SLAS Technology 43 (2026) 100467

Contents lists available at ScienceDirect

SLAS Technology

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

Full Length Article

Bayesian optimization for design parameters of 3D image data analysis

David Exler [a], Joaquin Eduardo Urrutia Gómez [a], Martin Krüger [a], Maike Schliephake [b], John Jbeily [c], Mario Vitacolonna [c], Rüdiger Rudolf [c], Markus Reischl [a,*]

[a] Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1, 76344, Eggenstein-Leopoldshafen, Germany
[b] Institute of Biological and Chemical Systems, Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1, 76344, Eggenstein-Leopoldshafen, Germany
[c] CeMOS Research and Transfer Center, Technische Hochschule Mannheim, Paul-Wittsack-Straße 10, 68163, Mannheim, Germany

ARTICLE INFO

Keywords:
Bayesian optimization
Classification
Segmentation
3D imaging
Cell tracking challenge

ABSTRACT

Deep learning-based segmentation and classification are crucial to large-scale biomedical imaging, particularly for 3D data, where manual analysis is impractical. Although many methods exist, selecting suitable models and tuning parameters remains a major bottleneck in practice, as manually identifying and comparing them requires significant computational and labor effort. Hence, we introduce the 3D data Analysis Optimization Pipeline, a method designed to facilitate the design and parameterization of segmentation and classification using two Bayesian Optimization stages. First, the pipeline selects a segmentation model and optimizes post-processing parameters using a domain-adapted synthetic benchmark dataset. To ensure a concise evaluation of segmentation performance, we introduce a segmentation quality metric that serves as the objective function. Second, the pipeline optimizes design choices of a classifier, such as encoder and classifier head architectures, incorporation of prior knowledge, and pretraining strategies. To reduce manual annotation effort, this stage includes an assisted class-annotation workflow that extracts predicted instances from the segmentation results and sequentially presents them to the operator, eliminating the need for manual tracking. In four case studies, the 3D data Analysis Optimization Pipeline efficiently identifies effective model and parameter configurations for individual datasets.

1. Introduction

Microscopy is a fundamental tool to observe, quantify, and interpret cell and tissue structures. Recently, 3D data analysis has gained increasing relevance in cellular research because it captures spatial context lost in 2D projections [1,2]. Modern devices can acquire hundreds to thousands of 3D images per sample, generating datasets that are nearly impossible to analyze manually. Segmentation and classification are, therefore, key steps in automatically extracting interpretable information [3].

Numerous methods have been proposed for segmentation and classification of 3D biomedical data [4–6]. Most segmentation approaches are based on the U-Net architecture [7], while classification models typically combine a Convolutional Neural Networks (CNN) or Vision Transformer encoder with a classifier head [8,9]. Adaptations of U-Net for 3D data follow a similar pattern [10,11]. Automated frameworks such as nnU-Net reduce manual design effort by adapting a U-Net pipeline to dataset characteristics, but are limited by the available training data and computational resources [12]. Several pretrained segmentation models for 3D biomedical images are available and highly generalizable, so they can be adapted to new datasets either by leveraging learned spatial embeddings that are applicable to many domains [13] or by pretraining on a large corpus of variable annotated data [14–18]. Postprocessing techniques such as morphological operations, watershed-based separation, or graph-based merging can further improve segmentation quality [19–23]. However, the performance of pretrained models and existing models varies strongly for each individual dataset.

For classification, encoder backbones pretrained on large-scale datasets like ImageNet are available [24–28], but adapting them typically requires additional training. Semi-supervised annotation methods, such as label spreading, can enhance datasets with limited annotations [29]. Classifier heads must be tailored to the downstream task, often incorporating explicit 3D operations such as 3D CNNs or self-attention mechanisms [30,31]. Classification performance, especially, is very dependent on the architecture, pipeline design, and the components' ability to generalize to the domain of a specific application [32].

We refer to the entirety of the design choices regarding the model architecture, the training, and the analysis pipeline as design parameters. We distinguish between design parameters and hyperparameters,

* Corresponding author.
E-mail addresses: david.exler@kit.edu (D. Exler), markus.reischl@kit.edu (M. Reischl).

https://doi.org/10.1016/j.slast.2026.100467
Received 18 February 2026; Received in revised form 24 August 2026; Accepted 25 September 2026
Available online 9 October 2026
2472-6303/© 2026 The Authors. Published by Elsevier Inc. on behalf of Society for Laboratory Automation and Screening. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).
Open access funding enabled and organized by Projekt DEAL.

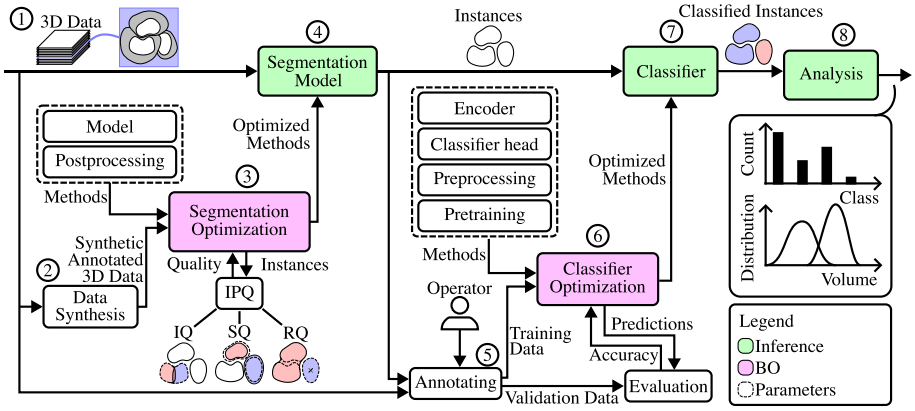


Fig. 1. Progression of the introduced pipeline. 3D data (1) is used for data synthesis (2), combining instance simulation and domain adaptation. The synthetic data serves as input for segmentation optimization (3), where a pretrained model and postprocessing parameters are optimized with respect to the IPQ metric. The optimized segmentation model (4) predicts instances, which are annotated by an operator (5). Classifier optimization (6) selects encoder and classifier head architectures, preprocessing, and pretraining strategies based on validation accuracy. The optimized classifier (7) predicts class labels, enabling the final analysis (8), which yields statistics such as class counts and instance volume distributions.

which control the training or inference of an individual design variation, for example via the learning rate, dropout rate, or normalization parameters. Because segmentation and classification performance of established methods vary heavily across datasets, dataset-specific adaptation of design parameters is imperative to achieve optimal performance. Current research on design parameters mainly focuses on new methods, new model architectures, more efficient training strategies, more effective transfer learning approaches, or strategies for increasing generalizability. Optimizing the design parameters, however, is challenging due to the large and expanding corpus of available methods and the limited availability of annotated 3D data [33].

To mitigate the limitation of data availability, synthetic 3D data generation and domain adaptation methods have been proposed, reducing the need for labor-intensive manual annotation [34–37]. Nevertheless, optimizing design parameters of analysis pipelines for specific 3D tasks remains difficult. Naive parameter tuning approaches such as grid or random search are computationally inefficient, whereas Bayesian Optimization (BO) enables efficient optimization of expensive black-box functions by modeling uncertainty with probabilistic surrogate models [38,39].

We therefore introduce the 3D data Analysis Optimization Pipeline (3D-AOP), a framework for efficient automated optimization of 3D image analysis pipelines. Our objective is to mitigate the manual effort of design parameter choices by proposing a pipeline to optimize the parameters in a repeatable and efficient setting. The 3D-AOP guides researchers from raw data to analysis by identifying dataset characteristics relevant to deep learning performance and optimizing segmentation and classification using two BO processes. The first process compares segmentation models and optimizes postprocessing parameters on a domain-adapted synthetic benchmark dataset, supported by a newly introduced quality criterion that distinguishes different segmentation error types. Using the optimized segmentation to facilitate 3D instance annotation, the second BO process optimizes classifier design parameters, including pretraining strategies, encoder and classifier head architectures, and the integration of prior knowledge. We demonstrate the applicability of the 3D-AOP on four cellular 3D microscopy datasets. First, we analyze nuclei in *in vitro* Myotube cultures [40]. Second, we apply the 3D-AOP to an additional cellular core–shell dataset. Finally, we perform segmentation optimization on two datasets from the Cell Tracking Challenge (CTC) [41]. Across datasets with differing optical properties, the results show that optimal design choices and parameters are highly data-dependent. The main contributions of this work are:

- an experimentally validated workflow for automated optimization of 3D image analysis pipelines,
- a Bayesian Optimization process to adapt conceptual parameters of instance segmentation inference to a specific image data domain without computationally expensive model retraining,
- a new metric to detect interpretable segmentation errors, and
- a Bayesian Optimization process to fine-tune design parameters of a classifier.

2. Methods

2.1. Overview

To mitigate the effort of manual 3D image data analysis pipeline design, we introduce the 3D-AOP, a framework that optimizes the design of automated 3D data analysis pipelines using two BO processes.

Fig. 1 illustrates the workflow of the 3D-AOP. Due to the limited availability of annotated 3D datasets, the pipeline begins with *data synthesis* (2), generating a segmentation benchmark from the input *3D data* (1). A *segmentation optimization* (3) then selects a pretrained segmentation model and optimizes postprocessing parameters using the newly introduced IPQ metric. The resulting *segmentation model* (4) predicts instances on the input data. To reduce the effort of annotating 3D data for classification, the predicted instances are integrated into an assisted annotation workflow, enabling efficient *annotation* (5). Subsequently, a *classifier optimization* (6) identifies optimal classifier design parameters, including encoder and classifier head architectures, preprocessing, and pretraining strategies, using validation accuracy as the objective. The optimized *classifier* (7) produces class predictions for each instance, enabling the final *analysis* (8). In contrast to recent research that optimizes training hyperparameters, the 3D-AOP focuses on optimizing conceptual design parameters [42,43].

2.2. Segmentation optimization

2.2.1. Objective: Injective panoptic quality

Segmentation performance is commonly evaluated using Panoptic Quality (PQ) [44]. However, PQ fails to penalize the splitting of a correctly detected instance if each fragment exceeds the Intersection over Union (IoU) threshold to be considered a True Positive (TP). Such errors bias downstream measurements and must therefore be considered during optimization. We introduce Injective Panoptic Quality (IPQ), an extension of PQ, as the objective function for segmentation optimization. IPQ penalizes instance splitting and is composed of three

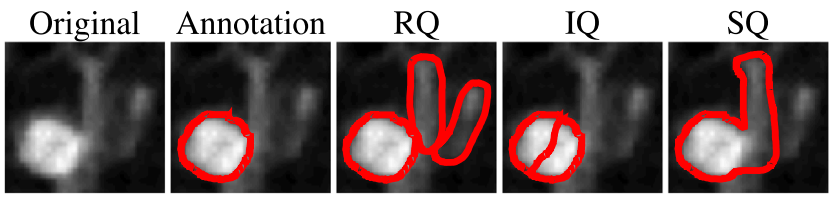


Fig. 2. Examples of idealized error types quantified by the IPQ metric. RQ detects hallucinations, IQ captures instance splitting, and SQ reflects oversegmentation.

factors, each capturing a distinct error type affecting interpretability. The Segmentation Quality (SQ) $\in [0,1]$ factor evaluates over- and undersegmentation and is defined as

$$\mathrm{SQ} = \frac{\sum_{(p,g)\in TP} \mathrm{IoU}\left(\bigcup_{p_i \in p} p_i,\, g\right)}{|TP|}, \tag{1}$$

where TP is the set of TP tuples (p_i, g), with p being a vector of predicted instances p_i and the associated annotation g.

The Recognition Quality (RQ) $\in [0,1]$ factor penalizes hallucinations and omissions:

$$\mathrm{RQ} = \frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}. \tag{2}$$

Because the Injective Quality (IQ) $\in [0,1]$ factor aims to penalize instance splitting, it requires a mapping of predictions to annotations. We count the number of predictions associated with an individual annotation instance with a simple greedy algorithm. To be assigned to an annotation 75% of the prediction must lie within the annotation, or their IoU must be larger than 0.3.

Algorithm 1 Greedy prediction-to-annotation mapping

Require: Predictions P, annotations G, IoU threshold $\tau_{\mathrm{IoU}} = 0.3$, intersection threshold $\tau_{\mathrm{int}} = 0.75$
Initialize $n_g \leftarrow 0$ for all $g \in G$
for all $p \in P$ **do**
  $g^* \leftarrow \arg\max_{g\in G} |p \cap g|$
  if $\mathrm{IoU}(p, g^*) > \tau_{\mathrm{IoU}}$ **or** $|p \cap g^*| > \tau_{\mathrm{int}}|p|$ **then**
    $n_{g^*} \leftarrow n_{g^*} + 1$
  end if
end for

The IQ factor is defined as

$$\mathrm{IQ} = \frac{|G|}{\sum_{p\in P} \max(1, n_p - 1)}, \tag{3}$$

where $|G|$ is the number of annotations and n_p is the number of predicted instances mapped to an annotation, as described in Algorithm 1. Finally, the combined metric is defined as

$$\mathrm{IPQ} = \left[1 - k_1(1 - \mathrm{SQ})\right]\left[1 - k_2(1 - \mathrm{RQ})\right]\left[1 - k_3(1 - \mathrm{IQ})\right]. \tag{4}$$

with weighting factors $k_i \in [0,1]$ to reduce the impact of the individual IPQ components if necessary. Here, we set all $k_i = 1$ because all error types are equally detrimental to the downstream analysis, but some practical implementations of this work will require different weighting. For bounding box annotations instead of per-pixel annotations, we suggest a lower k_1; for sparse annotations that omit instances, we suggest $k_2 = 0$. Fig. 2 illustrates the error types detected by each factor.

2.2.2. Data

Manual annotation of 3D data is particularly challenging due to slice-based interfaces and computational constraints [45]. We therefore recommend using synthetic 3D data to benchmark segmentation performance. Domain gaps between synthetic and real data are addressed by combining data simulation with cycleGAN-based domain adaptation [37,46,47]. With a suitably designed and weighted loss function, domain adaptation preserves the original instance geometry while providing complete synthetic annotations for segmentation optimization.

2.2.3. Parameter space and problem definition

To avoid computationally expensive retraining of pretrained segmentation models, segmentation optimization focuses on conceptual parameters, namely the choice of a model and postprocessing parameterization. Postprocessing methods are selected to directly address the error types captured by IPQ, including morphological operations, instance merging, and instance splitting. The optimization problem is formulated as

$$(\hat{m}, \hat{\theta}) = \arg\max_{m\in M,\, \theta\in\Theta} \sum_{x,y\in X} \mathrm{IPQ}\big(m(x),\, \theta,\, y\big), \tag{5}$$

where $M = \{m_i(x)\}$ is a discrete set of segmentation models, $\Theta \subset \mathbb{R}^d$ denotes the continuous postprocessing parameter space, and X is a synthetic 3D benchmark dataset of images x and annotations y. $\hat{m}$ and $\hat{\theta}$ are the optimal model and parameters obtained by the optimization process.

2.2.4. Bayesian optimization setup

Segmentation predictions are computed once per model and reused during optimization to avoid computationally expensive repetitions. A Gaussian process surrogate with a Matérn-5/2 kernel (see supplementary material 4.4) models the postprocessing parameter space. Categorical parameters are modeled using a categorical kernel, while continuous and ordinal parameters are modeled using the Matérn kernel. The acquisition function is expected improvement, which is optimized over the mixed parameter space by alternating optimization of the continuous and discrete parameters, with categorical and ordinal parameters restricted to their respective feasible values. Optimization is terminated after 120 iterations.

2.3. Classification optimization

2.3.1. Objective and data

Classification optimization requires annotated data. Instance segmentation using the optimized segmentation model predicts instances, enabling efficient annotation without manual instance identification. Additionally, a semi-supervised dataset is constructed using label spreading [29]. We choose to apply label spreading because it utilizes the differences in the geometry of the present classes, which is similar to the approach a human annotator would take. Feature vectors consist of the total area, total perimeter, total equivalent diameter, and the magnitudes of the first ten Fourier coefficients of the object contour. The vectors are normalized and reduced using principal component analysis. Similarities between the feature vectors of the predicted instances are computed as Euclidean distances using an RBF kernel in the reduced feature space to assign pseudo-annotations to all unannotated data in the semi-supervised dataset. Datasets are split into training, validation, and test sets, with validation accuracy serving as the optimization objective.

Table 1
Overview of the datasets used in the experiments.

Dataset	Dimensions	No. of instances	Voxel size (μm)	Classdistribution
Myotube real	24–64 × 1024 × 1024	–	0.568 × 0.568 × 1.0	–
Myotube synthetic	90 × 1024 × 1024 × 4	111,458	–	16/125/95/148
Core-Shell real	70–109 × 1024 × 1024 × 3	–	0.33 × 0.33 × 1.25	–
Core-Shell synthetic	70 × 1024 × 1024	3358	–	100/100/100
C3DH-H157	35 × 832 × 992	519	0.126 × 0.126 × 0.5	–
C3DL-MDA231	30 × 512 × 512	957	1.242 × 1.242 × 6.0	–

2.3.2. Parameter space

Classifier optimization targets conceptual design choices, including encoder and classifier head architectures, preprocessing, and pretraining strategies. Multiple pretrained encoders of different sizes are included to balance under- and overfitting. Classifier heads vary in the number and size of linear layers to adapt the encoder feature space to the number of classes, and in the design of additional transformations. Multiple architectures can be included in the BO process to identify an optimal fit for the data. Pretraining strategies determine whether encoder weights are frozen or retrained. The 3D-AOP chooses a pretraining strategy that reduces the risk of under- and overfitting to the data. Preprocessing methods can incorporate a priori knowledge derived from segmentation predictions, such as instance position and geometry. Each classifier configuration is trained using the Adam optimizer and cross-entropy loss, with data augmentation and early stopping.

2.3.3. Bayesian optimization setup

As each evaluation requires full classifier training, BO is employed to efficiently explore the discrete parameter space. A random forest surrogate is used, and *expected improvement* serves as the acquisition function.

3. Experiments

3.1. Overview

We conduct four experiments to validate the 3D-AOP. For each optimization process, we propose explicit options for all optimized parameters. These options can be replaced by alternatives suited to a given application. We further provide dedicated datasets for each BO process. For segmentation optimization, we construct a synthetic 3D benchmark per experiment and reduce the domain gap using a cycle-GAN [37]. For classifier optimization, class annotations are obtained using the 3D-AOP's assisted instance-based annotation workflow. Table 1 gives a brief overview over the data, while the subsequent subsections describe it in-depth.

3.2. Data

3.2.1. Myotube nuclei

The first experiment uses *in vitro* Myotube cultures derived from human induced pluripotent stem cells, prepared following Couturier et al. [40]. Three of the four immunofluorescence channels are used for classification: DAPI, α sarcomeric actinin, and S100β. We manually annotate 16 Schwann cell nuclei, 125 Myotube nuclei, 95 debris, and 148 other samples. Following common practice, a small validation set of 10% monitors over- and underfitting, while 15% of the data is used as the test set. Because the Schwann cell class is small, 50% of samples are assigned to the validation set. The synthetic Myotube benchmark contains 111,458 nuclei across ten volumes of size 90 × 1024 × 1024 pixels. Fig. 3 shows an exemplary 3D depiction of the real data and the synthetic benchmark.

Table 2
Comparison of the six encoders included.

Name	Abbrev.	Acc@5	Parameters (M)
ResNet18 [25]	RN18	89.08%	11.7
ResNet101 [25]	RN101	93.55%	44.5
Swin V2 [26]	Swin	96.87%	87.9
EfficientNet V2 [27]	EN	97.79%	118.5
ConvNeXt-L V1 [28]	CN	96.98%	197.8
CellposeSAM [18]	CSAM	–	305

3.2.2. Core-shell

The second experiment is conducted on cellular core–shell assembloids composed of transfected HEK293T cells, which comprise three cell classes. A total of 360 cells are annotated using assisted annotation. To emphasize validation-driven optimization, only 60 cells are used during training (48 train, 12 test), while 300 cells form an extensive validation set. For segmentation optimization, we generate five synthetic volumes of size 70 × 1024 × 1024 pixels containing 3358 cells arranged inside the spheroid.

3.2.3. Cell tracking challenge

The 3D CTC comprises ten datasets with annotations for segmentation and tracking [41]. We include Fluo-C3DH-H157 (120 volumes, size 35 × 832 × 992 pixels, 519 instances) and Fluo-C3DL-MDA231 (24 volumes, size 30 × 512 × 512 pixels, 957 instances). For these CTC datasets, we do not synthesize data, as they are already established segmentation benchmarks.

3.3. Parameters

3.3.1. Encoders

The encoders in Table 2 are included in classifier optimization. Five of the encoders are pretrained on ImageNet. The table shows their Acc@5, which indicates whether the correct class appears among the five most confident predictions on the ImageNet dataset [24]. Additionally, the CellposeSAM encoder is included.

3.3.2. Classifier heads

We include two standardized classifier heads derived from common practices [30,31]: a *slice classifier* and a *volume classifier*. The *slice classifier* applies self-attention over a stack of averaged 2D slice representations to focus inter-slice dependency features. The *volume classifier* performs 3D convolutions over a 3D stack of 2D representations to efficiently encode volumetric features. Architectural details are provided in the supplementary material 4.4.

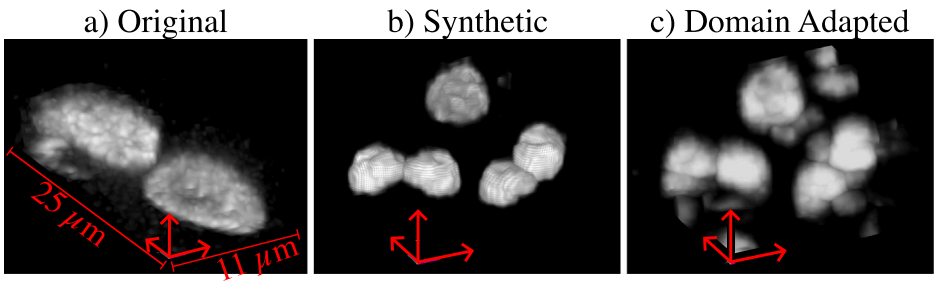


Fig. 3. 3D depiction of the synthetic segmentation benchmark. (a) Original reference, (b) synthetic image, and (c) synthetic image after domain adaptation via cycleGAN.

Table 3
Segmentation models included in all experiments.

Model	Abbreviation	Parameters
StarDist 2D Versitile Fluo [48–50]	SD	1.4M
PlantSeg 3D Nuc Platinum [51]	PSP	1M
PlantSeg 3D Confocal UNet [16]	PSG	4M
CellposeSAM [18]	CSAM	304M

3.3.3. Pretraining methods

To balance over- and underfitting, we include three pretraining options. (i) *Fully-supervised pretraining*: ImageNet-pretrained encoder weights, and the classifier head are retained on manual labels with 80% of encoder weights frozen. (ii) *Semi-supervised pretraining*: the encoder is first pretrained using the semi-supervised dataset, then retrained on manual labels with 80% of weights frozen [29]. (iii) *None*: the classifier is trained exclusively on the manually annotated dataset.

3.3.4. Preprocessing methods

Segmentation predictions provide a priori knowledge about instance position and geometry. We include two preprocessing methods to incorporate this knowledge. The *mask method* replaces the instance channel with the binary prediction to suppress neighboring instances and emphasize geometry, at the cost of attenuating surface features. The *distance method* retains surface features and reduces the effect of incorrect or incomplete segmentation prediction while encoding instance proximity by scaling intensities via a distance transform:

$$I'(p_1) = I(x) \times \exp\left(-\frac{1}{\sigma} \min_{p_2 \in P} \|p_1 - p_2\|_2\right), \tag{6}$$

where $I : \mathbb{N}^3 \to [0,1]$ is the intensity, $P \subseteq \mathbb{N}^3$ is the set of voxel positions inside the binary prediction, and p_1 denotes a voxel outside the mask. Any distance transformation can be used here. We choose an exponential decay because of the rapid initial decrease in the area immediately surrounding the instance and a sigma parameter to manually control the spatial extent of the surrounding image that remains sufficiently preserved.

3.3.5. Segmentation models

Table 3 summarizes the segmentation models included in segmentation optimization. As the included StarDist model is a 2D model, we perform connectivity-based slice merging.

3.3.6. Segmentation postprocessing

We optimize seven postprocessing parameters chosen to directly address the error types detected by IPQ. Morphological operations (erosion/dilation and opening/closing) are applied to a binary mask that is then applied to the instance mask. Instance splitting is performed using watershed, and instance merging is based on an adjacency graph with a distance metric combining contour continuity, connecting surface smoothness, and relative intersection area. All optimized parameters are listed in Table 4. All parameters of a category are set to zero to omit that postprocessing category entirely.

4. Results & discussion

4.1. Segmentation

To ensure data- and model-independent comparisons of the segmentation performance, we relate the results of each experiment to an experiment-specific baseline and to a random search with the same evaluation budget as the executed BO process. This baseline is computed as the mean IPQ of all models without any postprocessing, representing the performance of an unoptimized configuration. Fig. 4 shows the IPQ (gray) values as well as the RQ (orange), SQ (pink), and IQ (blue) factors of the baseline (filled bar), the random search (thick-striped bars), and the optimized segmentation model (thin-striped bar) for every experiment. n_1 indicates the number of IPQ values for the random search and BO process, and n_2 indicates the number of values for the baseline. Statistical significance is assessed using pairwise single-sided Welch's t-tests, with the null hypothesis being that the mean value of the lower-performing group is greater than or equal to that of the higher-performing group. Significance levels are $p < 0.05$ (*), $p < 0.01$ (**), and $p < 0.001$ (***), and are denoted by stars in the figure.

The optimized bars show significantly higher values than the baseline and the randomized search across all metrics in almost all experiments, indicating that the optimization process yields substantially better segmentation performance than randomly selecting a segmentation model and applying no postprocessing. The low mean SQ value of 0.37 for the Core-Shell experiment suggests that the segmentation models cannot predict the correct instance areas from the synthetic data. Because of this, morphological operators have great potential to improve segmentation performance, as they affect predicted instance sizes. The significant improvement of 0.13 in SQ by the BO process, and the decrease of 0.08 by the random search, show that the BO process was successful in identifying parameters to enhance the predicted areas, whereas the random search was not.

The Myotube experiment shows no RQ and IQ improvement by the BO process over the baseline, but a 0.48 improvement in the SQ metric. This indicates that the pretrained segmentation models already successfully identify all instances, but they predict them either too large or too small, highlighting the importance of morphological operators for the Myotube data. For the Myotube experiment, every metric decreases significantly with the random search. This shows that the parameter space is too complex for a random search to find a suitable solution in a limited number of iterations.

In the CTC experiments, we observe a particularly strong improvement of 0.79 and 0.33 in the IQ metric, while RQ only improves by 0.01 and 0.06. Before optimization, the models tested here confidently find almost all instances, as indicated by the high baseline RQ of 0.99 and 0.86, but they repeatedly split individual instances in their predictions, as indicated by the low baseline IQ of 0.21 and 0.67, rendering a good parameterization of merging algorithms imperative. While the random search does identify a parameterization that improves IQ by 0.38 and 0.24, the improvement is significantly lower than that of the BO process, and it entails a decrease of RQ by 0.02 and 0.35. The

Table 4
Parameters included in the optimization of postprocessing methods.

	Parameter	Values	Description
Morphology	θ_{ED}	$[-10, 10]$	Radius of the spherical structuring element for erosion (negative) or dilation (positive).
	θ_{CO}	$[-5, 5]$	Radius of the spherical structuring element for an opening (negative) or closing (positive) operation.
Merging	θ_{Mc}	$[0, 1]$	Weight of contour continuity at intersections (from 1D profiles along the connecting axis) for the adjacency graph.
	θ_{Ms}	$[0, 1]$	Weight of connecting-surface smoothness for the adjacency graph.
	θ_{Mr}	$[0, 1]$	Weight of relative intersecting area (normalized by smaller-instance surface area) for the adjacency graph.
Splitting	$\theta_{S\sigma}$	$[0, 1]$	Gaussian smoothing strength for watershed.
	θ_{St}	$[0, 1]$	Peak density via distance-transform quantile threshold for watershed.

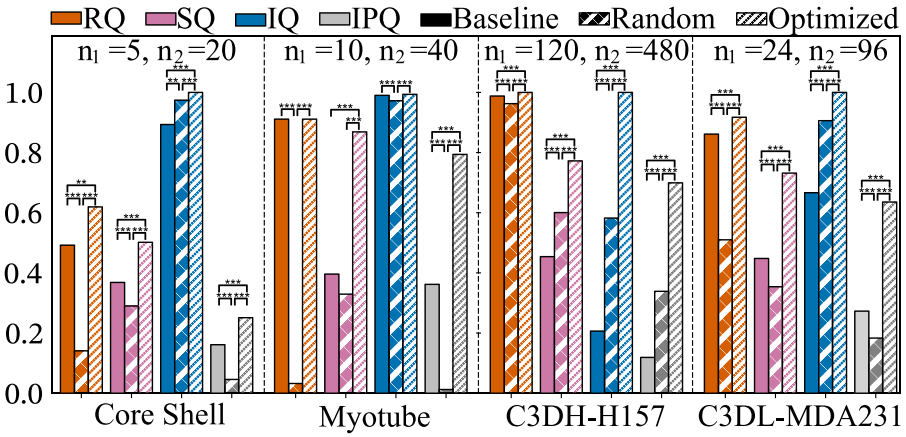


Fig. 4. IPQ results of every experiment. Full bars present the baseline, thick-striped bars the random search, and thin-striped bars the model optimized by the 3D-AOP.

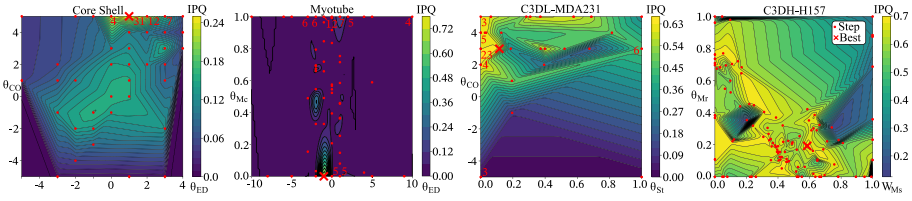


Fig. 5. 2D projection of a subset of the parameter space with resulting IPQ values. The dots denote exploration steps, and the cross denotes the found optimum. Red numbers next to dots indicate that a specific parameter value combination was evaluated multiple times.

BO process thus not only alleviates the weakness of the baseline more efficiently than a random search, but also improves upon the pretrained models' strengths.

This opposing behavior of IQ and RQ holds for every experiment comparing random search to the baseline, revealing a trade-off introduced by the new IQ factor. Optimizing the parameterization of the splitting and the merging algorithm with respect to the IQ factor can lead to false merging of neighboring individual instances because the resulting gain in IQ can outweigh the accompanying reduction in RQ. With the optimized search, however, this is the case in none of the experiments.

Next, we demonstrate that identifying an optimal parameterization is difficult to do manually in this complex search space, and that a BO process can determine effective settings. Fig. 5 shows exemplary contour plots of two postprocessing parameters that are particularly important for each experiment. The plots show the resulting IPQ and include only evaluations that use the model achieving the highest IPQ.

The first contour plot shows the Core-Shell experiment search space, including both morphological operators. As mentioned, they are especially important to this experiment. In the plot, two local optima are visible, one being $\theta_{ED} = 0, \theta_{CO} = 0$, which is also the starting point for the BO process. While approaches like Coordinate Descent will get stuck in local optima, as this plot shows, the BO process is able to find a superior configuration in this case, indicating that the implementation of a BO process is worthwhile. Next, the θ_{ED}-θ_{Mc} parameter space of the Myotube data experiment is displayed. The mean baseline RQ and IQ are 0.92 and 0.99, but the SQ is only 0.4. Because the pretrained model already identifies almost all instances correctly, instance merging was effectively ruled out by the BO process, and the contour plot only shows IPQ values higher than 0.2 at the optimum of $\theta_{Mc} = 0.0$. At the global optimum, all merging and splitting parameters are zero. This shows that the BO process can decide to omit postprocessing methods completely if required. Along the $\theta_{Mc} = 0$ axis, $\theta_{ED} = -1$ shows a substantial improvement with an IPQ value of 0.8. This indicates that the pretrained segmentation model was

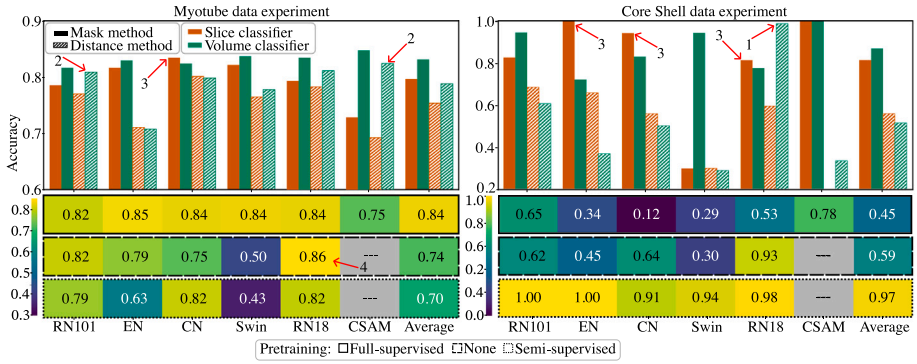


Fig. 6. Results of the classification optimization processes per encoder, classifier head, pretraining method, and preprocessing method with four highlights: (1) On average worst combination performs best, (2) second to worst combination performs best, (3) *slice classifier* outperforms *volume classifier*, and (4) best pretraining strategy is outperformed.

biased toward predicting slightly larger instances, which the BO process identified and fixed. A table of the optimal parameterization with the resulting IPQ for every experiment and segmentation model is given in the supplementary material 4.4.

The third contour plot is of the C3DH-H157 data. As mentioned before, the parameterization of the merging algorithm is especially important for this experiment, thus the merging parameters θ_{Ms} and θ_{Mr} are displayed. The shape of the search space is very complex due to the high number of other parameters influencing the performance. Still, an optimum is found in a densely explored region of the search space. This demonstrates the necessity of an efficient optimization of postprocessing parameterization, as manually finding a good solution would be challenging here. Similarly, the second CTC experiment, the C3DL-MDA231 data, shows a splitting algorithm parameter and the Closing/Opening parameter. The contour plot shows only a few exploration steps that include a high value for the splitting algorithm. Because they lead to low IPQ values, the majority of the parameter configurations tested, as well as the optimum, lie in the range $\theta_{St} < 0.2$. The second parameter on display shows that higher θ_{CO} values lead to higher IPQ values, presumably because the segmentation models introduce gaps between the predicted instances they split an individual annotation instance with, which are closed by the closing operator. Search space regions of negative θ_{CO} values are thus not extensively explored.

4.2. Classification

Fig. 6 shows an overview of the validation accuracies of the classifiers trained in the classifier optimization process. A full table including every validation accuracy is given in the supplementary material. On the left, the results of the Myotube data experiment are given, and on the right, the Core-Shell data experiment results are displayed. Both bar graphs show the mean accuracy of each classifier using a specific encoder and a specific classifier head-preprocessing method combination. Below that, heatmaps visualize the mean validation accuracies achieved by the classifier for every encoder–pretraining combination.

The bar graphs reveal that the *mask method* is, on average, superior to the *distance method* across both experiments, with accuracy differences of 4% and 29%, respectively. For the Myotube data experiment, the *volume classifier* is, on average, 3% better than the *slice classifier*. Validation accuracies range from 12% to 100%. This indicates that the design choices matter and that the process is able to identify methods that outperform others.

We observe that some combinations outperform combinations with higher average accuracies. For example, the *volume classifier* using the *distance method* with the RN18 encoder achieves an accuracy of 98.7% on the Core-Shell data (1), even though the combination of *volume classifier* and *distance method* is the worst on average (51%).

Table 5
Comparison of inference durations per encoder.

Name	Duration one image (s)
ResNet18	0.42
ResNet101	0.47
Swin V2	0.61
EfficientNet V2	0.59
ConvNeXt-L V1	0.73
CellposeSAM	2.23

Similarly, the *volume classifier-distance method* combination has a 10% and 2% higher accuracy than *slice classifier-mask method* for the CSAM and RN101 encoders, respectively, on the Myotube data (2). The *slice classifier-mask method* combination, in turn, often outperforms the on average best combination across both experiments (3). Out of the six encoders per experiment, only two show the same interaction pattern between design parameters as observed in the average of the encoders. For the remaining encoders, the relative performance of each classifier head-preprocessing method combination differs. This variation of relative combination performance suggests that the average performance is not representative of individual combinations and, therefore, that a more in-depth search is necessary to identify optimal combinations.

For the Myotube data experiment, *fully-supervised pretraining* yields the best results on average with 84%. In contrast, for the Core-Shell data, it is better to pretrain the encoders on a semi-supervised dataset (97%). This shows that the results of the optimization process cannot be generalized to find the perfect classifier design for every dataset.

The overall best classifier for the Myotube data is the smallest model, an RN18 encoder with no pretraining (4), likely because the CNN architecture is sufficiently simple, and the encoder is small enough to generalize well on the limited dataset. We further prove the hypothesis that smaller, simpler encoders generalize better by computing the Pearson correlation between the number of parameters of the CNN encoders and the validation accuracy, obtaining a correlation coefficient of $r = -0.97, p = 0.027$. Table 5 shows the inference times of every encoder to process one image.

We observe that, for example, the RN18 encoder achieves the same maximum accuracy on the Core-Shell dataset as the CSAM encoder while processing about five times faster. Similarly, on the Myotube data, the RN18 encoder (86%) and the EN encoder (85%) achieve maximum accuracies on par with the overall maximum, achieved by the CSAM encoder (87%), with a much shorter processing duration. The optimization process thus also reveals trade-offs of processing time and accuracy, enabling classifier design based on application requirements.

4.3. Computational cost

We estimate the total computational cost for a representative large dataset comprising 15 volumes of size 70 × 1024x1024 with 150,000

Table 6
Computational cost of running the full 3D-AOP on a large dataset of 15 volumes of size 70 × 1024 × 1024, 150,000 total instances, 4500 labeled instances, 75 epochs of classifier training, and 30 iterations of each optimization stage.

Pipeline stage	Evaluations	GPU hours	CPU hours
Segmentation optimization	30	4	25
Classification optimization	30	120	0.4
Total		124	25.4

total instances, 4500 of which are labeled for classification, by performing 30 iterations of both optimization stages. Our annotated datasets include significantly fewer total and labeled instances. The GPU time of both BO processes scales with the volume size, and the classifier optimization duration increases with the number of labeled instances. The computational cost of the classifier optimization is also dependent on the number of epochs and early stopping implementation of the training routine. We include no early stopping for the timing analysis and execute 75 epochs. Table 6 shows the hours required on an NVIDIA GeForce RTX 3090 Ti with 24 GB of VRAM as well as the total non-GPU-accelerated hours.

In total, optimizing a large dataset of this size requires about 124 GPU hours and 30 CPU hours, with classifier optimization accounting for most GPU utilization. Thus, the main computational bottleneck of the 3D-AOP is the classifier design search, particularly when large pseudo-labeled training sets are available. The classification optimization process, however, does not require the full training data set. Especially with large datasets, a subset is sufficient.

4.4. Limitations

The limitations of the 3D-AOP are subject to the ever-expanding corpus of 3D analysis methods. We do not present any new model that will generalize to unexplored domains, but rather propose a method to compare existing methods and models for a specific domain. Because there are numerous segmentation and classification models that perform well for many biomedical domains, the remaining bottleneck for experts in these domains is a streamlined and accurate assessment of the current model landscape. The 3D-AOP is thus applicable to a wide range of common biological samples like bright field, fluorescence, and electron microscopy of tissue, cells, and others. If no model and pre- or postprocessing method performs well on this domain, the optimization can instead identify the limitations of the available model landscape. Filamentous structures such as the myotube strings surrounding the nuclei analyzed in this study, for example, are not captured well by any current segmentation model, and the 3D-AOP thus finds no parameterization that achieves a satisfactory IPQ value.

CRediT authorship contribution statement

David Exler: Writing – original draft, Visualization, Software, Methodology, Conceptualization. **Joaquin Eduardo Urrutia Gómez:** Writing – review & editing, Supervision. **Martin Krüger:** Software. **Maike Schliephake:** Data curation. **John Jbeily:** Data curation. **Mario Vitacolonna:** Data curation. **Rüdiger Rudolf:** Data curation. **Markus Reischl:** Writing – review & editing, Supervision, Project administration.

Funding

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant INST 874/9-1 (R. Rudolf).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.slast.2026.100467.

References

[1] Ghose S, Ju Y, McDonough E, Ho J, Karunamurthy A, Chadwick C, Cho S, Rose R, Corwin A, Surrette C, Martinez J, Williams E, Sood A, Al-Kofahi Y, Falo LD, Börner K, Ginty F. 3D reconstruction of skin and spatial mapping of immune cell density, vascular distance and effects of sun exposure and aging. Commun Biol. 2023;6:718. http://dx.doi.org/10.1038/s42003-023-04991-z, URL https://pmc.ncbi.nlm.nih.gov/articles/PMC10356782/.

[2] Du L, Yang H. Spatial omics in 3D culture model systems: decoding cellular positioning mechanisms and microenvironmental dynamics. J Transl Med 2025;23(1):1356. http://dx.doi.org/10.1186/s12967-025-07390-6, URL https://doi.org/10.1186/s12967-025-07390-6.

[3] Minaee S, Boykov Y, Porikli F, Plaza A, Kehtarnavaz N, Terzopoulos D. Image Segmentation Using Deep Learning: A Survey. IEEE Trans Pattern Anal Mach Intell 2022;44(7):3523–42. http://dx.doi.org/10.1109/TPAMI.2021.3059968, URL https://ieeexplore.ieee.org/abstract/document/9356353.

[4] Gertych A, Ma Z, Tajbakhsh J, Velásquez-Vacca A, Knudsen BS. Rapid 3-D delineation of cell nuclei for high-content screening platforms. Comput Biol Med 2016;69:328–38. http://dx.doi.org/10.1016/j.compbiomed.2015.04.025, URL https://www.sciencedirect.com/science/article/pii/S0010482515001432.

[5] Ling Y, Wang Y, Dai W, Yu J, Liang P, Kong D. MTANet: Multi-Task Attention Network for Automatic Medical Image Segmentation and Classification. IEEE Trans Med Imaging 2024;43(2):674–85. http://dx.doi.org/10.1109/TMI.2023.3317088, URL https://ieeexplore.ieee.org/document/10255649.

[6] Shaker A, Maaz M, Rasheed H, Khan S, Yang M-H, Shahbaz Khan F. UNETR++: Delving Into Efficient and Accurate 3D Medical Image Segmentation. IEEE Trans Med Imaging 2024;43(9):3377–90. http://dx.doi.org/10.1109/TMI.2024.3398728, URL https://ieeexplore.ieee.org/document/10526382.

[7] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab N, Hornegger J, Wells WM, Frangi AF, editors. Medical image computing and computer-assisted intervention. Cham; 2015, p. 234–41. http://dx.doi.org/10.1007/978-3-319-24574-4_28.

[8] Boser BE, Guyon IM, Vapnik VN. A training algorithm for optimal margin classifiers. In: Fifth annual workshop on computational learning theory. New York, NY, USA; 1992, p. 144–52. http://dx.doi.org/10.1145/130385.130401, URL https://dl.acm.org/doi/10.1145/130385.130401.

[9] Yarowsky D. Unsupervised word sense disambiguation rivaling supervised methods. In: 33rd annual meeting of the association for computational linguistics. 1995, p. 189–96, URL https://aclanthology.org/P95-1026.pdf.

[10] Milletari F, Navab N, Ahmadi S-A. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. 2016, http://dx.doi.org/10.48550/arXiv.1606.04797, URL http://arxiv.org/abs/1606.04797 arXiv:1606.04797 [cs.CV].

[11] Çiçek Ö, Abdulkadir A, Lienkamp SS, Brox T, Ronneberger O. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. 2016, http://dx.doi.org/10.48550/arXiv.1606.06650, URL http://arxiv.org/abs/1606.06650 arXiv:1606.06650 [cs.CV].

[12] Isensee F, Jaeger PF, Kohl SAA, Petersen J, Maier-Hein KH. nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. Nature Methods 2021;18(2):203–11. http://dx.doi.org/10.1038/s41592-020-01008-z, URL https://www.nature.com/articles/s41592-020-01008-z.

[13] Lalit M, Tomancak P, Jug F. Embedding-based Instance Segmentation in Microscopy. In: Proceedings of the fourth conference on medical imaging with deep learning. PMLR; 2021, p. 399–415, URL https://proceedings.mlr.press/v143/lalit21a.html.

[14] Archit A, Freckmann L, Nair S, Khalid N, Hilt P, Rajashekar V, Freitag M, Teuber C, Spitzner M, Tapia Contreras C, Buckley G, von Haaren S, Gupta S, Grade M, Wirth M, Schneider G, Dengel A, Ahmed S, Pape C. Segment Anything for Microscopy. Nature Methods 2025;22(3):579–91. http://dx.doi.org/10.1038/s41592-024-02580-4, URL https://www.nature.com/articles/s41592-024-02580-4.

[15] Israel U, Marks M, Dilip R, Li Q, Yu C, Laubscher E, Iqbal A, Pradhan E, Ates A, Abt M, Brown C, Pao E, Li S, Pearson-Goulart A, Perona P, Gkioxari G, Barnowski R, Yue Y, Van Valen D. CellSAM: A Foundation Model for Cell Segmentation. BioRxiv 2025. http://dx.doi.org/10.1101/2023.11.17.567630, URL https://pmc.ncbi.nlm.nih.gov/articles/PMC10690226/.

[16] Wolny A, Cerrone L, Vijayan A, Tofanelli R, Barro AV, Louveaux M, Wenzl C, Strauss S, Wilson-Sánchez D, Lymbouridou R, Steigleder SS, Pape C, Bailoni A, Duran-Nebreda S, Bassel GW, Lohmann JU, Tsiantis M, Hamprecht FA, Schneitz K, Maizel A, Kreshuk A. Accurate and versatile 3D segmentation of plant tissues at cellular resolution. ELife 2020;9. http://dx.doi.org/10.7554/eLife.57613, URL https://doi.org/10.7554/eLife.57613.

[17] Schwartz MS, Moen E, Miller G, Dougherty T, Borba E, Ding R, Graf W, Pao E, Valen DV. Caliban: Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning. BioRxiv 2024. http://dx.doi.org/10.1101/803205, URL https://www.biorxiv.org/content/10.1101/803205v5.

[18] Pachitariu M, Rariden M, Stringer C. Cellpose-SAM: superhuman generalization for cellular segmentation. 2025, http://dx.doi.org/10.1101/2025.04.28.651001, BioRxiv. URL https://www.biorxiv.org/content/10.1101/2025.04.28.651001v1.

[19] Soille PJ, Ansoult MM. Automated basin delineation from digital elevation models using mathematical morphology. Signal Process 1990;20(2):171–82. http://dx.doi.org/10.1016/0165-1684(90)90127-K, URL https://www.sciencedirect.com/science/article/pii/016516849090127K.

[20] Meyer F. Color image segmentation. In: 1992 international conference on image processing and its applications. 1992, p. 303–6, URL https://ieeexplore.ieee.org/abstract/document/785528.

[21] Chang R-F, Chen C-J, Liao C-H. Region-based image retrieval using edgeflow segmentation and region adjacency graph. In: IEEE international conference on multimedia and expo (ICME), vol. 3, 2004, p. 1883–6 Vol.3. http://dx.doi.org/10.1109/ICME.2004.1394626, URL https://ieeexplore.ieee.org/abstract/document/1394626.

[22] Belaid LJ, Mourou W. Image Segmentation: A Watershed Transformation Algorithm. Image Anal Ster 2009;28(2):93–102. http://dx.doi.org/10.5566/ias.v28.p93-102, URL https://www.ias-iss.org/ojs/IAS/article/view/852.

[23] Neubert P, Protzel P. Compact Watershed and Preemptive SLIC: On Improving Trade-offs of Superpixel Segmentation Algorithms. In: 22nd international conference on pattern recognition. 2014, p. 996–1001. http://dx.doi.org/10.1109/ICPR.2014.181, URL https://ieeexplore.ieee.org/abstract/document/6976891.

[24] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L. ImageNet Large Scale Visual Recognition Challenge. Int J Comput Vis 2015;115(3):211–52. http://dx.doi.org/10.1007/s11263-015-0816-y, URL https://doi.org/10.1007/s11263-015-0816-y.

[25] 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. 2016, p. 770–8, URL https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

[26] Liu Z, Hu H, Lin Y, Yao Z, Xie Z, Wei Y, Ning J, Cao Y, Zhang Z, Dong L, Wei F, Guo B. Swin Transformer V2: Scaling Up Capacity and Resolution. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022, p. 12009–19, URL https://openaccess.thecvf.com/content/CVPR2022/html/Liu_Swin_Transformer_V2_Scaling_Up_Capacity_and_Resolution_CVPR_2022_paper.html.

[27] Tan M, Le Q. EfficientNetV2: Smaller Models and Faster Training. In: Proceedings of the 38th international conference on machine learning. 2021, p. 10096–106, URL https://proceedings.mlr.press/v139/tan21a.html.

[28] Liu Z, Mao H, Wu C-Y, Feichtenhofer C, Darrell T, Xie S. A ConvNet for the 2020s. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022, p. 11976–86, URL https://openaccess.thecvf.com/content/CVPR2022/html/Liu_A_ConvNet_for_the_2020s_CVPR_2022_paper.html.

[29] Zhou D, Bousquet O, Lal T, Weston J, Schölkopf B. Learning with Local and Global Consistency. In: Advances in neural information processing systems. 16, 2003, URL https://proceedings.neurips.cc/paper/2003/hash/87682805257e619d49b8e0dfdc14affa-Abstract.html.

[30] Wang X, Girshick R, Gupta A, He K. Non-Local Neural Networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 7794–803, URL https://openaccess.thecvf.com/content_cvpr_2018/html/Wang_Non-Local_Neural_Networks_CVPR_2018_paper.html.

[31] Liu L, Zhang S, Kuang Z, Zhou A, Xue J-H, Wang X, Chen Y, Yang W, Liao Q, Zhang W. Group Fisher Pruning for Practical Network Compression. In: Proceedings of the 38th international conference on machine learning. 2021, p. 7021–32, URL https://proceedings.mlr.press/v139/liu21ab.html.

[32] Hay EA, Parthasarathy R. Performance of convolutional neural networks for identification of bacteria in 3D microscopy datasets. PLoS Comput Biol 2018;14(12):e1006628. http://dx.doi.org/10.1371/journal.pcbi.1006628.

[33] Nikolenko SI. Synthetic Data for Deep Learning, vol. 174, Springer; 2021, http://dx.doi.org/10.1007/978-3-030-75178-4, URL https://link.springer.com/10.1007/978-3-030-75178-4.

[34] Dunn KW, Fu C, Ho DJ, Lee S, Han S, Salama P, Delp EJ. DeepSynth: Three-dimensional nuclear segmentation of biological images using neural networks trained with synthetic data. Sci Rep 2019;9(1):18295. http://dx.doi.org/10.1038/s41598-019-54244-5, URL https://www.nature.com/articles/s41598-019-54244-5.

[35] Fu C, Lee S, Joon Ho D, Han S, Salama P, Dunn KW, Delp EJ. Three Dimensional Fluorescence Microscopy Image Synthesis and Segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2018, p. 2221–9, URL https://openaccess.thecvf.com/content_cvpr_2018_workshops/w44/html/Fu_Three_Dimensional_Fluorescence_CVPR_2018_paper.html.

[36] Eschweiler D, Rethwisch M, Jarchow M, Koppers S, Stegmaier J. 3D fluorescence microscopy data synthesis for segmentation and benchmarking. PLoS One 2021;16(12):e0260509. http://dx.doi.org/10.1371/journal.pone.0260509, URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0260509.

[37] Bruch R, Vitacolonna M, Nürnberg E, Sauer S, Rudolf R, Reischl M. Improving 3D deep learning segmentation with biophysically motivated cell synthesis. Commun Biol. 2025;8(1):43. http://dx.doi.org/10.1038/s42003-025-07469-2, URL https://www.nature.com/articles/s42003-025-07469-2.

[38] O'Hagan A. Curve Fitting and Optimal Design for Prediction. J R Stat Soc 1978;40(1):1–24. http://dx.doi.org/10.1111/j.2517-6161.1978.tb01643.x, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1978.tb01643.x.

[39] Jones DR, Perttunen CD, Stuckman BE. Lipschitzian optimization without the Lipschitz constant. J Optim Theory Appl 1993;79(1):157–81. http://dx.doi.org/10.1007/BF00941892, URL https://doi.org/10.1007/BF00941892.

[40] Couturier N, Hörner SJ, Nürnberg E, Joazeiro C, Hafner M, Rudolf R. Aberrant evoked calcium signaling and nAChR cluster morphology in a SOD1 D90A hiPSC-derived neuromuscular model. Front Cell Dev Biology 2024;12. http://dx.doi.org/10.3389/fcell.2024.1429759, URL https://www.frontiersin.org/journals/cell-and-developmental-biology/articles/10.3389/fcell.2024.1429759/full.

[41] Maška M, Ulman V, Delgado-Rodriguez P, Gómez-de Mariscal E, Nečasová T, Guerrero Peña FA, Ren TI, Meyerowitz EM, Scherr T, Löffler K, Mikut R, Guo T, Wang Y, Allebach JP, Bao R, Al-Shakarji NM, Rahmon G, Toubal IE, Palaniappan K, Lux F, Matula P, Sugawara K, Magnusson KEG, Aho L, Cohen AR, Arbelle A, Ben-Haim T, Raviv TR, Isensee F, Jäger PF, Maier-Hein KH, Zhu Y, Ederra C, Urbiola A, Meijering E, Cunha A, Muñoz-Barrutia A, Kozubek M, Ortiz-de Solórzano C. The Cell Tracking Challenge: 10 years of objective benchmarking. Nature Methods 2023;20(7):1010–20. http://dx.doi.org/10.1038/s41592-023-01879-y, URL https://www.nature.com/articles/s41592-023-01879-y.

[42] Ramalakshmi K, Krishna Kumari L. U-Net-based architecture with attention mechanisms and Bayesian Optimization for brain tumor segmentation using MR images. Comput Biol Med 2025;195:110677. http://dx.doi.org/10.1016/j.compbiomed.2025.110677, URL https://www.sciencedirect.com/science/article/pii/S0010482525010285.

[43] Douglas Z, Rahman A, Duggar WN, Wang H. Automatic head and neck tumor segmentation through deep learning and Bayesian optimization on three-dimensional medical images. Comput Biol Med 2025;192:110309. http://dx.doi.org/10.1016/j.compbiomed.2025.110309, URL https://www.sciencedirect.com/science/article/pii/S0010482525006602.

[44] Kirillov A, He K, Girshick R, Rother C, Dollar P. Panoptic Segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 9404–13, URL https://openaccess.thecvf.com/content_CVPR_2019/html/Kirillov_Panoptic_Segmentation_CVPR_2019_paper.html.

[45] Zheng H, Zhang Y, Yang L, Wang C, Chen DZ. An Annotation Sparsification Strategy for 3D Medical Image Segmentation via Representative Selection and Self-Training. Proc the AAAI Conf Artif Intell 2020;34(04):6925–32. http://dx.doi.org/10.1609/aaai.v34i04.6175, URL https://pmc.ncbi.nlm.nih.gov/articles/PMC7710151/.

[46] Böhland M, Bruch R, Löffler K, Reischl M. Unsupervised GAN epoch selection for biomedical data synthesis. Curr Dir Biomed Eng 2023;9(1):467–70. http://dx.doi.org/10.1515/cdbme-2023-1117, URL https://www.degruyterbrill.com/document/doi/10.1515/cdbme-2023-1117/html.

[47] Bruch R, Keller F, Böhland M, Vitacolonna M, Klinger L, Rudolf R, Reischl M. Synthesis of large scale 3D microscopic images of 3D cell cultures for training and benchmarking. PLoS One 2023;18(3):e0283828. http://dx.doi.org/10.1371/journal.pone.0283828, URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0283828.

[48] Weigert M, Schmidt U. Nuclei Instance Segmentation and Classification in Histopathology Images with Stardist. In: IEEE international symposium on biomedical imaging challenges. 2022, p. 1–4. http://dx.doi.org/10.1109/ISBIC56247.2022.9854534, URL https://ieeexplore.ieee.org/abstract/document/9854534.

[49] Weigert M, Schmidt U, Haase R, Sugawara K, Myers G. Star-convex Polyhedra for 3D Object Detection and Segmentation in Microscopy. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. 2020, p. 3666–73, URL https://openaccess.thecvf.com/content_WACV_2020/html/Weigert_Star-convex_Polyhedra_for_3D_Object_Detection_and_Segmentation_in_Microscopy_WACV_2020_paper.html.

[50] Schmidt U, Weigert M, Broaddus C, Myers G. Cell Detection with Star-Convex Polygons. In: Medical image computing and computer assisted intervention. Cham; 2018, p. 265–73. http://dx.doi.org/10.1007/978-3-030-00934-2_30.

[51] Yu Q. PlantSeg Plant Nuclei 3D UNet. 2023, http://dx.doi.org/10.5281/ZENODO.10070349.