GeneSegNet: A deep learning framework for cell segmentation by integrating gene expression and imaging

Wenguan Wang[1,*], **Yuxing Wang**[2,3,*], **Dongfang Liu**[2,*], **Wenpin Hou**[4], **Tianfei Zhou**[5,†], **and Zhicheng Ji**[3,†]

[1]The ReLER Lab from the Australian Artificial Intelligence Institute (AAII), University of Technology Sydney, Sydney, New South Wales, AUS.
[2]Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY, USA.
[3]Department of Biostatistics and Bioinformatics, Duke University School of Medicine, Durham, NC, USA.
[4]Department of Biostatistics, Columbia University Mailman School of Public Health, New York City, NY, USA
[5]Computer Vision Lab, ETH Zurich, Zurich, Switzerland.
[*]These authors contributed equally: Wenguan Wang, Yuxing Wang, and Dongfang Liu
[†]Corresponding author. E-mail: ztfei.debug@gmail.com; zhicheng.ji@duke.edu

ABSTRACT

When analyzing data from in situ RNA detection technologies, cell segmentation is an essential step in identifying cell boundaries, assigning RNA reads to cells, and studying the gene expression and morphological features of cells. We developed a deep-learning-based method, GeneSegNet, that integrates both gene expression and imaging information to perform cell segmentation. GeneSegNet also employs a recursive training strategy to deal with noisy training labels. We show that GeneSegNet significantly improves cell segmentation performances over existing methods that either ignore gene expression information or underutilize imaging information.

Main

RNA in situ hybridization (ISH) is a type of technology that measures the spatial locations of RNAs within tissue sections[1–5]. It provides great potential to understanding the spatial organizations of different cell types within a tissue, how different cell types interact with each other, and sizes and morphologies of cells. The RNA spatial location information is often accompanied by fluorescent staining (e.g., DAPI) images indicating locations of cells. However, the fluorescent images are noisy and cell boundaries are not directly captured and need to be inferred computationally. Accurately identifying cell boundaries is crucial for downstream cell-level analysis that relies on the identities and characteristics of individual cells.

There are two major categories of methods to identify cell boundaries. The first type of methods segment the image into small domains representing cell instances. They were originally designed for biomedical images without gene expression information and only utilize the imaging information. A classical example is the Watershed algorithm[6,7], and methods based on deep neural network (DNN) techniques such as U-Net[8], Deepcell[9], and Cellpose[10] have also been developed in recent years. A fundamental limitation of these methods is that imaging information alone may not be enough to accurately capture cell boundaries. For example, these methods are more likely to capture cell nuclei boundaries instead of cell boundaries when only nuclei staining is available. Plus, their performances are affected by noise in images and training labels. The second type of methods utilize spatial locations of RNA reads to infer cell boundaries. These methods include Baysor[11] and JSTA[12]. Baysor assigns RNAs to individual cells and optionally accepts prior cell segmentation results obtained using the first type of methods such as Watershed. JSTA improves initial Watershed cell segmentation by iteratively reassigning boundary pixels based on cell type probabilities. A major limitation of these methods is that the imaging information is not directly used to obtain cell boundaries. As a result, cell boundaries do not fully match the actual images and can appear rather artificial in real applications. Plus, Baysor requires an estimate of cell sizes as input and may yield problematic results if the cell size estimate is biased. JSTA requires reference single-cell RNA-seq and cell type annotations as inputs which are not always available in real data. Another method SSAM[13] directly assigns RNAs to cell types and does not provide cell segmentation.

Cell segmentation methods based on DNN techniques have shown superior performances in extracting information from extensive amounts of images with annotated cell segmentation labels[14,15]. In current practice, the training cell segmentation

labels are assumed to be unambiguous and accurate. However, this assumption often does not hold. Accurately annotating cell masks is challenging even for experienced experts, due to difficulties such as low contrast of cell boundaries and cell adhesion. Moreover, manual annotations are subjective in general, and collecting cell segmentation annotations of high quality is time-consuming and expensive, making it almost prohibitive to manually clean or correct the labels in large-scale. Consequently, existing cell segmentation datasets tend to contain inaccurate labels (Fig. 1a). Training DNNs on such noisy labeled datasets may cause suboptimal performance because it has been proved that DNNs easily overfit to noisy labels given their powerful learning ability[16]. Although the problem of learning from noisy labels has attracted growing attention in the field of deep learning[17,18], this fundamental challenge is less touched by recent DNN-based cell segmentation methods.

To address these issues, we propose GeneSegNet, a deep-learning-based cell segmentation method (Fig. 1b-c). GeneSegNet integrates both RNA locations and imaging information within a single unified and fully differentiable network architecture, which enables both visually plausible and biologically reasonable cell segmentation. The rational is that regions with high densities of RNAs are more likely to be within cells, and these regions may not have high pixel values in images such as nucleus staining. By converting discrete RNA spatial locations into a "pseudoimage" of 2D continuous probabilistic location maps, both RNA and imaging information can be simultaneously fed into the neural network as inputs. This allows GeneSegNet to effectively extract and deeply fuse meaningful features from the two modalities from the beginning and throughout the whole neural network. To tackle the challenge of noisy training labels, GeneSegNet is trained through a recursive strategy: network predictions of the previous training round are refined and used as new training labels for the next-round network training. In this way, GeneSegNet boosts the robustness by making a joint optimization of learning network parameters and approximating true training labels, as opposed to previous methods treating the noisy labels as fixed. Such a recursive training strategy is able to correct inaccurate labels and hence improve the performance of GeneSegNet.

We applied GeneSegNet to a mouse hippocampal area CA1 (hippocampus) dataset generated by in situ RNA detection[19] and a human non-small-cell lung cancer (NSCLC) dataset generated by Nanostring CoxMx platform[20] . RNA reads are sparse in the hippocampus dataset and dense in the NSCLC dataset, representing diverse spatial patterns (Fig. 1a). We compared GeneSegNet to two types of competing methods: image-based methods of watershed algorithm and Cellpose, and gene-based methods of Baysor and JSTA. Baysor was performed both with or without prior cell segmentation as input. For NSCLC dataset, both cell nucleus and cytoplasm staining information are available. Cellpose was performed with only nucleus images or with both nucleus and cytoplasm images, while GeneSegNet and Watershed algorithm were performed with only nucleus images.

Fig. 2a-b show cell segmentation results for some example cells in both datasets. Compared to image-based methods of Watershed algorithm and Cellpose that do not consider RNA information, GeneSegNet generates larger cell boundaries which include more RNA reads within cells. Plus, GeneSetNet is less likely to oversegment cells. Gene-based methods of Baysor and JSTA perform poorly. Both methods tend to oversegment or miss entire cells in some cases. JSTA generates artificial circles and fails to include considerable amounts of RNA reads and some pixels with high pixel values within cell boundaries. Baysor uses a kernel density estimate (KDE) approach that purely based on RNAs to obtain cell boundaries. The cell boundaries mismatch with actual images and result in unrealistic cell shapes. In summary, GeneSegNet benefits from both imaging and gene expression information, and generates smooth cell boundaries that capture actual cell shapes, include more RNA reads, and are less susceptible to imaging noise and oversegmentation.

We next performed a global and systematic evaluation of all methods. We designed a cell calling score to test the ability of each method to absorb RNA reads near a cell into cell boundaries (Methods). GeneSegNet outperforms all methods in both datasets (Fig. 2c). It is noteworthy that although GeneSegNet does not utilize the cytoplasm images that provide more information about cell boundaries, GeneSegNet still outperforms Cellpose with both nucleus and cytoplasm images in NSCLC dataset, demonstrating the advantage of using RNA information in cell segmentation. GeneSegNet and image-based methods have comparable performances to include pixels with high pixel values within cell boundaries, while gene-based methods fail to assign many high pixel values within cell boundaries (Fig. 2d). In comparison, GeneSegNet and gene-based methods have comparable performances to include RNAs residing in pixels with low pixels values within cell boundaries, while image-based methods have significantly lower performance, except Cellpose with both nucleus and cytoplasm images (Fig. 2e). These results demonstrate that GeneSegNet is able to fully utilize both gene expression and imaging information, while existing methods only focus on a single information source. As a result, GeneSegNet is able to capture more cells with larger numbers of reads assigned to each cell (Fig. 2f) and larger cell sizes (Fig. 2g). Consistent with observations in Fig. 2a-b, JSTA yields artificial cell boundaries with elongation close to 1 (representing a circle), while other methods yield more realistic cell boundary elongations (Fig. 2h). Baysor segmentation also yields significantly lower convexity compared to other methods (Fig. 2i), which is unrealistic in many cases and is consistent with observations in Fig. 2a-b.

Finally, to demonstrate the impact of cell segmentation on down-stream analysis, we calculated gene-gene correlations using gene expression matrices created by RNA assignments to cells (Methods). Since there is no gold standard, we tested if the gene-gene correlations calculated within biological replicates are consistent (Methods). GeneSegNet again has the best overall performance in both datasets (Fig. 2j), showing that the gene-gene correlations calculated by GeneSegNet are more

stable and reproducible.

GeneSegNet is a general framework. Its components such as the U-Net architecture can be replaced with more advanced machine learning architectures to appear in the future to further improve its performance. GeneSegNet also has the potential to be applied to other types of imaging data where information such as chromatin accessibility is simultaneously measured.

Methods

Methods Overview

Overall Model

GeneSegNet exploits both imaging information and spatial locations of RNA reads for cell segmentation, based on a general U-Net architecture[8] (Fig. 1b-c). Specifically, our GeneSegNet network f, parameterized by θ, takes as inputs an *intensity image* $I \in [0,1]^{W\times H}$ and an *RNA location map* $G \in [0,1]^{W\times H}$, both with $W \times H$ spatial resolution. The location map G is constructed by putting 2D Gaussian kernels on the detected RNA positions of image I, for easing the encoding of RNA location information. The outputs of GeneSegNet include i) a *confidence map* $\hat{M} \in [0,1]^{W\times H}$ that stores the probability of each pixel being inside cell regions, ii) a *center map* $\hat{C} \in [0,1]^{W\times H}$ that gives the likelihood of each pixel being a cell center, and iii) a two-channel *offset map* $\hat{V} \in \mathbb{R}^{2\times W\times H}$ indicating the offset vector of each pixel to the center of its corresponding cell instance. Through the confidence map $\hat{M}$, center map $\hat{C}$, and offset map $\hat{V}$, GeneSegNet can discover individual cells and recover their precise boundaries.

Recursive Training with Noisy Annotations

To address the issue of learning with ambiguous and noisy annotations, GeneSegNet is trained in a *recursive* fashion, *i.e.*, alternatively optimizing network parameters and estimating true cell segmentation labels on training data. Specifically, given a set of N training images $\boldsymbol{I} = \{I_n \in [0,1]^{W\times H}\}_{n=1}^N$, and corresponding RNA location maps $\boldsymbol{G} = \{G_n \in [0,1]^{W\times H}\}_{n=1}^N$, we denote the initial network parameters as $\theta^{(0)}$, and the initial labels for the confidence map, center map, and offset map as $\boldsymbol{Y}^{(0)} = \{(M_n^{(0)} \in \{0,1\}^{W\times H}, C_n^{(0)} \in \{0,1\}^{W\times H}, V_n^{(0)} \in \mathbb{R}^{2\times W\times H})\}_{n=1}^N$, which are derived from the noisy cell mask annotations of training datasets (*cf.* §Network Training). The update rules of network parameters θ and labels $\boldsymbol{Y}$ can be described as:

$$\boxed{\hat{\boldsymbol{Y}}} = \{f(I_n, G_n; \boxed{\theta})\}_{n=1}^N \quad \boxed{\boldsymbol{Y}}. \tag{1}$$

gradient update

level-set refinement

Specifically, at each recursive training round $t = \{0,1,2,\cdots\}$, we have:

- *Update θ with fixed $\boldsymbol{Y}$*: The network parameters θ are updated by the AdamW[21] optimizer on a cell segmentation loss function $\mathscr{L}$, which evaluates the network predictions $\hat{\boldsymbol{Y}}^{(t)}$ against current training labels $\boldsymbol{Y}^{(t)}$:

$$\theta^{(t+1)} \leftarrow \arg\min_{\theta} \mathscr{L}(\boldsymbol{Y}^{(t)}, \{f(I_n, G_n; \theta)\}_{n=1}^N). \tag{2}$$

  The detailed definition of the training loss $\mathscr{L}$ is given in §Network Training.

- *Update $\boldsymbol{Y}$ with fixed θ*: We use the updated network parameters $\theta^{(t+1)}$ to make predictions $\hat{\boldsymbol{Y}}^{(t+1)}$ over the training data $(\boldsymbol{I}, \boldsymbol{G})$. Then the level-set algorithm[22] is used for boundary refinement. Level-set has been widely used in image segmentation applications, due to its decent boundary detection capacity, which can robustly represent cell contours with weak edges and inhomogeneities[23]. Then the training labels are updated as:

$$\boldsymbol{Y}^{(t+1)} \leftarrow \text{level-set}(\hat{\boldsymbol{Y}}^{(t+1)}), \quad \text{where } \hat{\boldsymbol{Y}}^{(t+1)} = \{(\hat{M}_n^{(t+1)}, \hat{C}_n^{(t+1)}, \hat{V}_n^{(t+1)})\}_{n=1}^N = \{f(I_n, G_n; \theta^{(t+1)})\}_{n=1}^N. \tag{3}$$

The recursive training process is terminated when the IoU (Intersection over Union) score[10,24], computed based on the predictions $\hat{\boldsymbol{Y}}^{(t)}$ and $\hat{\boldsymbol{Y}}^{(t+1)}$, is larger than 0.93. An IoU that is high enough indiciates that the estimation of the training labels $\boldsymbol{Y}$ is converged and the optimization of network parameters θ is saturated.

Cell Mask Inference

When performing cell mask inference, GeneSegNet leverages its outputs of confidence map $\hat{M}$, center map $\hat{C}$, and offset map $\hat{V}$, to group pixels into individual cell instances. Specifically, we first locate a set of cell centers $\{\hat{o}_1, \hat{o}_2, \cdots, \hat{o}_K\}$ using center map $\hat{C}$, where $\hat{o}_k = (x_k^o, y_k^o) \in \{1,\cdots,W\} \times \{1,\cdots,H\}$ and $\hat{C}(\hat{o}_k) \geq 0.5$. Next, we assign each pixel $p = (x,y) \in \{1,\cdots,W\} \times \{1,\cdots,H\}$ that falls in cell regions ($\hat{M}(p) \geq 0.5$) to k^*-*th* cell center, according to:

$$k^* = \arg\min_k \|(p + \hat{V}(p)) - \hat{o}_k\|_2, \tag{4}$$

where $\hat{V}(p) \in \mathbb{R}^2$ gives the center-pointing vector along x- and y-axes, *i.e.*, $p+\hat{V}(p)$ is expected to be the cell center of p. Hence the cell pixel p is clustered to the detected center $\hat{o}_{k^*}$ that is nearest to $p+\hat{V}(p)$.

All notations are explained in Supplementary Table **S**1. Next, we articulate each of the contributing components of this study.

Datasets

Mouse hippocampus CA1 region (hippocampus) dataset

In this dataset, in situ sequencing technology was used to identify the spatial locations of RNA transcripts in 28 samples of mouse hippocampus CA1 regions[19]. Each sample has spatial locations of RNA reads and a one-channel grayscale image stained by DAPI. Watershed segmentation was conducted by the original study based on the DAPI stain. The image resolution varies from 3560×3034 to 6639×5548 pixels. The spatial locations of RNA reads and the DAPI stain images were used as inputs to GeneSegNet, and the watershed segmentation was used as the initial cell instance labels for GeneSegNet model training. The spatial locations of RNA reads, DAPI images, and watershed segmentation were directly downloaded from the original publication.

Human non-small cell lung cancer (NSCLC) dataset

In this dataset, NanoString CosMx^{TM} platform was used to detect the spatial locations of RNA in human NSCLC tissues[20]. Two imaging samples from the same NSCLC tissue were used in this study. Each image sample provides 5 stains with both nuclear and membrane markers (Membrane stain, PanCK, CD45, CD3, DAPI), and each image is a one-channel grayscale image. Cellpose[10] was used to perform cell segmentation using both nuclear and cytoplasm images by the original study. Each NSCLC sample contains around 30 high-resolution images with RNA reads information and the image resolution is 5472×3648 pixels. The spatial locations of RNA reads and the DAPI stain images were used as inputs to GeneSegNet, and the Cellpose segmentation was used as the initial cell instance labels for GeneSegNet model training. The spatial locations of RNA reads, DAPI images, and Cellpose segmentation were directly downloaded from NanoString website (https://nanostring.com/products/cosmx-spatial-molecular-imager/ffpe-dataset/).

Neural Network Architecture

Network Input

Driven by the premise that a comprehensive use of imaging information and gene expression can facilitate cell segmentation, we intend to feed both the image $I \in [0,1]^{W \times H}$ and RNA spatial positions $\boldsymbol{g} = \{g_l = (x_l^g, y_l^g) \in \{1, \cdots, W\} \times \{1, \cdots, H\}\}_{l=1}^{L}$ into our GeneSegNet network. To facilitate the simultaneous encoding of imaging and RNA information, we transform discrete RNA positions $\boldsymbol{g}$ into a single, continuous RNA location map $G \in [0,1]^{W \times H}$, following the common practice in human pose estimation[25]. Specifically, the RNA location map is achieved by placing an unnormalized 2D Gaussian with a small variance σ (*e.g.*, 7, 9) (Supplementary Table **S**2) at each gene position. Formally, the value of each position $p = (x,y)$ in G is computed as:

$$G(x,y) = \min(\textstyle\sum_{l=1}^{L} G_l(x,y), 1), \quad \text{where} \;\; G_l(x,y) = \exp\left(-\frac{(x-x_l^g)^2 + (y-y_l^g)^2}{2\sigma^2}\right) \tag{5}$$

Here $G_l(x,y)$ encodes the confidence that a gene appears in coordinate (x,y), based on the l-th detected gene location, *i.e.*, (x_l^g, y_l^g). The RNA location heatmap G is an aggregation of the Gaussian peaks of all the L gene location distributions, *i.e.*, $\{G_l\}_{l=1}^{L}$, in a single "heatmap", and we truncate each of its element to the range $[0,1]$ for numerical stability.

Based on the unified representation format of imaging information and RNA location information, the input to our GeneSegNet network is a two-channel "image" with $W \times H$ spatial resolution:

$$X^0 = [I,\ G] \in \mathbb{R}^{2 \times W \times H}, \tag{6}$$

where $[\cdot,\cdot]$ denotes the matrix concatenation along the channel dimension. For X^0, the first channel corresponds to the 2D density image I, and the second channel corresponds to the continuous RNA location map G.

Detailed Architecture

Our GeneSegNet is built upon the U-Net architecture[8,10] (Fig. 1c), which is arguably the standard neural network architecture for medical image segmentation. Basically, U-Net downsamples convolutional features several times and then reversely upsamples them in a mirror-symmetric manner. This U-shaped architecture is implemented as a fully convolutional encoder-decoder network, which is composed of an *encoder* for downsampling/extracting the feature maps and a *decoder* for upsampling/reconstructing the feature maps. Specifically, the encoder conducts feature extraction at four spatial scales. At each scale, 2×2 max-pooling is first adopted to downsample the input feature map, followed by two residual blocks with additive

identity mapping[26]. Each encoder residual block has two 3×3 convolutions, each of which is preceded by batch normalization (BN) and ReLU activation. Formally, the following operations are performed at *i-th* encoding scale:

$$\begin{aligned} \text{downsampling:} \quad & \dot{X}^i = \text{MaxPool}_{2\times2}(X^{i-1}), \\ \text{first residual block:} \quad & \ddot{X}^i = \text{BN-ReLU-Conv}_{3\times3}(\text{BN-ReLU-Conv}_{3\times3}(\dot{X}^i)) + \dot{X}^i, \\ \text{second residual block:} \quad & X^i = \text{BN-ReLU-Conv}_{3\times3}(\text{BN-ReLU-Conv}_{3\times3}(\ddot{X}^i)) + \ddot{X}^i. \end{aligned} \tag{7}$$

As such, the encoder repeatedly extracts more contextual feature maps X^i from preceding shallow feature maps X^{i-1}, yet at the cost of reduced spatial resolution. The valuable information from both the image I and the RNA location map G can be automatically gathered and deeply fused.

To rebuild the spatial context for fine-grained segmentation, the decoder progressively upsamples the final most expressive yet coarse-grained feature maps of the encoder. More specifically, analogous to the encoder, the decoder conducts feature reconstruction at four spatial scales. At each scale, 2×2 bilinear upsampling is first adopted, followed by two residual blocks, each of which also has two 3×3 convolutions. Formally, the following operations are performed at *i-th* decoding scale:

$$\begin{aligned} \text{upsampling:} \quad & \dot{Z}^i = \text{B-Upsample}_{2\times2}(Z^{i+1}), \\ \text{first residual block:} \quad & \ddot{Z}^i = \text{BN-ReLU-Conv}_{3\times3}(\text{BN-ReLU-Conv}_{3\times3}(\dot{Z}^i) + X^i) + \dot{Z}^i, \\ \text{second residual block:} \quad & Z^i = \text{BN-ReLU-Conv}_{3\times3}(\text{BN-ReLU-Conv}_{3\times3}(\ddot{Z}^i)) + \ddot{Z}^i. \end{aligned} \tag{8}$$

Note that the first residual block takes as input not only the previous feature map $\dot{Z}^i$, but also the feature map X^i from the counterpart of the encoder. Such *skip connection*, which appends encoder features into the decoder features at the same resolutions, allows to retain the spatial details that are dropped during downsampling. Through the above encoding-decoding process, the GeneSegNet model generates an expressive yet full-resolution feature map, allowing for fine-grained analysis of the input image I and RNA location map G.

Network Output

The output of the decoder is a 32-channel feature map of full resolution, *i.e.*, $Z^0 \in \mathbb{R}^{32\times W\times H}$. This feature map is separately fed into three different 1×1 convolutions. The first two convolutions are cascaded with sigmoid activation for respectively predicting the confidence map $\hat{M} \in [0,1]^{W\times H}$ and the center map $\hat{C} \in [0,1]^{W\times H}$. The last convolution is to directly regress the two-channel offset map $\hat{V} \in \mathbb{R}^{2\times W\times H}$. Formally, the final outputs of GeneSegNet are computed as:

$$\begin{aligned} \text{confidence map:} \quad & \hat{M} = \text{sigmoid}(\text{Conv}_{1\times1}(Z^0)) \quad \in [0,1]^{W\times H}, \\ \text{center map:} \quad & \hat{C} = \text{sigmoid}(\text{Conv}_{1\times1}(Z^0)) \quad \in [0,1]^{W\times H}, \\ \text{offset map:} \quad & \hat{V} = \text{Conv}_{1\times1}(Z^0) \quad \in \mathbb{R}^{2\times W\times H}. \end{aligned} \tag{9}$$

Network Training

Image data processing

We first split all images into training, validation, and testing sets. For the hippocampus dataset, 28 raw images were randomly split into 20 training images, 4 validation images, and 4 testing images. For NSCLC dataset, 59 raw images from two lung cancer samples are randomly split into 30 training images, 15 validation images, and 14 testing images. Since the raw images in both datasets are of very high resolution, it is infeasible to train and test deep learning models directly on the whole images due to the limitation of GPU memory. As a common practice, we randomly crop patch images with relatively small size of $W \times H$ from the raw training, validation, and test images, leading to thousands of patch images per dataset to be fed into GeneSegNet. Initial training labels were also cropped in the same way as the images. We empirically set $W \times H$ as 256×256, which is affordable for our GPUs.

Initial Training Label Generation

Before network training, we first derive the initial labels for the confidence map, center map, and offset map, from the instance segmentation annotations provided by the training datasets. Specifically, each training image I is associated with a set of K cell instance masks, *i.e.*, $\{S_k^{(0)} \in \{0,1\}^{W\times H}\}_{k=1}^K$. Note that the number of instances, K, varies across different images. For each pixel $p = (x,y)$, the corresponding value in the *k-th* cell instance mask, *i.e.*, $S_k^{(0)}(x,y)$, denotes whether p belongs to cell instance k (1) or not (0). Thus, the initial label $M^{(0)} \in \{0,1\}^{W\times H}$ for the confidence map is hence given as:

$$M^{(0)}(x,y) = \max(\{S_k^{(0)}(x,y)\}_{k=1}^K). \tag{10}$$

Therefore, $M^{(0)}(x,y)$ refers to if the pixel (x,y) is inside (1) or outside (0) of a certain cell instance.

The initial label $C^{(0)} \in \{0,1\}^{W\times H}$ for the center map is given as:

$$C^{(0)}(x,y) = \begin{cases} 1 & \text{if } (x,y) \in \{o_k\}_{k=1}^{K}, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Here $o_k = (x_k^o, y_k^o)$ indicates the spatial coordinates of the center of *k-th* cell instance, computed by $(\text{median}(\{x_k\}), \text{median}(\{y_k\}))$. $\{x_k\}$ and $\{y_k\}$ are the x- and y-axis coordinates of $S_k^{(0)}$. Thus $C^{(0)}$ gathers all the instance centers.

The initial label $V^{(0)} \in \mathbb{R}^{2\times W\times H}$ for the offset map is given as:

$$V^{(0)}(x,y) = \begin{cases} (x - x_{k_p}^o, y - y_{k_p}^o) & \text{if } M^{(0)}(x,y) = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

Here $o_{k_p} = (x_{k_p}^o, y_{k_p}^o)$ denotes the center position of the cell instance k_p that the cell pixel p belongs to. Thus, for each cell pixel p within a cell instance, $V^{(0)}(p)$ stores the offset from $p = (x,y)$ to its corresponding center $o_{k_p} = (x_{k_p}^o, y_{k_p}^o)$.

Recursive Training Label optimization

As mentioned in §Methods Overview, we adopt a recursive training strategy (*cf.* Eq. 1) to mitigate the negative influence of noise in training labels. At each recursive training round $t = \{0,1,2,\cdots\}$, after the converge of the network training (*cf.* Eq. 2), we use the updated network parameter $\theta^{(t)}$ to make predictions over the training data $(\boldsymbol{I}, \boldsymbol{G})$, *i.e.*, $\hat{\boldsymbol{Y}}^{(t+1)} = \{(\hat{M}_n^{(t+1)}, \hat{C}_n^{(t+1)}, \hat{V}_n^{(t+1)})\}_{n=1}^{N} = \{f(I_n, G_n; \theta^{(t+1)})\}_{n=1}^{N}$, and a new set of K' cell instance masks $\{\hat{S}_{k'}^{(t+1)} \in \{0,1\}^{W\times H}\}_{k'=1}^{K'}$ is further derived for each training sample (I,G). Then, for each mask $\hat{S}_{k'}^{(t+1)}$, we apply the level-set algorithm[22] for further boundary refinement, which is given by:

$$Q(C, \hat{S}_{k'}^{(t+1)}) = \nu \cdot \mathbb{L}(C) + \lambda_1 \sum_{(x,y)\in\mathscr{O}(C)} \left| \hat{\mathbf{S}}_{k'}^{(t+1)}(x,y) - \frac{\sum_{(x,y)\in\mathscr{O}(C)} \hat{\mathbf{S}}_{k'}^{(t+1)}(x,y)}{|\mathscr{O}(C)|} \right|^2 + \lambda_2 \sum_{(x,y)\in\mathscr{I}(C)} \left| \hat{\mathbf{S}}_{k'}^{(t+1)}(x,y) - \frac{\sum_{(x,y)\in\mathscr{I}(C)} \hat{\mathbf{S}}_{k'}^{(t+1)}(x,y)}{|\mathscr{I}(C)|} \right|^2, \tag{13}$$

where $Q(\cdot)$ denotes the level set operation and $|\cdot|$ indicates the cardinality of a set. C represents the evolving contour of the k'*-th* cell instance mask $\hat{S}_{k'}^{(t+1)}$ at $(t+1)$*-th* recursive training round. $\mathbb{L}(C)$ denotes the length of the curve C. $\mathscr{O}(C)$ and $\mathscr{I}(C)$ represent the regions outside and inside the contour C. ν, λ_1 and λ_2 are modulating parameters and are all set to be 1 in our study. Improved cell boundaries are obtained by minimizing the level set operation $Q(\cdot)$.

The initial contour of the cell instance mask $\hat{S}_{k'}^{(t+1)}$ is defined implicitly via a Lipschitz function φ, by $C_0 = \{(x_c, y_c) \mid \varphi(x_c, y_c) = 0\}$, which denotes the set of all contour points of the function $\varphi(z_c, x_c, y_c) = 0$ at time $z_c = 0$, meaning zero level set. Therefore, the evolution of contour C can be carried out by gradually changing the value of the level set function $\varphi(\cdot)$ over time, which is equivalent to solving the following differential equation in the normal direction at each time:

$$\frac{\partial \varphi}{\partial t} + V|\nabla\varphi| = 0, \quad V = div(\frac{\nabla\varphi}{|\nabla\varphi|}), \tag{14}$$

where V is the curvature of the function $\varphi(\cdot)$ passing through (x_z, y_z)[22,27].

As the level-set algorithm takes into account the object's geometric properties, such as curvature and gradient, it improves the convexity and elongation of the cell contour. The refined training labels $\{S_{k'}^{(t+1)}\}_{k'=1}^{K'}$ are used to create refined confidence map, center map, and offset map, *i.e.*, $\boldsymbol{Y}^{(t+1)} = \{(M_n^{(t+1)} \in \{0,1\}^{W\times H}, C_n^{(t+1)} \in \{0,1\}^{W\times H}, V_n^{(t+1)} \in \mathbb{R}^{2\times W\times H})\}_{n=1}^{N}$, analogous to Eqs. 10, 11, and 12. The newly created training labels $\boldsymbol{Y}^{(t+1)}$ are used for the further optimization of the network parameter.

Training Loss

Given current training labels $\boldsymbol{Y}^{(t)} = \{(M_n^{(t)}, C_n^{(t)}, V_n^{(t)})\}_{n=1}^{N}$ and network outputs $\hat{\boldsymbol{Y}}^{(t)} = \{(\hat{M}_n^{(t)}, \hat{C}_n^{(t)}, \hat{V}_n^{(t)})\}_{n=1}^{N} = \{f(I_n, G_n; \theta^{(t)})\}_{n=1}^{N}$ on the training data $\{I_n, G_n\}_{n=1}^{N}$, our training loss $\mathscr{L}$ in Eq. 2 is defined as:

$$\mathscr{L}(\boldsymbol{Y}^{(t)}, \hat{\boldsymbol{Y}}^{(t)}) = \frac{1}{N}\sum_{n=1}^{N} \lambda_1 \mathscr{L}_{ce}(M_n^{(t)}, \hat{M}_n^{(t)}) + \frac{1}{N}\sum_{n=1}^{N} \lambda_2 \mathscr{L}_{l2}(C_n^{(t)}, \hat{C}_n^{(t)}) + \frac{1}{N}\sum_{n=1}^{N} \lambda_3 \mathscr{L}_{l2}(V_n^{(t)}, \hat{V}_n^{(t)}), \tag{15}$$

Algorithm 1 Recursive Training Procedure of GeneSegNet

Require: Training images $\boldsymbol{I} = \{I_n\}_{n=1}^{N}$, RNA location maps $\boldsymbol{G} = \{G_n\}_{n=1}^{N}$ (Eq. 5),
Initial labels $\boldsymbol{Y}^{(0)} = \{(M_n^{(0)}, C_n^{(0)}, V_n^{(0)})\}$ (Eqs. 10, 11, and 12);
Ensure: Optimal network parameter θ of GeneSegNet network f;

for $t = 0, 1, 2, \cdots$ **do**
  / update θ with fixed $\boldsymbol{Y}$ in Eq. 2 */* ▷ See §Training Details for details
  $\theta^{(t+1)} \leftarrow \arg\min_{\theta} \mathscr{L}(\boldsymbol{Y}^{(t)}, \{f(I_n, G_n, \theta)\}_{n=1}^{N})$
  / update $\boldsymbol{Y}$ with fixed θ in Eq. 3 */* ▷ See §Recursive Training Label optimization for details
  for $n = 1, 2, \cdots, N$ **do**
    $\hat{M}_n^{(t+1)}, \hat{C}_n^{(t+1)}, \hat{V}_n^{(t+1)} = f(I_n, G_n; \theta^{(t+1)})$
  end for
  $\hat{\boldsymbol{Y}}^{(t+1)} = \{(\hat{M}_n^{(t+1)}, \hat{C}_n^{(t+1)}, \hat{V}_n^{(t+1)})\}_{n=1}^{N}$
  $\boldsymbol{Y}^{(t+1)} = \text{level-set}(\hat{\boldsymbol{Y}}^{(t+1)})$
  / terminate the recursive training */*
  if $\text{IoU}(\hat{\boldsymbol{Y}}^{(t)}, \hat{\boldsymbol{Y}}^{(t+1)}) > 0.93$ **then**
    return
  end if
end for
Return: Network parameter θ in the last training round

where $\mathscr{L}_{ce}$ and $\mathscr{L}_{l2}$ demonstrate the standard cross-entropy loss and L2 loss, respectively, which are applied in a position-wise manner; the coefficients are empirically set as $\lambda_1 = 1$, $\lambda_2 = 1$, and $\lambda_3 = 1$, to balance the relative contributions among different training terms. Alg. 1 provides the pseudo code for the recursive network training strategy.

Training Details

The proposed GeneSegNet is trained in an recursive manner. In each recursive stage, the network f is trained for 500 epochs using the AdamW optimizer[21], with a batch size of 8, an initial learning rate of 0.001, a momentum of 0.9, and a weight decay of 1e-5. We use standard data augmentation techniques, including random scale jittering with a factor in $[0.75, 1.25]$, random horizontal flipping, random cropping, and random rotation. The recursive training procedure will be terminated when the IoU score, computed between $\{S_k^{(t)}\}_k$ and $\{S_{k'}^{(t+1)}\}_{k'}$, is larger than 0.93, averaged over all the cell instances in $\{S_k^{(t)}\}_k$. The IoU score for *k-th* cell instance $S_k^{(t)}$ is computed by:

$$IoU_k = \max(\{\frac{|S_k^{(t)} \cap S_{k'}^{(t+1)}|}{|S_k^{(t)} \cup S_{k'}^{(t+1)}|}\}_{k'=1}^{K'}), \tag{16}$$

where $|\cdot|$ denotes the Jaccard index, and $\cap$ and $\cup$ represents the area of overlap and union between cell instances, respectively. In addition, to effectively verify the training performance of GeneSegNet, we also present training evaluations for each recursive training round (Supplementary Fig. **S**1). GeneSegNet is implemented using PyTorch 1.11.0.

Network Inference

Following the common practice[10] in this field, we divide the raw image into overlapping patches of the size of 256×256 pixels, with their corresponding RNA location maps. This allows GeneSegNet to handle images of arbitrary resolutions. These patches are overlapped both in vertical and horizontal directions by 50%, and fed into GeneSegNet individually for cell mask inference (*cf.* Eq. 4). Thus each pixel in the original image are processed four times. The final full-resolution segmentation is obtained through averaging the four predictions (in the coordinate system of the original image) for every pixel.

Evaluation Metrics

Cell calling

Let $\boldsymbol{g}_{\text{in}} = \{g_l : \hat{M}(g_l) \geq 0.5\}$ and $\boldsymbol{g}_{\text{out}} = \{g_l : \hat{M}(g_l) < 0.5, D(g_l) \leq \delta\}$ denote the sets of RNAs located inside or within a neighborhood outside predicted cell regions, respectively. $D(g_l)$ is the distance between g_l and the nearest pixel that belongs to a cell boundary. δ is a threshold and takes values of 3, 5, or 7. The cell calling metric is calculated as $\frac{|\boldsymbol{g}_{\text{in}}|}{|\boldsymbol{g}_{\text{out}}|}$, where $|\cdot|$ denotes the cardinality of a set. A higher cell calling means the method has a stronger ability to recruit RNAs inside cell boundaries.

Cell convexity

For each cell instance with boundary perimeter r_{cell}, we define its convex hull as the smallest possible convex shape that completely contains the cell region and denote the perimeter of the convex hull as r_{convex}. Then, the cell convexity of each cell is computed as $\frac{r_{\text{convex}}}{r_{\text{cell}}}$.

Cell elongation

Let $\boldsymbol{B}$ be a set of two-dimensional points representing the boundary of a cell. We compute the covariance matrix of the $\boldsymbol{B}$, and then compute the two eigenvalues e_a and e_b of the covariance matrix, where $e_a > e_b$. Cell elongation is computed as $\frac{e_b}{e_a}$.

Cell area

For each detected cell instance, its cell area is defined as the number of pixels it contains.

Gene-gene correlation reproducibility

Gene-gene correlation was calculated separately within each of the 28 mouse hippocampus samples and 2 NSCLC samples. For each hippocampus sample, segmented cells with positive expression in at least 5 genes were retained. For each NSCLC sample, segmented cells with positive expression in at least 30 genes were retained. Then for both hippocampus and NSCLC samples, the read count of each cell was divided by the total number of reads of that cell to produce normalized gene expression values. Genes with positive expression in at least 5% of cells were retained. Pearson correlation coefficient of normalized gene expression was calculated for each pair of genes across all cells.

For each pair of hippocampus samples from the same tissue (e.g., left hippocampus of the first mouse), a Pearson correlation is calculated across the gene-gene correlations of all pairs of genes. The reproducibility is calculated as the median of correlations across all pairs of samples from the same tissue. For NSCLC dataset, the reproducibility is calculated as the Pearson correlation across gene-gene correlations of all pairs of genes between the two NSCLC samples.

Competing Methods

We compare GeneSegNet against five competing methods: Watershed algorithm[6,7], Cellpose[10], JSTA[12], Baysor[11], and Baysor(prior)[11]. The implementation details are provided as follows.

Watershed Algorithm

For the hippocampus dataset, we directly obtained the segmentation results by the Watershed algorithm from the original publication[19]. For the NSCLC dataset, we implemented the Watershed algorithm using the nucleus images based on *scikit-image*[28] and *SciPy*[29] python packages. Specifically, we segment each image by (1) performing morphology erosion on the image to enable separations of cell instances that are adhesive together, (2) computing the distance map of the image and generating a set of markers as local maxima of the distance to the background, and (3) feeding the image and markers into the watershed algorithm in *scikit-image* for segmentation.

Cellpose Algorithm

Cellpose was performed with the default settings[10]. For the hippocampus dataset, Cellpose was trained using the DAPI nucleus staining images. For the NSCLC dataset, Cellpose segmentation with both nucleus and cytoplasm images was obtained directly from the NanoString website. In addition, we also performed Cellpose with only the nucleus images using the default settings.

JSTA Algorithm

JSTA was performed with the default settings[12] for both datasets. Spatial locations of cell centers and cell types of individual cells were directly downloaded from the original publication or NanoString website.

Baysor Algorithm

Baysor was performed with the default settings[12] for both datasets. In this mode, Baysor was performed without using imaging or prior segmentation information.

Baysor(prior) Algorithm

Baysor was performed with the default settings[12] for both datasets. In this mode, Baysor was performed with prior segmentation information. Specifically, for the hippocampus dataset, Watershed segmentation results were used as prior segmentation information. For the NSCLC dataset, Cellpose segmentation results with both nucleus and cytoplasm images were used as prior segmentation information.

Data and code availability

The raw data of hippocampus and NSCLC datasets were downloaded from the original publication or from NanoString website. Source code of GeneSegNet is available on GitHub: https://github.com/BoomStarcuc/GeneSegNet.

Acknowledgements

We thank Drs. Kenneth D. Harris, Dimitris Nicoloutsopoulos, and Mats Nilsson for providing the mouse hippocampus dataset and its annotations. We also thank NanoString company for providing the NSCLC dataset. Z.J. was supported by the Whitehead Scholars Program at Duke University School of Medicine.

Author contributions

Z.J. conceived the study. W.W., Y.W., D.L., T.Z., and Z.J. designed the algorithm and experiments. Y.W. performed all experiments with the aid of T.Z.. Y.W., W.H., and Z.J. analyzed the experiment results. All authors contributed to the writing of the manuscript.

Competing interests

None declared.

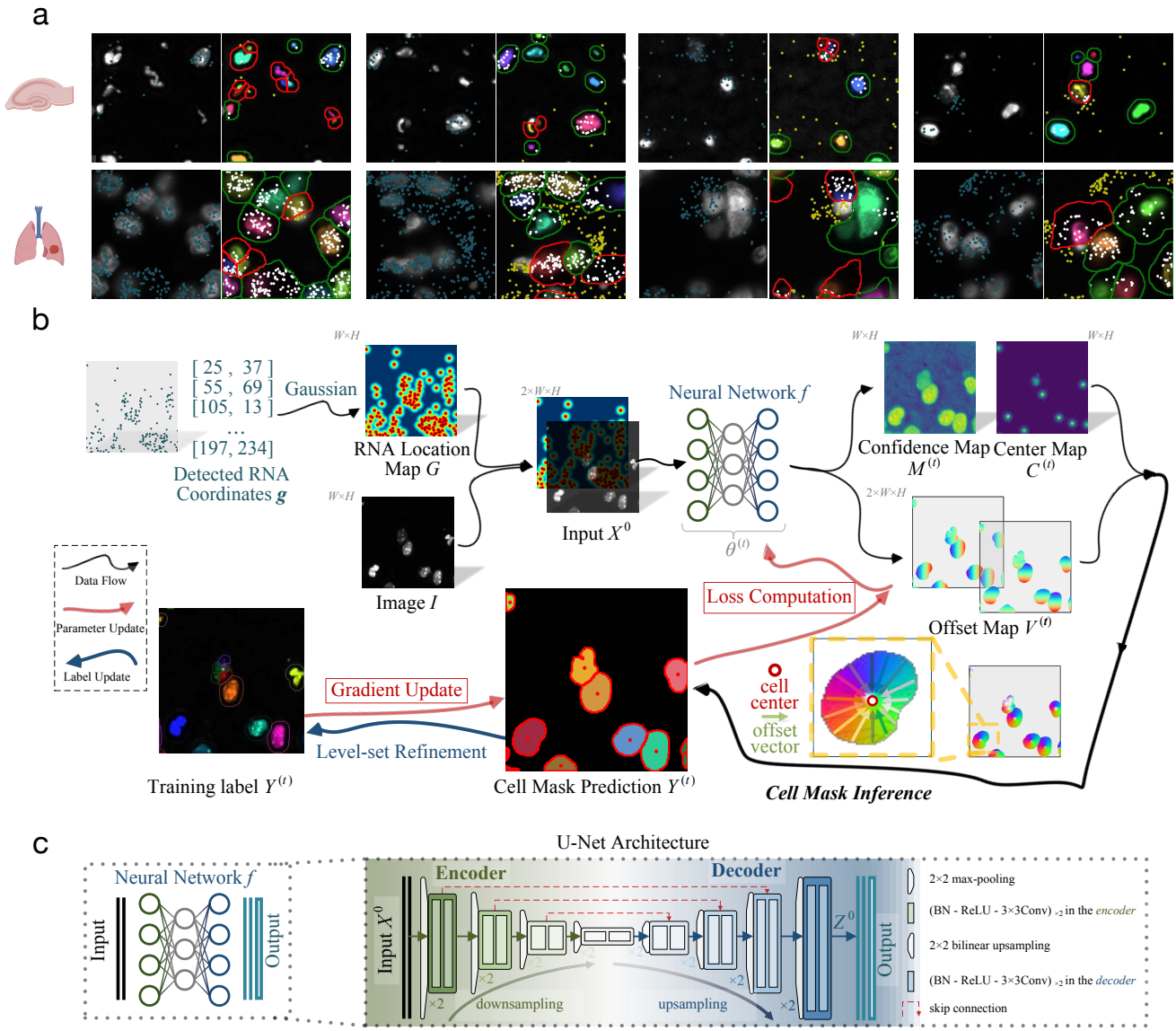

Figure 1. a, Examples of noisy training label annotations in hippocampus and NSCLC datasets. Four examples are given for each dataset. The left image in each example shows the image with detected RNAs (marked in blue). The right image indicates the corresponding annotations, where the potential noisy annotations are marked in red. RNAs falling inside annotated cell regions are marked in white and RNAs falling outside annotated cell regions are marked in yellow. **b**, Overview of GeneSegNet framework. GeneSegNet makes a joint use of gene spatial coordinates and imaging information for cell segmentation, and is recursively learned by alternating between the optimization of network parameters and estimation of training labels for noise-tolerant training. The outputs of GeneSegNet include a confidence map $\hat{M} \in [0,1]^{w \times h}$ that stores the probability of each pixel being inside cell regions, a center map $\hat{C} \in [0,1]^{w \times h}$ that gives the likelihood of each pixel being a cell center, and a two-channel offset map $\hat{V} \in \mathbb{R}^{2 \times w \times h}$ indicating the offset vector of each pixel to the center of its corresponding cell instance. Combining the outputs together, GeneSegNet can recover the precise boundary of each individual cell. **c**, GeneSegNet is built upon the general U-Net network architecture, where an encoder progressive downsamples the inputs for more expressive feature extraction, and a decoder upsamples the feature maps in a mirror-symmetric fashion for fine-grained cell segmentation.

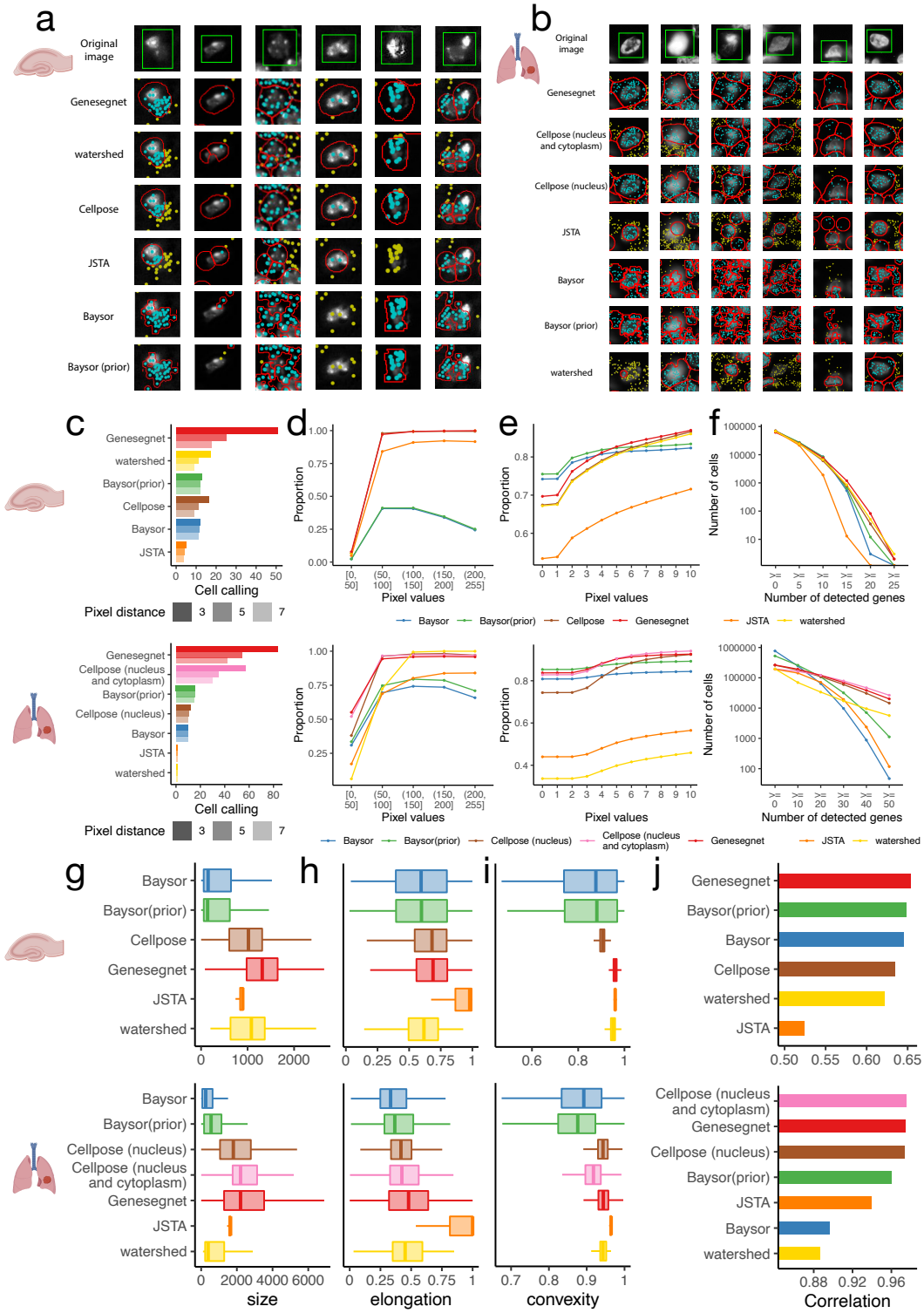

Figure 2. a-b, Examples of cell segmentations in hippocampus dataset (**a**) and NSCLC dataset (**b**). Each row is a method. The first row shows the original image. The cell of interest is highlighted in a green window. Each column is an example. The detected cell boundaries are marked in red. Light blue dots represent RNAs falling within cell boundaries. Yellow dots represent RNAs falling outside of cell boundaries. **c**, Cell calling metrics in hippocampus (top) and NSCLC (bottom) datasets. **d**, Proportion of pixels within cell boundaries (y-axis) for pixels with different levels of pixel values (x-axis), in hippocampus (top) and NSCLC (bottom) datasets. **e**, Proportion of RNA reads within cell boundaries (y-axis) for RNA reads with different pixel values (x-axis), in hippocampus (top) and NSCLC (bottom) datasets. **f**, Number of cells (y-axis) with different numbers of detected genes (x-axis), in hippocampus (top) and NSCLC (bottom) datasets. A gene is detected if it has at least one read in that cell. **g-i**, The distribution of cell size in pixels (**g**), elongation (**h**), and convexity (**i**), in hippocampus (top) and NSCLC (bottom) datasets. **j**, Reproducibility of gene-gene correlations across biological replicates.

References

1. Crosetto, N., Bienko, M. & Van Oudenaarden, A. Spatially resolved transcriptomics and beyond. *Nat. Rev. Genet.* **16**, 57–66 (2015).
2. Chen, K. H., Boettiger, A. N., Moffitt, J. R., Wang, S. & Zhuang, X. Spatially resolved, highly multiplexed rna profiling in single cells. *Science* **348**, aaa6090 (2015).
3. Femino, A. M., Fay, F. S., Fogarty, K. & Singer, R. H. Visualization of single rna transcripts in situ. *Science* **280**, 585–590 (1998).
4. Eng, C.-H. L. *et al.* Transcriptome-scale super-resolved imaging in tissues by rna seqfish+. *Nature* **568**, 235–239 (2019).
5. Wang, F. *et al.* Rnascope: a novel in situ rna analysis platform for formalin-fixed, paraffin-embedded tissues. *The J. molecular diagnostics* **14**, 22–29 (2012).
6. Roerdink, J. B. & Meijster, A. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta informaticae* **41**, 187–228 (2000).
7. Gamarra, M., Zurek, E., Escalante, H. J., Hurtado, L. & San-Juan-Vergara, H. Split and merge watershed: A two-step method for cell segmentation in fluorescence microscopy images. *Biomed. signal processing control* **53**, 101575 (2019).
8. 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*, 234–241 (2015).
9. Van Valen, D. A. *et al.* Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. *PLoS computational biology* **12**, e1005177 (2016).
10. Stringer, C., Wang, T., Michaelos, M. & Pachitariu, M. Cellpose: a generalist algorithm for cellular segmentation. *Nat. Methods* **18**, 100–106 (2021).
11. Petukhov, V. *et al.* Cell segmentation in imaging-based spatial transcriptomics. *Nat. Biotechnol.* **40**, 345–354 (2022).
12. Littman, R. *et al.* Joint cell segmentation and cell type annotation for spatial transcriptomics. *Mol. Syst. Biol.* **17**, e10108 (2021).
13. Park, J. *et al.* Cell segmentation-free inference of cell types from in situ transcriptomics data. *Nat. communications* **12**, 1–13 (2021).
14. Alom, M. Z., Hasan, M., Yakopcic, C., Taha, T. M. & Asari, V. K. Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. *arXiv preprint arXiv:1802.06955* (2018).
15. Hofmanninger, J. *et al.* Automatic lung segmentation in routine imaging is primarily a data diversity problem, not a methodology problem. *Eur. Radiol. Exp.* **4**, 1–13 (2020).
16. Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM* **64**, 107–115 (2021).
17. Reed, S. *et al.* Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596* (2014).
18. Arpit, D. *et al.* A closer look at memorization in deep networks. In *International Conference on Machine Learning*, 233–242 (2017).
19. Qian, X. *et al.* Probabilistic cell typing enables fine mapping of closely related cell types in situ. *Nat. Methods* **17**, 101–106 (2020).
20. He, S. *et al.* High-plex multiomic analysis in ffpe at subcellular level by spatial molecular imaging. *bioRxiv preprint* (2022).
21. Loshchilov, I. & Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).
22. Osher, S. & Sethian, J. A. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.* **79**, 12–49 (1988).
23. Li, C., Xu, C., Gui, C. & Fox, M. D. Distance regularized level set evolution and its application to image segmentation. *IEEE Transactions on Image Process.* **19**, 3243–3254 (2010).
24. Greenwald, N. F. *et al.* Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning. *Nat. Biotechnol.* **40**, 555–565 (2022).
25. Cao, Z., Simon, T., Wei, S.-E. & Sheikh, Y. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7291–7299 (2017).

26. 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*, 770–778 (2016).

27. Chan, T. F. & Vese, L. A. Active contours without edges. *IEEE Transactions on Image Process.* **10**, 266–277 (2001).

28. van der Walt, S. *et al.* scikit-image: image processing in Python. *PeerJ* **2**, e453 (2014). URL https://doi.org/10.7717/peerj.453. DOI 10.7717/peerj.453.

29. Virtanen, P. *et al.* SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat. Methods* **17**, 261–272 (2020). DOI 10.1038/s41592-019-0686-2.