two 3×3 convolutional layers, see Figure 2a) and our Residual U-Net is formed by full pre-activation [21] residual blocks (two 3×3 convolutional layers with a shortcut as shown in Figure 2b). Both basic U-Net and Residual U-Net use concatenation as feature merge operation while our Attention U-Net introduces there an attention gate [48].

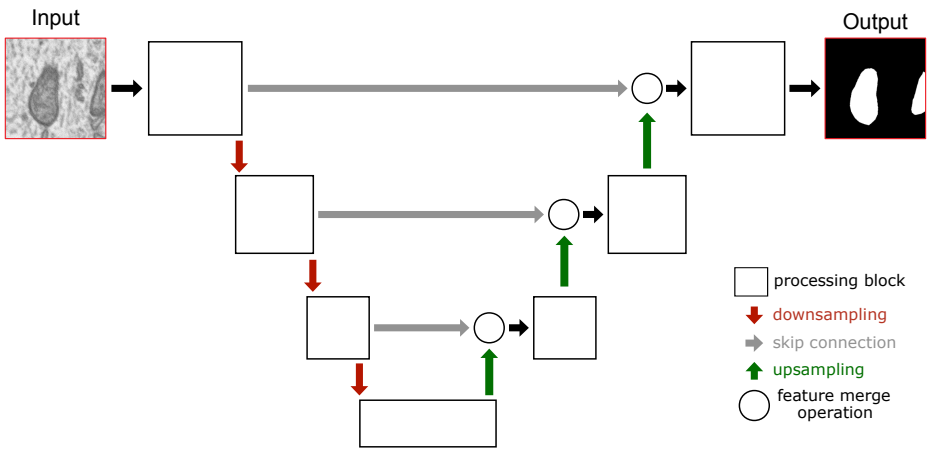


Fig. 1: Graphical representation of the proposed network architectures. Depending on the model of choice, the processing blocks can be either simply convolutional or residual blocks, while the feature merge operations may imply a single concatenation or an additional attention gate.

Next, we describe the best configuration found for each architecture, based in the thorough hyperparameter exploration described in the appendixes. Namely, the best performing architectures are as follows:

– **Basic U-Net**. In 2D, it is a 4-level U-Net with 16 filters in the initial level (reducing the amount of trainable parameters from over 31M in the original architecture [45] to less than 2M) that get double on each level, dropout in each block (from 0.1 up to 0.3 in the bottleneck and reversely, from 0.3 to 0.1 in the upsampling layers), ELU activation functions and transposed convolutions to perform the upsampling in the decoder.

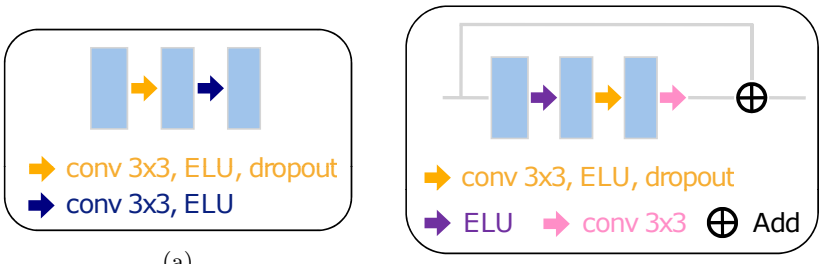


Fig. 2: Types of processing blocks: (a) convolutional block used in U-Net and Attention U-Net architectures, and (b) residual block used in Residual U-Net.

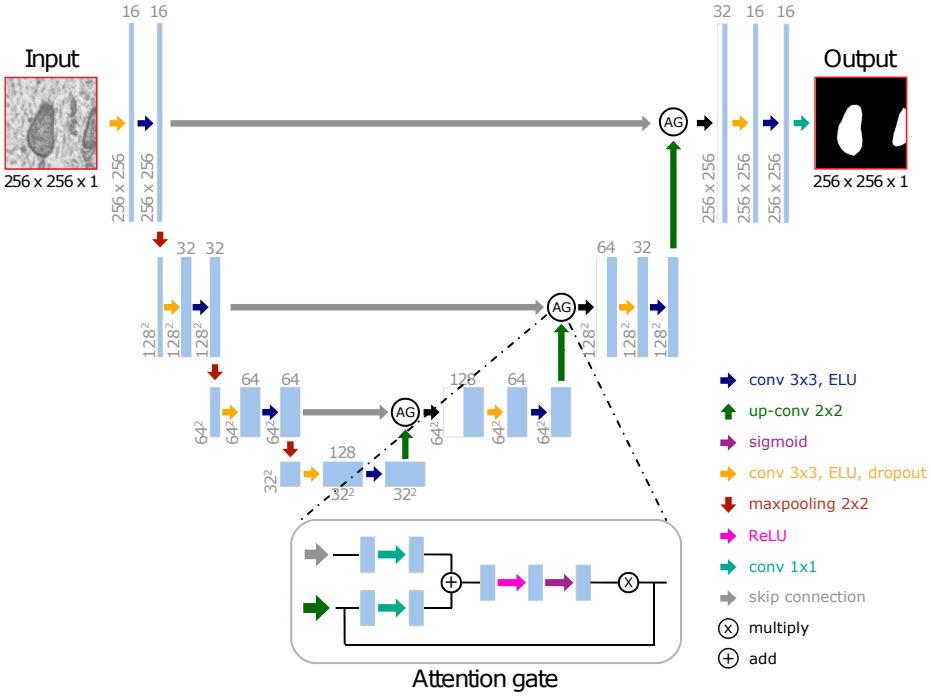


Fig. 3: Example of 2D Attention U-Net architecture with 3 downsampling levels and detailed description of the attention gates used in the skip connections.

In 3D, the architecture is very similar, but using depth 3, with 28, 36, 48 and 64 (in the bottleneck) 3D filters on each layer, what reduces the amount of trainable parameters to less than 0.8M.

– **Residual U-Net**. In 2D, this network is identical to our best basic U-Net architecture but swapping each convolutional block by a residual block [20].

  For the 3D residual approach, we achieved our best results going one level deeper than the non-residual 3D network. The number of feature maps used in this case are 28, 36, 48, 64 and 80 (bottleneck).

– **Attention U-Net**. These networks follow our 2D/3D basic U-Net architectures but incorporate attention gates [48] in the features passed by the skip connections. Such attention mechanism emphasizes salient feature maps that are in charge of the class decision and suppress irrelevant ones. This technique endows the network with the ability to focus on relevant regions of the image useful for a specific task. A full 3-level Attention U-Net is depicted in Figure 3.

3.2 Post-processing

As the network outputs are pixel-wise predictions, it is common practice to apply basic post-processing methods as an easy way to improve the results. In particular, we experimented with three different techniques and studied their impact in the final segmentation result:

- **Ensemble estimation.** Following a popular test-time data augmentation strategy to boost model performance, inference is applied on the multiples of 90° rotations and flipped versions of each image. Consequently, eight versions are created in 2D and 16 versions in 3D. Finally, the individual transformations are undone and the results are averaged into a final prediction for ensemble effect.
- **Blending overlapped patches.** When networks work on image patches instead of full images, the final prediction is reconstructed as a mosaic of the predictions of the patches. A recurrent problem of this strategy, as shown in Figure 4, is that inference may produce jagged predictions on the borders of the predicted images. To solve this, a common approach consist on creating overlapping patches and smoothly blending the resulting predictions using a second order spline window function[2]. Due to its computational cost, we only experimented with this technique in 2D.
- **Median Z-filtering.** A simple median filter along the Z-axis [5,43] can be used to correct label predictions in consecutive image slices.

3.3 Output reconstruction

During the training of deep networks, the input images are commonly divided into patches due to GPU memory limitations. Therefore, those patches need to be merged back together to form the final output at full-image size. In some publications, the authors specify clearly the way they infer and merge their predictions [55], while in others this process is not described at all [7,43,5], hindering a direct comparison between methods' performance. As it shown in Section 4, the evaluation scores obtained vary substantially depending on the reconstruction strategy. For that reason, and following the code of good practices to show deep learning-based results proposed by Dodge *et al.* [11], all results presented in this paper state the reconstruction strategy used. Namely, the implemented options are as follows:

1. **Per patch:** The metric value corresponds to the average value over all patches.
2. **Per image (no overlap):** The patches are merged into a mosaic without overlap and the metric value is the average over the fully reconstructed images. Notice this option is only possible if the image size is a multiple of the patch size.

[2] https://github.com/Vooban/Smoothly-Blend-Image-Patches

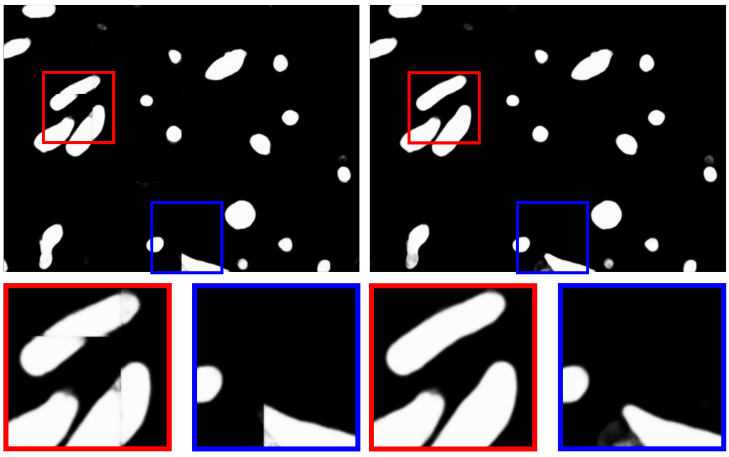


Fig. 4: Border effect in output image reconstruction. From left to right: output image reconstructed from patches with visible jagged predictions; and output image reconstructed using both the blending and ensemble techniques. Blue and red boxes show zoomed areas on both images.

3. **Per image (with 50% overlap):** The patches are merged together using 50% of overlap and the metric value is the average over all reconstructed images.
4. **Full image:** Inference is applied on the full-sized images and the metric value is the average over all images. Notice this strategy is not always feasible, since it depends on the input image size and the available GPU memory.

4 Experiments

In order to test our hypothesis and focusing on model reproducibility and stability, we conducted a thorough study on the top-performing segmentation methods recently published in the Lucchi dataset. Additionally, we introduce our own solutions, compare them with state-of-the-art approaches in biomedical semantic segmentation, and test them in other public datasets. In all our experiments, we present average scores obtained running the same configuration 10 times (hereafter referred as a *run*) together with the corresponding standard deviation.

4.1 Datasets

All the experiments performed in this work are based on the following publicly available datasets:

EPFL Hippocampus or Lucchi dataset. Introduced by Lucchi *et al.* [36], this dataset has since become the *de facto* standard for mitochondria segmentation in EM. The original volume represents a $5 \times 5 \times 5$ μm section of the CA1

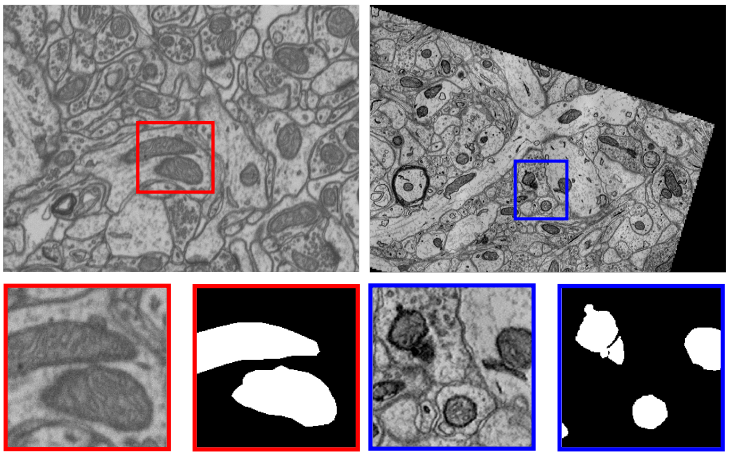


Fig. 5: Sample images from public mitochondria datasets. From left to right: Lucchi and Kasthuri++ data sample with their corresponding binary mask. Blue and red boxes show zoomed areas on both images.

hippocampus region of a mouse brain, with an isotropic resolution of $5 \times 5 \times 5$ nm per voxel. The volume of $2048 \times 1536 \times 1065$ voxels was acquired using focused ion beam scanning electron microscopy (FIB-SEM). The mitochondria of two subvolumes formed by 165 images of 1024×768 pixels were manually labeled by experts, and are commonly used as training and test data. An image sample is presented in Figure 5 (red).

Lucchi++ dataset. Presented by Casser *et al.* [5], this is a version of the Lucchi dataset after two neuroscientists and a senior biologist re-labeled mitochondria by fixing misclassifications and boundary inconsistencies.

Kasthuri++ dataset. Also presented by Casser *et al.* [5], this is a re-labeling of the mouse cortex dataset by Kasthuri *et al.* [27]. The volume corresponds to a part of the somatosensory cortex of an adult mouse and was acquired using serial section electron microscopy (ssEM). The train and test volume dimensions are $85 \times 1463 \times 1613$ voxels and $75 \times 1334 \times 1553$ voxels respectively, with an anisotropic resolution of $3 \times 3 \times 30$ nm per voxel. An image sample is presented in Figure 5 (blue).

4.2 Experimental setup

Evaluation metrics. Following common practice in the field, we evaluate our methods using the Jaccard index of the positive class or *foreground IoU*.

The foreground IoU is defined as follows:

$$IoU_F(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP + FP + FN} \tag{1}$$

where A is the segmentation proposal, B is the ground truth, TP are the true positives, FP the false positives and FN are the false negatives. As a convention, the positive class is foreground and the negative class is background. The background IoU is defined likewise by swapping the positive and negative classes. To obtain these values, the probability image returned by the network is binarized using a threshold value of 0.5.

Nevertheless, to compare our results with other related works we also define the *overall IoU* as the average of the foreground and background IoU:

$$IoU_O = \frac{(IoU_F + IoU_B)}{2} \tag{2}$$

where IoU_F and IoU_B are the foreground and background IoU, respectively. Notice the high proportion of background pixels typically inflates the overall IoU score, resulting in greater values than the foreground IoU.

Training setup and data augmentation. In order to find the best solutions, we made an exhaustive search of hyperparameters and training configurations, exploring different loss functions, optimizers, learning rates, batch sizes and data augmentation techniques. We explored as well the use of different input patch sizes, their selection method (random or systematic), and the discarding of image patches with low foreground class information as in [43]. When selecting a random patch, we define a probability map to choose patches with a higher probability of containing mitochondria, therefore addressing the class imbalance problem. Finally, we have also studied the effect of selecting the validation set as either consecutive training images or at random. Here we describe the best training configuration found. However, the details of this exhaustive search of the best training workflow can be consulted in the appendices.

In particular, we minimize the binary cross entropy (BCE) loss using the Stochastic Gradient Descent (SGD) optimizer, 0.99 momentum and no decay, with a learning rate of 0.002 and a batch size value of 6. The validation set is formed by 10% of the training images selected at random. We use a GeForce GTX 1080 GPU card to train the network for 360 epochs, completing an epoch when all training data is explored, with a patience established at 100 epochs monitoring the validation loss and picking up the model that performs best in the validation set. Moreover, we apply data augmentation on-the-fly making random rotations and vertical and horizontal flips. For the 3D networks, we employ elastic transformations as well (in 2D we did not observe an improvement).

4.3 Experiments on Lucchi dataset

4.3.1 Reproducing top state-of-the-art methods

In this section, we present a clear example of the difficulty of reproducing published methods when the training workflow is not fully described and the code is not provided. In particular, we aimed at reproducing the state-of-the-art deep learning-based methods that report top performance in the Lucchi dataset published by Cheng *et al.* [7], Casser *et al.* [5], Xiao *et al.* [55] and Oztel *et al.* [43].

Only the code by Casser *et al.* [5] is publicly available, so we plugged their network architecture into our training workflow. The code from the rest of methods was unsuccessfully requested to their corresponding authors.

In all cases, a first implementation attempt was made following the methodology and exact parameters described on each publication. When finding missing information, we proceeded using the most common practice in the field. On top of that, following the same procedure we use for our own models, we modified the original configuration (i.e., architecture and training workflow) aiming at improving the results and their stability (full details available in the appendices). These configurations are hereafter referred to as *original* and *modified* respectively. As can be seen in Table 1, although a systematic search of the best hyperparameters and training configurations was performed, the scores obtained in some cases do not reach the reported ones.

The original configuration of the 2D network presented by Cheng *et al.* [7] produces very unstable results (high standard deviation), probably due to the high learning rate employed (0.05), even though it is reduced when reaching the 50% and 75% of total epochs. Our modified configuration differs in the optimizer used (Adam instead of SGD) and learning rate (fixed to 0.0001). Additionally, we performed extra DA with random rotations, removed the dropout layers, reduced the number of epochs and extracted 12 random patches per training image instead of just one. Without post-processing (none is used in the original publication), the foreground IoU value reported (0.865) can only be reached through our modified configuration and by taking the maximum values of the 50% *overlap* or *full image* reconstruction strategies. Notice that even better values can be obtained thanks to post-processing.

The 3D approach of the same authors [7] produces IoU values close to 0 in its original form, since using the proposed learning rate (0.1), the network gets easily trapped in local minima. Moreover, the subvolume shape adopted, $128 \times 128 \times 96$ pixels, makes train/validation data splitting difficult, so we train the network until convergence with no validation data. Our modified configuration produces better results but far from the reported ones and highly unstable (0.800 in its best run vs the reported 0.889).

The original configuration proposed by Casser *et al.* [5] reaches high IoU values with high standard deviation as well. We modified their configuration selecting two random patches per training image instead of one and using a probability map to prioritize patches having mitochondria pixels in the center, which leads to more stable results. The maximum value was obtained applying Z-filtering to the predictions over full test images, measuring 0.870 of foreground IoU. In their original code, Casser *et al.* [5] optimized the training by using the test set as validation set, what could explain their better reported value.

Network	Param. number	Reported	Per Patch	Per Image (no overlap)		Per image (50% overlap)					Full Image		
					+Z-Fil.		+Ensemble	+Ensemble +Z-Fil.	+Blended Ensemble	+Blended Ensemble +Z. Fil.		+Ensemble	+Ensemble +Z-Fil.
Cheng 2D [7]	0.6M	0.865											
Original	0.59M		0.503±0.233	0.503±0.233	0.511±0.238	0.517±0.240	0.517±0.239	0.521±0.243	0.541±0.250	0.548±0.254	0.526±0.244	0.537±0.244	0.543±0.252
Modified	0.59M		0.848±0.012	0.843±0.012	0.852±0.011	0.851±0.011	0.863±0.010	0.868±0.010	0.865±0.008	0.871±0.008	0.853±0.011	0.865±0.009	0.871±0.008
Maximum	-		0.864	0.858	0.865	0.865	0.877	0.881	0.878	0.883	0.865	0.878	0.881
Casser [5]	1.96M	0.890											
Original	1.96M		0.824±0.014	0.817±0.016	0.828±0.016	0.815±0.016	0.825±0.013	0.831±0.013	0.831±0.011	0.838±0.011	0.820±0.016	0.833±0.011	0.839±0.012
Modified	1.96M		0.844±0.014	0.838±0.008	0.845±0.009	0.837±0.008	0.846±0.016	0.850±0.017	0.850±0.016	0.855±0.017	0.842±0.006	0.853±0.015	0.858±0.015
Maximum	-		0.846	0.844	0.852	0.846	0.861	0.865	0.862	0.867	0.848	0.865	0.870
Oztel [43]	0.14M	0.907											
Original	0.14M		-	-	-	-	-	-	-	-	0.425±0.080	0.457±0.060	0.466±0.061
Modified	0.07M			-	-	-	-	-	-	-	0.451±0.042	0.476±0.049	0.487±0.053
Maximum	-		-	-	-	-	-	-	-	-	0.500	0.531	0.544
Cheng 3D [7]	0.63M	0.889											
Original	0.79M		0.053±0.000(†)	0.053±0.000(†)	0.053±0.000(†)	0.053±0.000	0.053±0.000	0.053±0.000	-	-	-	-	-
Modified	0.79M		0.623±0.039(†)	0.691±0.049(†)	0.693±0.049(†)	0.714±0.040	0.0737±0.034	0.738±0.034	-	-	-	-	-
Maximum	-		0.694	0.777	0.779	0.787	0.799	0.800	-	-	-	-	-
Xiao [55]	1.1M	0.900											
Original	1.08M		0.874±0.003(†)	0.863±0.004(†)	0.864±0.004(†)	0.863±0.004	0.866±0.004	0.867±0.004	-	-	-	-	-
Modified	1.08M		0.882±0.002(†)	0.873±0.003(†)	0.874±0.003(†)	0.872±0.003	0.874±0.003	0.874±0.003	-	-	-	-	-
Maximum	-		0.885	0.879	0.880	0.880	0.880	0.880	-	-	-	-	-

Table 1: Foreground IoU (mean±standard deviation) of reproduced state-of-the-art works in Lucchi dataset. Different scores discussed in previous sections are shown, the post-processing methods adopted are indicated (*Z-Fil.* refers to Z-filtering). *Blended Ensemble* refers to combining blending and ensemble estimation. *Original* versions refer to exact configurations as reported by the authors. *Modified* corresponds to our best approach modifying *Original* in some way to improve method's performance and results stability. The patch size and overlap (marked with †) in each work is as follows: 256×256 pixels for Cheng 2D, $128 \times 128 \times 96$ voxels ($0 \times 0 \times 27$ voxels overlap in $x \times y \times z$) for the subvolumes in Cheng 3D; 512×512 pixels (256×0 pixels overlap in $x \times y$) in Casser; $448 \times 576 \times 20$ voxels ($128 \times 128 \times 10$ voxels overlap in $x \times y \times z$) in Xiao; and 768×1024 pixels in Oztel.

Xiao *et al.* [55] provided a detailed explanation of their training procedure, architecture and output reconstruction strategy. Thus, the unique modification that we added is the use of elastic transformations in DA. As it is shown in Table 1, this change improves substantially the results obtained. They merge the predictions with overlap and ensemble, so to be fair, the maximum value of patch merging using 50% overlap and ensemble predictions should be used for comparison. They reported 0.900 of foreground IoU compared to the maximum 0.880 achieved by our modified version.

Finally, the original configuration proposed by Oztel *et al.* [43] produces very low foreground IoU values. Indeed, the amount of relevant details regarding the architecture, hyperparameter and post-processing methods that are missing in the original publication is remarkable. Thus, even after modifying their network by adding more non-linearities (ReLU), changing the dropout values or the feature maps used, the results obtained are far from those presented by the authors. The number of the original network parameters compared with other state-of-the-art approaches is also relatively low, only 0.14M, which in our opinion are insufficient to capture mitochondria shapes. Furthermore, we implemented their post-processing pipeline, whose results are presented in Table 2. We adapted these post-processing methods to specifically improve the segmentation made by the proposed network. Although the segmentation is improved by a large margin, it is still far to achieve the IoU reported by the authors.

	Full Image	**Spurious Detection**	**Watershed**	**Z-Filtering**
Original	0.425±0.080	0.426±0.091	0.540±0.100	0.573±0.106
Modified	0.451±0.042	0.449±0.067	0.562±0.057	0.599±0.067
Maximum	0.500	0.539	0.619	0.683

Table 2: Foreground IoU results by the original and modified configurations of Oztel *et al.* [43] using their consecutive post-processing methods, i.e., *Spurious Detection* is applied over *Full Images*, then they are passed through *Watershed*, and finally through Z-filtering.

4.3.2 Proposed networks vs state-of-the-art networks for semantic segmentation

In this section, we introduce the performance of our proposed architectures together with a study in-depth of the main state-of-the-art semantic segmentation networks for natural and biomedical images. Namely, FCN 8/32 [31], MultiResUNet [23], MNet [12], Tiramisu [25], U-Net++ [56], 3D Vanilla U-Net [8] and nnU-Net [24]. All implementations have been obtained or ported from their official sites and all networks have been optimized under the same conditions: same

training and validation partitions, DA transformations, optimizers and learning rate ranges (see appendices). The case of the nnU-Net is special, since it is designed to optimize the whole segmentation pipeline. In order to compare it in equal conditions with the other approaches, we extract the optimal architecture following the nnU-Net regular processing and plugged it into our own workflow.

For all 2D networks, we use an input patch size of 256×256 pixels, while for the 3D networks we use $80 \times 80 \times 80$ voxels subvolumes to exploit the isotropic resolution of the Lucchi dataset. Notice the difference between the input shapes of 2D and 3D architectures makes them not directly comparable when looking at the metric values obtained per patch or with a 0% overlap output reconstruction, since some overlap was needed in 3D to infer the whole test volume. Therefore, for a fair comparison between 2D and 3D networks, we refer to the results based on output reconstructions using 50% of patch overlap.

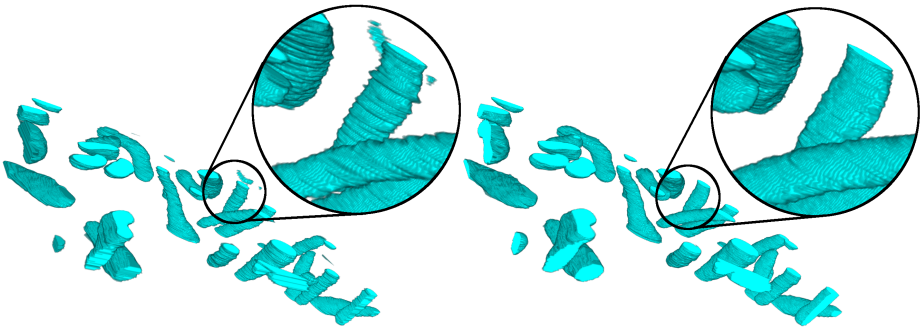
The results from the best configuration found for each network are shown in Table 3. Notice the 3D networks do not have results using full image reconstructions due to GPU memory limitations, as the whole dataset should be fed to the network. In the same way, blending estimation was not implemented in 3D networks given their computational cost.

Performance of state-of-the-art biomedical segmentation networks. The results of Tiramisu [25], MNet [12], nnU-Net [24], MultiResUNet [23] and 3D Vanilla U-Net [8] are below 0.880 of foreground IoU even when using output reconstructions with 50% of overlap and post-processing techniques such as ensemble predictions or Z-filtering. On top of these networks, the U-Net++ achieved the best results, scoring 0.881 ± 0.004 of foreground IoU. The 3D Vanilla U-Net, nnU-Net, U-Net++ and MNet seem to produce stable results (low standard deviation), while Tiramisu and MultiResUNet have larger variability within their results. Besides that, the difference in their number of trainable parameters is remarkable. The 3D Vanilla U-Net, nnU-Net and U-Net++ models have between $2\times$ and $5\times$ more parameters than the other state-of-the-art approaches.

With respect to the FCN networks [31], the FCN32 reports low IoU values that do not reach state-of-the-art results, but the FCN8 achieves results comparable with our best 2D U-Net configuration. Nevertheless, the number of trainable parameters in FCN8 is 50.38M compared to less than 2M in our proposed 2D models.

Network	Param. number	Per Patch	Per image (no overlap)		Per image (50% overlap)					Full Image		
				+Z-Fil.		+Ensemble	+Ensemble +Z-Fil.	+Blended Ensemble	+Blended Ensemble + Z-Fil.		+Ensemble	+Ensemble +Z-Fil.
FCN 32 [9]	50.38M	0.040±0.000	0.637±0.005	0.640±0.005	0.677±0.005	0.679±0.006	0.680±0.006	0.659±0.004	0.661±0.004	0.657±0.003	0.659±0.003	0.660±0.003
MultiResUNet [23]	7.26M	0.815±0.000	0.812±0.016	0.821±0.015	0.814±0.014	0.820±0.010	0.824±0.010	0.834±0.010	0.840±0.009	0.828±0.016	0.833±0.010	0.839±0.010
Tiramisu [25]	9.4M	0.810±0.028	0.809±0.030	0.821±0.029	0.833±0.027	0.851±0.018	0.857±0.017	0.850±0.016	0.855±0.016	0.830±0.029	0.846±0.019	0.851±0.018
MNet [12]	8.54M	0.851±0.011	0.854±0.009	0.861±0.009	0.865±0.008	0.870±0.007	0.874±0.007	0.874±0.006	0.878±0.006	0.867±0.008	0.872±0.006	0.876±0.008
nnU-Net [24]	52.1M	0.845±0.009	0.845±0.009	0.853±0.010	0.854±0.011	0.872±0.005	0.876±0.006	0.876±0.006	0.881±0.005	0.799±0.052	0.788±0.066	0.790±0.068
U-Net++ [56]	37.7M	0.731±0.014	0.860±0.008	0.867±0.008	0.872±0.005	0.877±0.004	0.881±0.004	0.880±0.003	0.884±0.003	0.875±0.004	0.878±0.003	0.882±0.003
2D Residual U-Net (ours)	2.03M	0.867±0.005	0.864±0.005	0.871±0.006	0.873±0.005	0.877±0.004	0.880±0.004	0.878±0.003	0.882±0.003	0.875±0.004	0.877±0.003	0.880±0.004
2D SE U-Net (ours)	1.95M	0.863±0.002	0.861±0.003	0.869±0.003	0.873±0.003	0.878±0.003	0.882±0.003	0.880±0.003	0.883±0.003	0.875±0.002	0.881±0.002	0.881±0.002
FCN 8 [9]	50.38M	0.860±0.005	0.864±0.005	0.871±0.005	0.880±0.003	0.884±0.002	0.888±0.002	**0.887±0.002**	0.891±0.002	0.881±0.003	0.886±0.002	**0.891±0.002**
2D U-Net (ours)	**1.95M**	0.874±0.003	0.872±0.003	0.880±0.003	0.881±0.002	0.884±0.002	0.888±0.002	0.884±0.000	0.889±0.002	0.882±0.003	0.884±0.002	0.887±0.003
2D Attention U-Net (ours)	1.99M	**0.875±0.004**	**0.873±0.003**	**0.882±0.003**	**0.882±0.003**	**0.885±0.001**	**0.890±0.002**	0.886±0.001	**0.892±0.001**	**0.884±0.002**	**0.886±0.001**	0.890±0.002
3D Vanilla U-Net [8]	19.07M	0.402±0.005(†)	0.842±0.004(†)	0.844±0.005(†)	0.851±0.004	0.857±0.006	0.857±0.006	-	-	-	-	-
3D SE U-Net (ours)	0.79M	0.387±0.007(†)	0.854±0.013(†)	0.855±0.013(†)	0.867±0.009	0.873±0.007	0.874±0.007	-	-	-	-	-
3D Attention U-Net (ours)	0.79M	0.389±0.005(†)	0.856±0.003(†)	0.857±0.003(†)	0.870±0.003	0.876±0.003	0.876±0.003	-	-	-	-	-
3D U-Net (ours)	**0.79M**	0.394±0.005(†)	**0.858±0.007**(†)	**0.859±0.007**(†)	0.871±0.006	0.878±0.004	0.878±0.004	-	-	-	-	-
3D Residual U-Net (ours)	1.50M	**0.394±0.004**(†)	0.857±0.004(†)	0.858±0.004(†)	**0.877±0.004**	**0.883±0.002**	**0.883±0.002**	-	-	-	-	-

Table 3: Performance of proposed networks and state-of-the-art networks for semantic segmentation in the Lucchi dataset. All values represent the foreground IoU (mean±standard deviation). Scores are shown using the different post-processing methods adopted (*Z-Fil.* refers to Z-filtering). *Blended ensemble* refers to combining blending and ensemble estimation. In 3D patches a minimum overlap was required so they are marked with †. Best results of each column and type of network (2D or 3D) are shown in bold.



(a) (b)

Fig. 6: 3D view of mitochondria labels on: (a) Lucchi and (b) Lucchi++ datasets. Labels in Lucchi are more jagged than in Lucchi++, penalizing 3D networks performance.

Performance of our proposed networks. Regarding our proposed approaches (Section 3.1), the best values were obtained with the 2D U-Net and its version with attention gates: 0.888 ± 0.002 and 0.890 ± 0.002 respectively applying ensemble estimation and Z-filtering post-processing. Our 3D networks do not reach the performance obtained with 2D versions. This may be explained by, first, the difference between the number of data samples. In 2D, with patches of 256×256 pixels, the training set has 1980 samples, while in 3D, with a subvolume size of $80 \times 80 \times 80$ voxels, the number of training samples without overlapping areas results in 216. Nevertheless, we try to alleviate this gap by means of elastic transformations in DA. Second, while inspecting mitochondria labels in 3D, we observed they frequently lose shape continuity through slices, penalizing the learning capacity of 3D networks (see Figure 6a). Remarkably, our 3D networks have three times less training parameters than our 2D approaches, leading to more computationally efficient models.

Finally, to complete the overview of the state-of-the-art networks and architectures, we experimented with *Squeeze-and-Excitation* (SE) blocks [22] in our proposed 2D and 3D models. These blocks perform dynamic channel-wise feature recalibration by *squeeze* and *excite* operations. The *Squeeze* operation consists in collecting global spatial information into a channel descriptor using global average pooling. After that, features are recalibrated by the *excite* operation, which emphasizes channel-wise features with a simple gating mechanism based on a ReLU and a Sigmoid activation. Their best results are obtained with SE blocks everywhere except the bottleneck, as suggested by [47]. Nevertheless, we experimented as well with inserting SE blocks after every convolutional layer. As shown in Table 3, these blocks do not imply a boost in performance in this case.

Description	Implementation	Code	Foreground IoU		Overall IoU	
			Reported	Reproduced	Reported	Reproduced
FCN 32	Ours using [9]	✓	0.688	0.688 (0.680±0.006)	0.835	0.835 (0.831±0.003)
MultiResUNet	Ours using [23]	✓	0.847	0.847 (0.824±0.010)	0.919	0.919 (0.902±0.007)
2D CNN	Cheng [7]		0.865	0.883 (0.871±0.008)	-	0.938 (0.932±0.004)
3D Vanilla U-Net	Ours using [8]	✓	0.866	0.866 (0.857±0.006)	0.929	0.929 (0.924±0.003)
Tiramisu	Ours using [25]	✓	0.872	0.872 (0.857±0.017)	0.932	0.932 (0.924±0.009)
2D U-Net	Casser [5]	✓	0.878	0.865 (0.853±0.015)	0.935	0.930 (0.922±0.007)
3D SE U-Net	Ours	✓	0.879	0.879 (0.874±0.007)	0.936	0.936 (0.933±0.004)
3D Attention U-Net	Ours	✓	0.880	0.880 (0.876±0.003)	0.936	0.936 (0.934±0.002)
nnU-Net framework	Isensee [24]	✓	0.882	-	0.938	-
MNet	Ours using [12]	✓	0.883	0.883 (0.874±0.007)	0.938	0.938 (0.929±0.004)
2D Residual U-Net	Ours	✓	0.885	0.885 (0.880±0.004)	0.939	0.939 (0.937±0.002)
3D U-Net	Ours	✓	0.885	0.885 (0.878±0.004)	0.939	0.939 (0.935±0.002)
nnU-Net	Ours using [24]	✓	0.888	0.888 (0.881±0.005)	0.941	0.941 (0.937±0.003)
3D Residual U-Net	Ours	✓	0.888	0.888 (0.883±0.002)	0.941	0.941 (0.938±0.001)
2D SE U-Net	Ours	✓	0.888	0.888 (0.882±0.003)	0.941	0.941 (0.937±0.002)
U-Net++	Ours using [56]	✓	0.888	0.888 (0.884±0.003)	0.941	0.941 (0.938±0.001)
3D CNN	Cheng [7]		0.889	0.800 (0.738±0.034)	-	0.894 (0.860±0.018)
2D U-Net+Z-filtering	Casser [5]	✓	0.890	0.870 (0.858±0.015)	0.942	0.931 (0.925±0.007)
FCN 8	Ours using [9]	✓	0.893	**0.893 (0.888±0.002)**	**0.943**	**0.943 (0.941±0.001)**
2D U-Net	Ours	✓	0.893	**0.893 (0.888±0.002)**	0.942	0.942 (0.941±0.001)
2D Attention U-Net	Ours	✓	0.893	**0.893 (0.890±0.002)**	**0.943**	**0.943 (0.942±0.001)**
3D U-Net	Xiao [55]		0.900	0.881 (0.875±0.003)	-	0.937 (0.934±0.002)
CNN+3 Post-proc.	Oztel [43]		**0.907**	0.683 (0.599±0.067)	-	0.800 (0.757±0.106)

Table 4: Reported vs. reproduced scores in the Lucchi dataset. The *Reported* columns correspond to the best score claimed by authors of published works or the maximum score obtained during our experiments for entries without associated publication. The *Reproduced* columns contain the maximum, mean and standard deviation values obtained while reproducing each corresponding method. Best scores of each column are presented in bold.

4.3.3 Comparison with reported results

To expose the sometimes large differences between the metric values reported in publications and those obtained by reproducing the very same methods, we have summarized in Table 4 the results of the top-performing published methods, together with those of state-of-the-art approaches and our proposed networks. Notice all reproduced values correspond to the best configuration found, i.e., using the optimal pre-processing, architecture, output reconstruction and post-processing strategies for each method. The availability of original code, including that of the present paper, is also indicated in the table. Following the traditional reporting of the results, the table is ordered by increasing value of reported foreground IoU score. Notice the gap between the averaged IoU and the reported value increases with the standard deviation, underling the importance of finding stable configurations (with low standard deviations) so as not to depend on a large computation budget [11].

Our proposed 2D U-Net and 2D Attention U-Net models, together with the FCN8 model reached the highest reproducible foreground IoU score with a value of 0.893. In particular, the 2D Attention U-Net achieved an slightly higher average score in a very consistent manner. Best values were obtained using blending and ensemble for output reconstruction and Z-filtering as post-processing (see Figure A.1 for an example of some of the proposed networks' predictions). As opposed to other approaches shown in Table 4, the standard deviation of our results is consistently low, guaranteeing good performance and reducing the number of experiments needed to reach optimal segmentations.

As expected, the lack of code associated with a publication enormously hinders the reproduction of the claimed results. Interestingly, in the case of the 2D approach from Cheng *et al.* [7], our implementation improved over their published results, stressing the benefits of optimizing the whole segmentation workflow. Notice there are two table entries for results with nnU-Net [24]: one using their entire training framework, and one plugging the best architecture found by their framework into ours.

4.3.4 Ablation Study

To investigate the impact and relevance of each component in our proposed networks, we performed an ablation study of our 2D U-Net architecture, as the rest of the proposed models are based on it. We compared six ablated versions with incremental changes: 1) a baseline four-level 2D U-Net model containing ReLU activations, Glorot/Xavier uniform kernel initialization [15], 16 feature maps in the first level of the network that are doubled on each level, and no regularization or DA; 2) the baseline with basic DA (random rotations and horizontal and vertical flips); 3) adding dropout as regularization method; 4) using ELU as activation function ($\alpha = 1$); 5) using *He normal* [19] as kernel initialization; 6) adding attention gates [48] in the skip connections.

The quantitative evaluation results on the Lucchi dataset for each case are shown in Table 5, where we divide the results using different evaluation frameworks introduced in Section 3.3. As it can be appreciated, the IoU values vary

	Foreground IoU		
Method	**Per Patch**	**50% Overlap**	**Full Image**
Baseline - 2D U-Net	0.725±0.020	0.748±0.027	0.739±0.002
+ DA	0.856±0.007	0.872±0.003	0.871±0.004
+ Dropout	0.867±0.003	0.880±0.002	0.881±0.002
+ ELU activation	0.873±0.003	0.880±0.001	0.881±0.002
+ He initializer	0.873±0.003	0.880±0.002	0.881±0.003
+ Attention Gates	**0.875±0.003**	**0.882±0.003**	**0.884±0.002**

Table 5: Ablation study or our full 2D model. From the top to the bottom, on each row, incremental modifications are applied based on the previous configuration. *DA* refers to applying data augmentation to the *Baseline* configuration, *Dropout* corresponds to applying dropout to the *Baseline+DA* configuration and so on.

significantly if they are provided by patch or by reconstructing the final output, highlighting once more the need of specifying the framework chosen when presenting the results.

As can be seen, the use of DA, together with dropout, clearly outperforms the baseline architecture by a large margin. In the same way, the usage of ELU improves over the use of ReLU activation functions. Conversely, changing the kernel initialization from Glorot uniform to He normal has marginal effects in the final result, so either can be used. Finally, introducing attention in the skip connections, as suggested in [48], helped increasing the network performance and maintaining results stability.

4.4 Results on Lucchi++ and Kasthuri++

Aiming to test how well the best solutions found for Lucchi would generalize in other datasets, we applied the same exact configurations to the Lucchi++ and Kasthuri++ datasets (see Section 4.3.2) and compared their performance with that reported by [5]. A full summary of the results is shown in Table 6, where we can see our models outperform all previously reported results by a large margin.

Notice the Kasthuri++ dataset is anisotropic and contains lower resolution in the z-axis. Therefore, we modified our proposed 3D networks by removing the downsampling in that axis in their pooling operations and using shallower architectures (three levels instead of four). A full description of the configurations tested can be found in the supplementary material.

5 Conclusions

By a complete experimental study of state-of-the-art deep learning models with modern training workflows, we have revealed significant problems of reproducibil-

Dataset	Description	Author	Foreground IoU		Overall IoU	
			Maximum	(mean±std)	Maximum	(mean±std)
Lucchi++	2D U-Net	Casser [5]	0.888	-	0.940	-
	2D U-Net+Z Filtering	Casser [5]	0.900	-	0.946	-
	2D Residual U-Net (†)	Ours	0.908	0.904±0.004	0.943	0.948±0.002
	2D U-Net (†)	Ours	0.916	0.911±0.006	0.955	0.952±0.003
	2D Attention U-Net (†)	Ours	0.919	0.914±0.003	0.956	0.954±0.001
	3D U-Net (¥)	Ours	0.923	0.915±0.007	0.958	0.954±0.004
	3D Attention U-Net (¥)	Ours	0.923	0.912±0.008	0.959	0.953±0.004
	3D Residual U-Net (¥)	Ours	**0.926**	0.919±0.005	**0.960**	0.957±0.003
Kasthuri++	2D U-Net	Casser [5]	0.845	-	0.920	-
	2D U-Net+Z Fil.	Casser [5]	0.846	-	0.920	-
	2D Residual U-Net (¥)	Ours	0.908	0.906±0.001	0.953	0.950±0.001
	2D Attention U-Net (¥)	Ours	0.915	0.913±0.001	0.956	0.954±0.001
	2D U-Net (¥)	Ours	0.916	0.913±0.002	0.955	0.954±0.001
	3D U-Net (¥)	Ours	0.934	0.932±0.001	0.965	0.965±0.001
	3D Residual U-Net (¥)	Ours	0.934	0.933±0.001	0.966	0.966±0.000
	3D Attention U-Net (¥)	Ours	**0.937**	0.934±0.001	**0.967**	0.966±0.001

(†) 0% overlap output reconstruction, blended ensemble and z-filtering post-processing
(¥) 50% overlap output reconstruction and ensemble post-processing

Table 6: Results obtained in the Lucchi++ and Kasthuri++ datasets.

ity in the domain of mitochondria segmentation in EM data. Moreover, by disentangling the effects of novel architectures from those of the training choices (i.e., pre-processing, data augmentation, output reconstruction and post-processing strategies) over a set of multiple executions of the same configurations, we have found stable lightweight models that consistently lead to state-of-the-art results on the existing public datasets.

Have novel methods reached human performance? To answer that question, Casser *et al.* [5] compared the results of human annotators in the Lucchi dataset, producing a foreground IoU value of 0.884. This would suggest that many of the models presented in Table 4 outperform indeed humans in this task. Nevertheless, all methods fell short of the threshold of 0.91 of foreground IoU, what could be due to the annotation inconsistencies discussed in Section 4.3.2. To investigate further, we created two slightly different versions of the mitochondria ground truth labels by 1-pixel morphological dilation and erosion. The foreground IoU value of the resulting labels against the original ground truth was 0.885 for the dilated version and 0.904 for the eroded one. Thus, this enforces the idea that the dataset is not pixel-level accurate, so it could be argued that all the methods with IoU values within a range of 0.009 or less can probably be considered to have similar performance. The same experiment was done with the ground truth labels of Lucchi++ and Kasthuri++. The foreground IoU values obtained dilating and eroding Lucchi++ were 0.898 and 0.919 respectively, while 0.927 and 0.922 were obtained in Kasthuri++. Indeed, even the average score of many of our models outperform those values (see Table 6). This suggests the performance on all three datasets has probably saturated, as new architectures

and training frameworks cannot improve beyond the limits inherent to semantic segmentation and the size of the datasets.

In closing, we believe further progress in mitochondria segmentation in EM will require (1) larger and more complex datasets [54], and (2) the adoption of a reproducibility checklist or set of best practices [11] to report more comprehensive results and allow robust future comparisons.

6 Acknowledgements

We acknowledge the support of Ministerio de Ciencia, Innovación y Universidades, Agencia Estatal de Investigación, under Grants TEC2016-78052-R and PID2019-109820RB-I00, MINECO/FEDER, UE, co-financed by European Regional Development Fund (ERDF), "A way of making Europe".

References

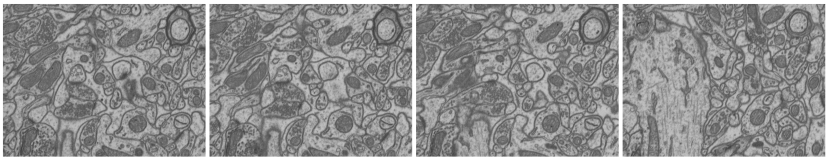
1. Arganda-Carreras, I., Turaga, S.C., Berger, D.R., Cireşan, D., Giusti, A., Gambardella, L.M., Schmidhuber, J., Laptev, D., Dwivedi, S., Buhmann, J.M., et al.: Crowdsourcing the creation of image segmentation algorithms for connectomics. Frontiers in Neuroanatomy **9**, 142 (2015) 4
2. Badrinarayanan, V., Kendall, A., Cipolla, R.: SegNet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(12), 2481–2495 (2017) 4
3. Bello, I., Fedus, W., Du, X., Cubuk, E.D., Srinivas, A., Lin, T.Y., Shlens, J., Zoph, B.: Revisiting ResNets: Improved Training and Scaling Strategies. arXiv preprint arXiv:2103.07579 (2021) 2, 5
4. Buhmann, J., Krause, R., Lentini, R.C., Eckstein, N., Cook, M., Turaga, S., Funke, J.: Synaptic partner prediction from point annotations in insect brains. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 309–316. Springer (2018) 4
5. Casser, V., Kang, K., Pfister, H., Haehn, D.: Fast mitochondria detection for connectomics. In: Medical Imaging with Deep Learning (2020) 4, 5, 8, 10, 11, 12, 13, 18, 20, 21, 35
6. Chaurasia, A., Culurciello, E.: LinkNet: Exploiting encoder representations for efficient semantic segmentation. In: 2017 IEEE Visual Communications and Image Processing (VCIP). pp. 1–4. IEEE (2017) 4
7. Cheng, H.C., Varshney, A.: Volume segmentation using convolutional neural networks with limited training data. In: 2017 IEEE International Conference on Image Processing (ICIP). pp. 590–594. IEEE (2017) 5, 8, 11, 12, 13, 18, 19, 33, 34
8. Çiçek, Ö., Abdulkadir, A., Lienkamp, S.S., Brox, T., Ronneberger, O.: 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 424–432. Springer (2016) 14, 15, 16, 18, 44
9. Dai, J., Li, Y., He, K., Sun, J.: R-FCN: Object Detection via Region-based Fully Convolutional Networks. In: Advances in Neural Information Processing Systems. pp. 379–387 (2016) 16, 18

10. De Moura, M.B., dos Santos, L.S., Van Houten, B.: Mitochondrial dysfunction in neurodegenerative diseases and cancer. Environmental and Molecular Mutagenesis **51**(5), 391–405 (2010) 2
11. Dodge, J., Gururangan, S., Card, D., Schwartz, R., Smith, N.A.: Show Your Work: Improved Reporting of Experimental Results. arXiv preprint arXiv:1909.03004 (2019) 8, 19, 22
12. Fu, H., Cheng, J., Xu, Y., Wong, D.W.K., Liu, J., Cao, X.: Joint Optic Disc and Cup Segmentation Based on Multi-label Deep Network and Polar Transformation. IEEE Transactions on Medical Imaging **37**(7), 1597–1605 (2018) 14, 15, 16, 18, 43
13. Fulda, S., Galluzzi, L., Kroemer, G.: Targeting mitochondria for cancer therapy. Nature reviews Drug discovery **9**(6), 447–464 (2010) 2
14. Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., Martinez-Gonzalez, P., Garcia-Rodriguez, J.: A survey on deep learning techniques for image and video semantic segmentation. Applied Soft Computing **70**, 41–65 (2018) 3
15. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. pp. 249–256 (2010) 19
16. Gu, Z., Cheng, J., Fu, H., Zhou, K., Hao, H., Zhao, Y., Zhang, T., Gao, S., Liu, J.: CE-Net: Context Encoder Network for 2D Medical Image Segmentation. IEEE Transactions on Medical Imaging **38**(10), 2281–2292 (2019) 4
17. Haque, I.R.I., Neubert, J.: Deep learning approaches to biomedical image segmentation. Informatics in Medicine Unlocked **18**, 100297 (2020) 3
18. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2961–2969 (2017) 4
19. He, K., Zhang, X., Ren, S., Sun, J.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1026–1034 (2015) 19
20. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016) 5, 7
21. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision. pp. 630–645. Springer (2016) 5
22. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7132–7141 (2018) 17, 40, 41
23. Ibtehaz, N., Rahman, M.S.: MultiResUNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation. Neural Networks **121**, 74–87 (2020) 4, 14, 15, 16, 18, 42
24. Isensee, F., Jaeger, P.F., Kohl, S.A., Petersen, J., Maier-Hein, K.H.: nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. Nature Methods **18**(2), 203–211 (2021) 2, 5, 14, 15, 16, 18, 19, 45
25. Jégou, S., Drozdzal, M., Vazquez, D., Romero, A., Bengio, Y.: The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition workshops. pp. 11–19 (2017) 4, 14, 15, 16, 18, 39
26. Jin, Q., Meng, Z., Pham, T.D., Chen, Q., Wei, L., Su, R.: DUNet: A deformable network for retinal vessel segmentation. Knowledge-Based Systems **178**, 149–162 (2019) 4
27. Kasthuri, N., Hayworth, K.J., Berger, D.R., Schalek, R.L., Conchello, J.A., Knowles-Barley, S., Lee, D., Vázquez-Reina, A., Kaynig, V., Jones, T.R., et al.: Saturated reconstruction of a volume of neocortex. Cell **162**(3), 648–661 (2015) 10

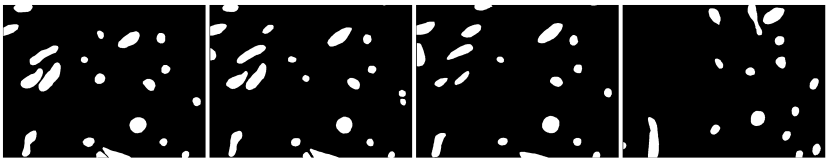
28. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems **25**, 1097–1105 (2012) 4
29. Litjens, G., Kooi, T., Bejnordi, B.E., Setio, A.A.A., Ciompi, F., Ghafoorian, M., Van Der Laak, J.A., Van Ginneken, B., Sánchez, C.I.: A survey on deep learning in medical image analysis. Medical Image Analysis **42**, 60–88 (2017) 3
30. Liu, J., Li, W., Xiao, C., Hong, B., Xie, Q., Han, H.: Automatic detection and segmentation of mitochondria from sem images using deep neural network. In: 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). pp. 628–631. IEEE (2018) 4
31. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3431–3440 (2015) 4, 14, 15, 38
32. Lucchi, A., Becker, C., Neila, P.M., Fua, P.: Exploiting enclosing membranes and contextual cues for mitochondria segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 65–72. Springer (2014) 4
33. Lucchi, A., Li, Y., Fua, P.: Learning for structured prediction using approximate subgradient descent with working sets. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1987–1994 (2013) 4
34. Lucchi, A., Li, Y., Smith, K., Fua, P.: Structured image segmentation using kernelized features. In: European Conference on Computer Vision. pp. 400–413. Springer (2012) 4
35. Lucchi, A., Márquez-Neila, P., Becker, C., Li, Y., Smith, K., Knott, G., Fua, P.: Learning Structured Models for Segmentation of 2-D and 3-D Imagery. IEEE Transactions on Medical Imaging **34**(5), 1096–1110 (2014) 4
36. Lucchi, A., Smith, K., Achanta, R., Knott, G., Fua, P.: Supervoxel-based segmentation of mitochondria in em image stacks with learned shape features. IEEE Transactions on Medical Imaging **31**(2), 474–486 (2011) 2, 4, 9
37. Meijering, E.: A bird's-eye view of deep learning in bioimage analysis. Computational and Structural Biotechnology Journal **18**, 2312 (2020) 2
38. Meyer, F.: Topographic distance and watershed lines. Signal Processing **38**(1), 113–125 (1994) 4
39. Milletari, F., Navab, N., Ahmadi, S.A.: V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: 2016 fourth International Conference on 3D Vision (3DV). pp. 565–571. IEEE (2016) 4
40. Minaee, S., Boykov, Y.Y., Porikli, F., Plaza, A.J., Kehtarnavaz, N., Terzopoulos, D.: Image segmentation using deep learning: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021) 3
41. Moen, E., Bannon, D., Kudo, T., Graf, W., Covert, M., Van Valen, D.: Deep learning for cellular image analysis. Nature methods pp. 1–14 (2019) 2
42. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1520–1528 (2015) 4
43. Oztel, I., Yolcu, G., Ersoy, I., White, T., Bunyak, F.: Mitochondria segmentation in electron microscopy volumes using deep convolutional neural network. In: 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). pp. 1195–1200. IEEE (2017) 4, 8, 11, 13, 14, 18, 37
44. Poole, A.C., Thomas, R.E., Andrews, L.A., McBride, H.M., Whitworth, A.J., Pallanck, L.J.: The pink1/parkin pathway regulates mitochondrial morphology. Proceedings of the National Academy of Sciences **105**(5), 1638–1643 (2008) 2

45. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 234–241. Springer (2015) 4, 6
46. Roy, A.G., Navab, N., Wachinger, C.: Concurrent Spatial and Channel Squeeze & Excitation in Fully Convolutional Networks. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 421–429. Springer (2018) 4
47. Roy, A.G., Navab, N., Wachinger, C.: Recalibrating fully convolutional networks with spatial and channel "squeeze and excitation" blocks. IEEE Transactions on Medical Imaging **38**(2), 540–549 (2018) 17
48. Schlemper, J., Oktay, O., Schaap, M., Heinrich, M., Kainz, B., Glocker, B., Rueckert, D.: Attention gated networks: Learning to leverage salient regions in medical images. Medical Image Analysis **53**, 197–207 (2019) 4, 5, 7, 19, 20
49. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014) 4
50. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–9 (2015) 4
51. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2818–2826 (2016) 5
52. Tait, S.W., Green, D.R.: Mitochondria and cell signalling. Journal of Cell Science **125**(4), 807–815 (2012) 2
53. Wallace, D.C.: Mitochondria and cancer. Nature Reviews Cancer **12**(10), 685–698 (2012) 2
54. Wei, D., Lin, Z., Franco-Barranco, D., Wendt, N., Liu, X., Yin, W., Huang, X., Gupta, A., Jang, W.D., Wang, X., et al.: MitoEM Dataset: Large-Scale 3D Mitochondria Instance Segmentation from EM Images. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 66–76. Springer (2020) 22
55. Xiao, C., Chen, X., Li, W., Li, L., Wang, L., Xie, Q., Han, H.: Automatic mitochondria segmentation for EM data using a 3D supervised convolutional network. Frontiers in Neuroanatomy **12**, 92 (2018) 5, 8, 11, 13, 14, 18, 36
56. Zhou, Z., Siddiquee, M.M.R., Tajbakhsh, N., Liang, J.: Unet++: A nested u-Net architecture for medical image segmentation. In: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, pp. 3–11. Springer (2018) 4, 14, 16, 18, 46
57. Zhuang, J.: LadderNet: Multi-path networks based on U-Net for medical image segmentation. arXiv preprint arXiv:1810.07810 (2018) 4

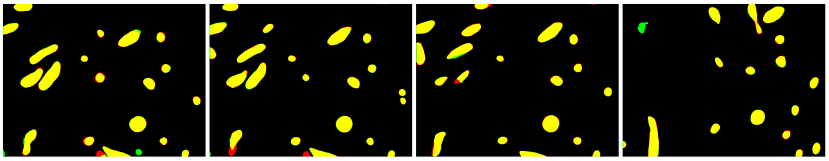
A Appendix: Network predictions on Lucchi



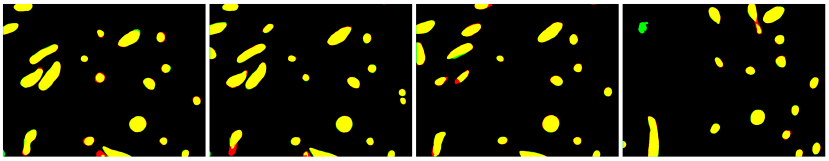
(a) Original images



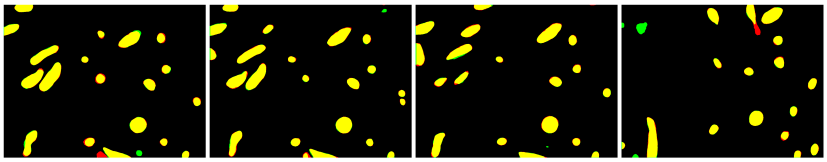
(b) Ground Truth



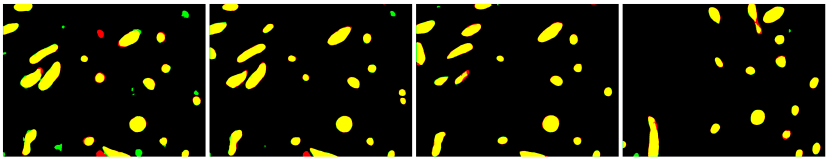
(c) Basic 2D U-Net predictions



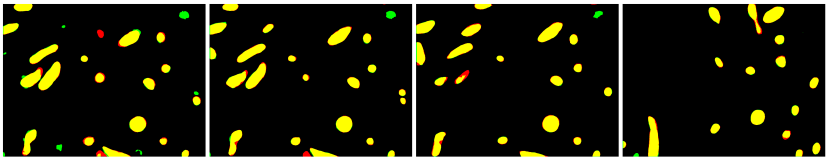
(d) 2D Attention U-Net predictions



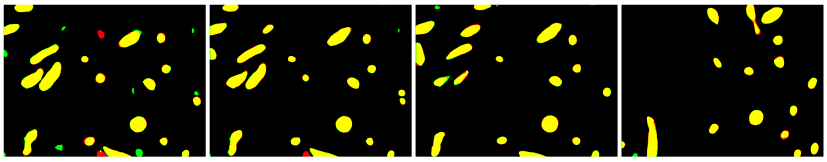
(e) 2D Residual U-Net predictions



(f) Basic 3D U-Net predictions



(g) 3D Attention U-Net predictions



(h) 3D Residual U-Net predictions

Fig. A.1: Predictions of the configuration that measures the best IoU score for each proposed network on Lucchi dataset. Yellow: True Positive; Green: False Positive; Red: False Negative.

B Appendix: Hyperparameter search space for each configuration

B.1 Notations

- $[a, b]$: Range between two possible values. E.g. *zoom([0.75,1.25])* correspond to random zoom value between 0.75 and 1.25.
- $[a, b, c]$: All values from a to b with c step. E.g. $[10, 300, 10]$ correspond to $10, 20, 30, 40, ..., 300$.
- $choice[a, b, ...]$: one value between a, b and so on. E.g. *[10, 15, 20, 30, 60]* possible values are: 10 or 15 or 20 or 30 or 60 (but only one).
- $a, b, c, ...$: all tested values. E.g. flips, rotations, etc.

B.2 Training setup

Computing Infrastructure	GeForce GTX 1080 Ti
GPU libraries	CUDA (10.1) + cuDNN (7.6.5)
Operating System	Ubuntu 16.04.6 LTS
Number of runs	10
Implementation	Tensorflow (2.1.0) + Keras (2.2.4-tf)
Code	Github
Math seed used	42
Environment used	Anaconda → DL_EM_base_env

B.3 2D U-Net

Template to reproduce the results: U-Net_2D_template.py

Hyperparameter	Search space	Best assignment
Duplicate train	2	-
Validation	True	True
Random validation	True	True
% of train as validation	10%	10%
Patches	*choice*[random during train, created from train data before network training]	created from train data before network training
Patch size	256×256	256×256
Discard patches with less than a % of the foreground class	*choice*[True(5%), True(10%), True(15%), True(20%), True(30%), False]	False
Shuffle train on each epoch	*choice*[True,False]	True
Probability map	*choice*[False,True]	False
Probability for each class	*choice*[Foreground: 0.94 ; Background: 0.06, Foreground: 0.9 ; Background: 0.1]	-
Data augmentation	flips, rotation_range([-180,180]), square_rotations([0,90,180,270]), shearing([0.1,0.3]), shift([0.1,0.3]), brightness_range([0.8,1.2]), median_filtering(*choice*[1,3,5]), elastic, zoom([0.75,1.25])	flips, rotation_range([-180,180])
Number of epochs	[10,600,10]	360
Patience	choice [30,50,100]	50
Batch size	*choice*[3,5,6,9] and [1,64,×2]	6
Loss type	*choice*[BCE,Dice,Jaccard]	BCE
Optimizer	*choice*[SGD,Adam,Adabound]	SGD
SGD learning rate	*choice*[0.0001,0.0005,0.001, 0.002,0.005,0.01,0.05,0.1]	0.002
Adam learning rate	0.0001	-
Adabound learning rate	*choice*[(lr=0.0005,final_lr=0.1), (lr=0.0001,final_lr=0.1), (lr=0.001,final_lr=0.1), (lr=0.003,final_lr=0.1)]	-
Number of feature maps to start with (x2 and /2 of each down and up levels respec.)	*choice*[16,32,64]	16
Dropout type	*choice*[dropout,spatial dropout]	dropout
Dropout	[0,0.4,0.1] and tiered on each downsampling: 0.1,0.2,0.3	tiered: 0.1,0.2,0.3
Pooling type	*choice*[Max-pooling,Average-pooling]	Max-pooling
Kernel initializer	*choice*[glorot_uniform,he_init]	he_init
Activation	*choice*[ReLU,ELU]	ELU

Table B.1: Hyperparameter search space for the proposed 2D U-Net.

B.4 2D Residual U-Net

Template to reproduce the results: Residual_U-Net_2D_template.py

Hyperparameter	**Search space**	**Best assignment**
Validation	True	True
Random validation	True	True
% of train as validation	10%	10%
Patches	created from train data before network training	created from train data before network training
Patch size	256×256	256×256
Data augmentation	flips, rotation_range([-180,180])	flips, rotation_range([-180,180])
Number of epochs	360	360
Patience	50	50
Batch size	*choice*[2,4,6,8]	6
Loss type	BCE	BCE
Optimizer	SGD	SGD
SGD learning rate	*choice*[0.001,0.002,0.003,0.004, 0.0001,0.0005,0.0007,0.0009]	0.002
Number of feature maps to start with (x2 and /2 of each down and up levels respec.)	*choice*[16,32]	16

Table B.2: Hyperparameter search space for the proposed 2D Residual U-Net.

B.5 3D U-Net

Template to reproduce the results: U-Net_3D_template.py

Hyperparameter	Search space	Best assignment
Validation	True	True
Random validation	*choice*[False, True]	False
% of train as validation	10%	10%
Subvolumes	created from train data before network training	created from train data before network training
Subvolume shape	*choice*[$80 \times 80 \times 80$, $128 \times 128 \times 48$, $128 \times 128 \times 80$, $128 \times 128 \times 128$, $112 \times 112 \times 112$]	$80 \times 80 \times 80$
Data augmentation	flips, square_rotations([0,90,180,270]), elastic, histogram_equalization, gaussian_blur(*choice*[$\sigma = (0, 2)$, $\sigma = (1, 2)$]), gamma_contrast(*choice*[$\sigma = (0.5, 2)$, $\sigma = (1.25, 1.75)$])	flips, elastic, square_rotations([0,90,180,270])
Number of epochs	360	360
Patience	200	200
Batch size	*choice*[1,2,4,6,8]	1
Loss type	BCE	BCE
Optimizer	*choice*[SGD,Adam]	Adam
Adam learning rate	*choice*[0.00001,0.00005,0.0001,0.0002, 0.0005,0.0007,0.001,0.005,0.01]	0.0001
SGD learning rate	*choice*[0.001]	-
Number of feature maps to start with (x2 and /2 of each down and up levels respec.)	*choice*[16,20,24,32]	-
Manually feature maps (on each level until bottleneck)	*choice*[{28,36,48,64,80,96}, {28,36,48,64,80}, {28,36,48,64}]	{28,36,48,64}
Dropout	*choice*[0,0.1,0.2]	-
Dropout type	*choice*[dropout,spatial_dropout]	-
Batch normalization	*choice*[False,True]	False
Network depth	*choice*[3,4,5]	3

Table B.3: Hyperparameter search space for proposed 3D U-Net.

B.6 3D Residual U-Net

Template to reproduce the results: Residual_U-Net_3D_template.py

Hyperparameter	Search space	Best assignment
Validation	True	True
Random validation	False	False
% of train as validation	10%	10%
Subvolumes	created from train data before network training	created from train data before network training
Subvolume shape	$80 \times 80 \times 80$	$80 \times 80 \times 80$
Data augmentation	flips, elastic, square_rotations([0,90,180,270])	flips, elastic, square_rotations([0,90,180,270])
Number of epochs	360	360
Patience	200	200
Batch size	*choice*[1,2,4,6]	1
Loss type	BCE	BCE
Optimizer	Adam	Adam
Adam learning rate	0.0001	0.0001
Number of feature maps to start with (x2 and /2 of each down and up levels respec.)	*choice*[16,32]	-
Manually feature maps (on each level until bottleneck)	*choice*[{28,36,48,64,80,96}, {28,36,48,64,80}, {28,36,48,64}]	{28,36,48,64,80}
Dropout	*choice*[0,0.1]	-
Dropout type	*choice*[dropout,spatial_dropout]	-
Batch normalization	*choice*[False,True]	False
Network depth	*choice*[3,4,5]	4

Table B.4: Hyperparameter search space for proposed 3D Residual U-Net.

B.7 Cheng 2D [7]

Templates to reproduce the results:
***Original*:** cheng_2D_template_V0.py
***Modified*:** cheng_2D_template_V1.py

Hyperparameter	**Search space**	**Best assignment**
Duplicate train	*choice*[1,12]	12
Validation	True	True
Random validation	True	True
% of train as validation	10%	10%
Patches	random selection from the whole data during train	random selection from the whole data during train
Patch shape	256×256	256×256
Probability map	*choice*[False,True]	True
Probability for each class	Foreground: 0.94 ; Background: 0.06	Foreground: 0.94 ; Background: 0.06
Data augmentation	flips, rotation_range([-180,180])	flips, rotation_range([-180,180])
Number of epochs	*choice*[4000,400]	400
Patience	200	200
Batch size	24	24
Loss type	BCE	BCE
Optimizer	*choice*[SGD,Adam]	Adam
SGD learning rate	*choice*[0.002,0.05]	-
Adam learning rate	*choice*[0.0001]	0.0001
learning rate scheduler	*choice*[True,False]	False
Dropout	*choice*[0,0.1]	0

Table B.5: Hyperparameter search space for 2D network proposed by Cheng *et al.* [7].

B.8 Cheng 3D [7]

Templates to reproduce the results:
***Original*:** cheng_3D_template_V0.py
***Modified*:** cheng_3D_template_V1.py

Hyperparameter	**Search space**	**Best assignment**
Duplicate train	*choice*[1,12]	12
Validation	False	False
Subvolumes	*choice*[created from train data before network training, random selection from the whole data during training]	created from training data before network training
Subvolume shape	$128 \times 128 \times 96$	$128 \times 128 \times 96$
Probability map	*choice*[False,True]	False
Data augmentation	flips, square_rotations([0,90,180,270]), elastic	square_rotations([0,90,180,270])
Number of epochs	*choice*[545,150]	150
Patience	*choice*[50,200]	50
Batch size	*choice*[1,3]	3
Loss type	BCE	BCE
Optimizer	*choice*[SGD,Adam]	Adam
SGD learning rate	0.1	-
Adam learning rate	0.0001	0.0001
learning rate scheduler	*choice*[True,False]	False
Dropout	0.1	0.1

Table B.6: Hyperparameter search space for 3D network proposed by Cheng *et al.* [7].

B.9 Casser [5]

Templates to reproduce the results:
***Original*:** casser_template_V0.py
***Modified*:** casser_template_V1.py

Hyperparameter	Search space	Best assignment
Duplicate train	*choice*[1,2,12]	2
Validation	True	True
Random validation	*choice*[True,False]	False
% of train as validation	*choice*[5%,10%,20%,30%]	10%
Patches	random selection from the whole data during train	random selection from the whole data during train
Patch size	512×512	512×512
Probability map	*choice*[False,True]	True
Probability for each class	Foreground: 0.9 ; Background: 0.1, Foreground: 0.94 ; Background: 0.06,	Foreground: 0.94 ; Background: 0.06
Data augmentation	flips, square_rotations([0,90,180,270]), rotation_range([0,180]), shift([0.1,0.3]), shearing([0.1,0.3]), brightness_range([0.8,1.2]), median_filtering(size=5)	flips, rotation_range([0,180])
Number of epochs	360	360
Patience	*choice*[50,200]	200
Batch size	*choice*[4,6]	4
Loss type	BCE	BCE
Optimizer	*choice*[SGD, Adam]	Adam
SGD learning rate	*choice*[0.001,0.002,0.005,0.008,0.01]	-
Adam learning rate	*choice*[0.0005,0.0001,0.001]	0.0005
Dropout	0.2	0.2

Table B.7: Hyperparameter search space for network proposed by Casser *et al.* [5].

B.10 Xiao [55]

Templates to reproduce the results:
***Original*:** xiao_template_V0.py
***Modified*:** xiao_template_V1.py

Hyperparameter	**Search space**	**Best assignment**
Duplicate train	*choice*[70,75,80,85]	75
Validation	True	False
Random validation	*choice*[True,False]	True
% of train as validation	*choice*[10%,20%]	10%
Subvolumes	created from train data before network training	created from train data before network training
Subvolume shape	$256 \times 256 \times 20$ (train) $448 \times 576 \times 20$ (test)	$256 \times 256 \times 20$ (train) $448 \times 576 \times 20$ (test)
Data augmentation	flips, square_rotations([0,90,180,270]), elastic	flips, square_rotations([0,90,180,270]), elastic
Number of epochs	30	30
Patience	30	30
Batch size	2	2
Loss type	BCE	BCE
Optimizer	Adam	Adam
Adam learning rate	0.0001	0.0001
Last network layer	*choice*[sigmoid,softmax]	softmax
L2 normalization	*choice*[0.1,0.01,0.001]	0.01

Table B.8: Hyperparameter search space for network proposed by Xiao *et al.* [55].

B.11 Oztel [43]

Templates to reproduce the results:
***Original*:** oztel_template_V0.py
***Modified*:** oztel_template_V1.py

Hyperparameter	Search space	Best assignment
Duplicate mitochondria samples	*choice*[2,3,6]	2
Reduce background samples	preserve 78% of samples	-
Validation	True	True
Random validation	True	True
% of train as validation	*choice*[10%,20%]	20%
Patches	created from train data before network training	created from train data before network training
Patch size	32×32	32×32
Data augmentation	flips, rotation_range([-180,180])	flips, rotation_range([-180,180])
Number of epochs	360	360
Patience	360	360
Batch size	32	32
Loss type	*choice*[CCE,BCE]	CCE
Optimizer	Adam	Adam
Adam learning rate	0.0001	0.0001

Table B.9: Hyperparameter search space for network proposed by Oztel *et al.* [43].

B.12 FCN [31]

Templates to reproduce the results:
FCN32: FCN32_template.py
FCN8: FCN8_template.py

Hyperparameter	**Search space**	**Best assignment**
Validation	True	True
Random validation	True	True
% of train as validation	10%	10%
Patches	created from train data before network training	created from train data before network training
Patch size	256×256	256×256
Data augmentation	flips, rotation_range([-180,180])	flips, rotation_range([-180,180])
Number of epochs	360	360
Patience	200	200
Batch size	6	6
Loss type	BCE	BCE
Optimizer	*choice*[SGD,Adam]	Adam
SGD learning rate	0.002	-
Adam learning rate	0.0001	0.0001

Table B.10: Hyperparameter search space for FCN32 anf FCN8 networks [31].

B.13 Tiramisu [25]

Template to reproduce the results: Tiramisu_template.py

Hyperparameter	Search space	Best assignment
Validation	True	True
Random validation	True	True
% of train as validation	10%	10%
Patches	created from train data before network training	created from train data before network training
Patch size	256×256	256×256
Data augmentation	flips, rotation_range([-180,180])	flips, rotation_range([-180,180])
Number of epochs	360	360
Patience	200	200
Batch size	*choice*[1,2,4,6,8,16]	4
Loss type	BCE	BCE
Optimizer	*choice*[SGD,Adam]	Adam
SGD learning rate	*choice*[0.01,0.005,0.001, 0.0005,0.0001,0.002,0.003]	-
Adam learning rate	*choice*[0.005,0.001, 0.0005,0.0001,0.00005]	0.0001

Table B.11: Hyperparameter search space for Tiramisu [25].

B.14 2D SE U-Net 2D

Template to reproduce the results: SE_U-Net_2D_template.py

Hyperparameter	**Search space**	**Best assignment**
Validation	True	True
Random validation	True	True
% of train as validation	10%	10%
Patches	created from train data before network training	created from train data before network training
Patch size	256×256	256×256
Data augmentation	flips, rotation_range([-180,180])	flips, rotation_range([-180,180])
Number of epochs	360	360
Patience	200	200
Batch size	6	6
Loss type	BCE	BCE
Optimizer	*choice*[SGD,Adam]	SGD
SGD learning rate	0.002	0.002
Adam learning rate	0.0001	-
SE blocks position	*choice*[after each conv, after each conv (but not in bottleneck)]	after each conv (but not in bottleneck)

Table B.12: Hyperparameter search space for 2D U-Net (adding SE blocks [22]).

B.15 3D SE U-Net

Template to reproduce the results: SE_U-Net_3D_template.py

Hyperparameter	Search space	Best assignment
Validation	True	True
Random validation	True	True
% of train as validation	10%	10%
Subvolumes	created from train data before network training	created from train data before network training
Subvolume size	$80 \times 80 \times 80$	$80 \times 80 \times 80$
Data augmentation	flips, square_rotations([0,90,180,270]), elastic	flips, square_rotations([0,90,180,270]), elastic
Number of epochs	360	360
Patience	200	200
Batch size	1	1
Loss type	BCE	BCE
Optimizer	Adam	Adam
Adam learning rate	0.0001	0.0001
SE blocks position	*choice*[after each conv, after each conv (but not in bottleneck)]	after each conv (but not in bottleneck)

Table B.13: Hyperparameter search space for 3D U-Net (adding SE blocks [22]).

B.16 MultiResUNet [23]

Template to reproduce the results: MultiResUNet_template.py

Hyperparameter	**Search space**	**Best assignment**
Validation	True	True
Random validation	True	True
% of train as validation	10%	10%
Patches	created from train data before network training	created from train data before network training
Patch size	256×256	256×256
Data augmentation	flips, rotation_range([-180,180])	flips, rotation_range([-180,180])
Number of epochs	360	360
Patience	200	200
Batch size	*choice*[1,2,4,6,8,16]	6
Loss type	BCE	BCE
Optimizer	*choice*[SGD,Adam]	Adam
SGD learning rate	*choice*[0.01,0.005,0.001, 0.0005,0.0001,0.002,0.003]	-
Adam learning rate	*choice*[0.005,0.001, 0.0005,0.0001,0.00005]	0.005

Table B.14: Hyperparameter search space for MultiResUNet [23].

B.17 MNet [12]

Template to reproduce the results: MNet_template.py

Hyperparameter	Search space	Best assignment
Validation	True	True
Random validation	True	True
% of train as validation	10%	10%
Patches	created from train data before network training	created from train data before network training
Patch size	256×256	256×256
Data augmentation	flips, rotation_range([-180,180])	flips, rotation_range([-180,180])
Number of epochs	360	360
Patience	200	200
Batch size	*choice*[1,2,4,6,8,16]	6
Loss type	BCE	BCE
Optimizer	*choice*[SGD,Adam]	SGD
SGD learning rate	*choice*[0.01,0.005,0.001, 0.0005,0.0001,0.002,0.003]	0.01
Adam learning rate	*choice*[0.005,0.001, 0.0005,0.0001,0.00005]	-

Table B.15: Hyperparameter search space for MNet [12].

B.18 3D Vanilla U-Net [8]

Template to reproduce the results: Vanilla_U-Net_3D_template.py

Hyperparameter	**Search space**	**Best assignment**
Validation	True	True
Random validation	True	True
% of train as validation	10%	10%
Subvolumes	created from train data before network training	created from train data before network training
Subvolume size	$80 \times 80 \times 80$	$80 \times 80 \times 80$
Data augmentation	flips, square_rotations([0,90,180,270]), elastic	flips, square_rotations([0,90,180,270]), elastic
Number of epochs	360	360
Patience	200	200
Batch size	1	1
Loss type	BCE	BCE
Optimizer	Adam	SGD
Adam learning rate	0.0001	0.0001

Table B.16: Hyperparameter search space for 3D Vanilla U-Net [8].

B.19 nnU-Net [24]

Template to reproduce the results: nnU-Net_template.py

Hyperparameter	Search space	Best assignment
Validation	True	True
Random validation	True	True
% of train as validation	10%	10%
Patches	created from train data before network training	created from train data before network training
Patch size	256×256	256×256
Network's final layer activation	*choice*[with sigmoid, with any, with softmax]	without softmax
Data augmentation	flips, rotation_range([-180,180])	flips, rotation_range([-180,180])
Number of epochs	360	360
Patience	200	200
Batch size	*choice*[1,2,4,6,8,16]	4
Loss type	*choice*[BCE,BCE+Dice]	BCE+Dice
Optimizer	*choice*[SGD,Adam]	Adam
SGD learning rate	*choice*[0.05,0.001,0.002]	-
Adam learning rate	*choice*[0.005,0.0005, 0.0001,0.00005]	0.0001

Table B.17: Hyperparameter search space for nnU-Net [24].

B.20 U-Net++ [56]

Template to reproduce the results: U-Net++_template.py

Hyperparameter	Search space	Best assignment
Validation	True	True
Random validation	True	True
% of train as validation	10%	10%
Patches	created from train data before network training	created from train data before network training
Patch size	256×256	256×256
Backbone	ResNet50	ResNet50
Data augmentation	flips, rotation_range([-180,180])	flips, rotation_range([-180,180])
Number of epochs	360	360
Patience	200	200
Batch size	*choice*[1,2,4,6,8,16]	1
Loss type	BCE	BCE
Optimizer	*choice*[SGD,Adam]	SGD
SGD learning rate	*choice*[0.0005,0.001, 0.002,0.005,0.01]	0.01
Adam learning rate	*choice*[0.0001, 0.0005,0.001]	-

Table B.18: Hyperparameter search space for U-Net++ [56]