

Supplementary Information

1 Sample preparation

Albino inbred mice (FVB/NJ, strain #001800), line M mice Tg(Thy1-EGFP)MJrs/J, C57BL/6J background (strain #007788-JAX), were obtained from the Jackson Laboratory (Bar Harbor, ME, USA). These mice were established and maintained in a vivarium, with four to five mice per cage and a 12 h light/dark cycle in the animal facility. Food and water were available ad libitum. All studies correspond to the principles of humane treatment of animals and were approved by the Bioethics Committee of Peter the Great St. Petersburg Polytechnic University in St. Petersburg, Russia, and also followed the principles of European convention (Strasbourg, 1986) and the Declaration of the International Medical Association for the Humane Treatment of Animals (Helsinki, 1996). All methods were carried out in accordance with relevant guidelines and regulations. The study was carried out in compliance with the ARRIVE guidelines.

Primary hippocampal neuronal cultures of dissociated hippocampal cells were prepared from newborn Albino inbred mice (FVB/NJ) and maintained in culture as described previously [1]. All used in experiments P0 mice pups were euthanized by decapitation with sharp surgical scissors. Briefly, the hippocampus from postnatal day 0–1 pups was incubated with papain solution for 30 min at 37 °C (Worthington Biochemical Corp., Lakewood, NJ, USA, #3176), then dissociated with a solution of deoxyribonuclease I (5 mg/mL; Macherey Nagel GMBH, Germany, #R1542S). The neurons were placed in a 24-well plate on 12 mm glass coverslips pre-coated with 1% poly-D-lysine (Sigma-Aldrich, St Louis, MO, USA, #P7886). Cells were grown in 1 mL Neurobasal-A (Thermo Fisher Scientific, Waltham, MA, USA, #10888022) supplemented with 2% B27 (Thermo Fisher Scientific, Waltham, MA, USA, #17504044), 1% heat-inactivated fetal bovine serum (FBS, Thermo Fisher Scientific, Waltham, MA, USA, #10500064), and 0.5 mM L-glutamine (Thermo Fisher Scientific, Waltham, MA, USA, #25030024), and were maintained at 37 °C in a 5% CO₂ incubator in a 24-well glass plate. Transfection was performed according to [2] with a calcium transfection kit purchased from Clontech (Takara Bio, Kusatsu, Japan, #631312) using the pLV-eGFP plasmid purchased from the Addgene plasmid database (Addgene, USA, #36083). Preparation of the oligomeric beta-amyloid was described in [3]. Cells were incubated with A β at a final concentration of 1 μ M for 72 h before fixation. At DIV16–17, hippocampal primary neurons were fixed in 4% paraformaldehyde (PFA) (Sigma-Aldrich, #158127) and 4% sucrose in PBS (pH 7.3). Fixed cells were stained for GFP using primary anti-GFP antibodies (Invitrogen, cat. no. A-11122) followed by Alexa Fluor 488-conjugated secondary antibodies (anti-rabbit, Invitrogen, cat. no. R37116; diluted 2 drops per mL) to enhance the fluorescence signal.

For all model training and cross-validation procedures, five image stacks of primary hippocampal culture dendrites (including two image stacks of neurons under low amyloid toxicity) were captured with a Leica TCS SP8 microscope (0.022 μ m/pixel resolution, Z interval of 0.1 μ m), and four image stacks obtained from hippocampal cultures were captured with a Thorlabs confocal microscope (USA) (0.022 μ m/pixel resolution, Z interval of 0.2 μ m). For the comparison dataset, Z-stack images were captured with a Leica TCS SP8 microscope (0.022 μ m/pixel resolution, Z interval of 0.1 μ m).

To obtain dendritic spines from brain slices of mice, line M mice Tg (Thy1-EGFP) MJrs/J was used as in details described in [1]. Mice were anesthetized via urethane (200 L of a 250 mg/mL solution) diluted in a 0.9% NaCl solution. Briefly, after anesthesia and transcardiac

perfusion brain of mice were placed in a 4% PFA solution for post-fixation for 1 week. Then, slices with a thickness of 40–50 μm were prepared. The slices were incubated for 16 h with primary monoclonal antibodies targeting GFP proteins (1:200 dilution, Invitrogen, Waltham, MA, USA, Cat. No. MA5-32977). The samples were stained with Alexa-Fluor 488 (2 drops per mL, anti-rabbit, Invitrogen, Cat. No. R37116) secondary antibodies for 2 h.

For all models training and cross-validation process, six image stacks of hippocampal dendrites (0.0189 μm /pixel resolution, Z interval of 0.2 μm) were captured with Leica TCS SP8.

All image stacks are 2048 $\times$ 2048-pixel size and were captured with 100 $\times$ objective lens (NA=1.4, UPlanSApo; Olympus, Tokyo, Japan).

2 Segmentation workflow and evaluation methods

2.1 Segmentation pipelines

To compare segmentation quality, segmentation was performed using DeepD3 [4] and RESPAN [5]. For DeepD3, the recommended pre-trained model `DeepD3_32F.h5` (32 base filters, original resolution) was used from published model zoo. DeepD3 does not support fine-tuning of the released weights, so the model was applied without adaptation to our data. RESPAN provides tools for model fine-tuning, but retraining was unsuccessful in our hands, the pre-trained Model 1A was therefore used with all parameters set as specified in the video tutorial provided by the developers. The resulting masks were taken from the Validation folder. Prediction masks were binarized in ImageJ for DeepD3 and RESPAN. The threshold was selected individually for each image stack to maximize the foreground Dice coefficient. In contrast, for SpineTool 2.0 was used built-in threshold of 0.5 without tuning.

Segmentation of dendritic spines using the VSOT [6] algorithm was performed for two purposes. The first was to obtain reference results for evaluating the accuracy of the new algorithm, segmentation was performed on ground-truth, manually annotated dendrites. The second was as part of the proposed segmentation pipeline, to prepare data for the fourth stage of the neural network pipeline.

The program code from an open-source repository was used to run the algorithm. the VSOT code was called non-interactively from Python using Oct2Py and GNU Octave 11.1.0 [7]. Segmentation was run by calling the VSOT function `level_1_segmentation_function_fin`.

The input to VSOT was the binary three-dimensional dendrite mask. A dendrite centreline was extracted from the binary mask with the TEASAR implementation in Kimimaro [8]. Physical voxel dimensions were supplied as the anisotropy parameter. Cycles in the skeleton graph were removed by retaining its minimum spanning tree, and terminal branches shorter than ~ 6000 nm were iteratively pruned. Branch points were removed to obtain individual shaft segments. These segments were propagated over the dendrite mask by marker-controlled watershed of the negative anisotropic Euclidean distance transform, producing a labelled segment volume.

Each labelled segment was processed within its bounding box. Its binary mask, pruned skeleton, and anisotropic Euclidean distance transform were converted to floating point and transposed from (Z, Y, X) to (X, Y, Z) . The parameters passed to `level_1_segmentation_function_fin` were the voxel sizes `res_x`, `res_y`, and `res_z` in nanometres. Segments exceeding 600 voxels in any dimension were divided into overlapping chunks with a 400-voxel step and 200-voxel margins; chunks containing at most 50 skeleton voxels were skipped. Overlapping predictions were merged by logical union. The VSOT output was thresholded at zero to obtain the shaft mask, transposed back to (Z, Y, X) , and saved as an 8-bit binary TIFF. The spine mask was obtained by subtracting the predicted shaft from the input dendrite mask: every foreground voxel of the input dendrite that was not classified as shaft was assigned to the spine class,

$$M_{\text{spine}} = M_{\text{dendrite}} \cap \overline{M_{\text{shaft}}}.$$

Thus, background voxels outside the input dendrite could not be included in the spine mask. For the subsequent neural-network stage, both masks were combined into a min-max-normalized channel (background 0, shaft 1, and spines 2), supplemented with normalized distances to the shaft skeleton and estimated spine axes.

2.2 Segmentation Performance Evaluation

Segmentation was evaluated at two levels: at the voxel level- measures how closely the predicted volume overlaps the annotation, and at the object level- measures how many individual spines were detected [9]

Voxel-level metrics Predicted masks were compared with the ground-truth annotations voxel by voxel within the annotated region of interest. Each voxel was assigned to one of three classes- background, dendritic shaft and dendritic spine and metrics were computed one-versus-rest for the class of interest, where TP , FP and FN are the numbers of correctly assigned, falsely assigned and missed voxels of that class:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad (1)$$

$$\text{IoU} = \frac{TP}{TP + FP + FN}, \quad \text{Dice} = \frac{2TP}{2TP + FP + FN}. \quad (2)$$

Object-level metrics An annotated spine was counted as a true positive when its best IoU with a predicted component reached 0.5; unmatched predictions were counted as false positives and unmatched annotations as false negatives. Precision and recall follow the definitions above applied to these object counts, and

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (3)$$

The F_1 -score and the Dice coefficient differ in the unit of analysis, spine objects and voxels respectively.

All metrics were computed per image stack and are reported as mean $\pm$ standard deviation across stacks.

2.3 Cross-validation

To evaluate the accuracy of the proposed segmentation models on a limited dataset and to determine the algorithm’s generalization capability, cross-validation [10] was used to divide the datasets into subsets. At each stage of the segmentation pipeline, the test, validation, and training datasets were divided into 5 groups, and a separate U-Net model was trained independently for each group. Accuracy metrics were measured on a test subset of the data.

The process of dividing the sample into subsamples was designed so that each manually labeled z-stack prepared in the laboratory was included in the test set of one of the cross-validation models. Thus, the five resulting models predicted the outcome for each of the test samples, and the average accuracy was used as an accuracy measure for the proposed dendritic spine segmentation architecture. This accuracy was compared with the segmentation accuracy of the DeepD3, Respan, and VSOT models.

The accuracy metrics obtained for each model in the five-fold cross-validation are reported in Table S1. Each fold contained three held-out test images:

- Fold 1: in vitro 2, in vitro 4, and ex vivo 14;
- Fold 2: in vitro 1, in vitro 3, and ex vivo 10;

Table S1: Cross-validation accuracy metrics for the four model stages.

Stage 1					
Fold	IoU	Dice			
1	0.6732	0.8031			
2	0.7130	0.8318			
3	0.6385	0.7759			
4	0.5726	0.7195			
5	0.6026	0.7509			
Stage 2					
Fold	IoU	Dice			
1	0.6192	0.7648			
2	0.7102	0.8299			
3	0.6418	0.7790			
4	0.6247	0.7658			
5	0.6208	0.7655			
Stage 3					
Fold	IoU	Dice			
1	0.7072	0.8277			
2	0.6910	0.8163			
3	0.6975	0.8177			
4	0.5385	0.6905			
5	0.6392	0.7795			
Stage 4					
Fold	Voxels		Spines		
	IoU	Dice	F1	Recall	Precision
1	0.7152	0.8315	0.5613	0.7183	0.4613
2	0.7208	0.8377	0.6156	0.7321	0.5346
3	0.6379	0.7769	0.5879	0.6944	0.5303
4	0.6621	0.7946	0.6119	0.7457	0.5404
5	0.6079	0.7541	0.4276	0.6092	0.3424

- Fold 3: in vitro 5, in vitro 10, and ex vivo 5;
- Fold 4: in vitro 6, in vitro 9, and ex vivo 16;
- Fold 5: in vitro 8, ex vivo 12, and ex vivo 18.

The training, validation, and test partition was generated once for each fold and kept identical across all four segmentation stages. This constraint is required because the models in subsequent stages rely on predictions from the models in previous stages. To conduct a clean accuracy experiment on the test data, we must ensure that no stage in the pipeline has ever seen any part of the test image during training or fine-tuning. Training and validation loss curves, together with prediction accuracy, are shown in Figure S2C.

The distributed SpineTool2 segmentation software is supplied with the model weights for Fold 2, as they demonstrated the best overall accuracy among the evaluated model sets from other folds.

2.4 Comparison with existing software

A comparative analysis of existing software solutions demonstrates a broad spectrum of architectural frameworks, image processing workflows, and functional capabilities for dendritic spine analysis. Most tools are implemented as standalone applications in Python, C++, or MATLAB, or as ImageJ/Fiji plugins [11], primarily processing 2D/3D TIFF stacks and projections. Segmentation methodologies show a clear transition from classical thresholding and watershed algorithms (SpineJ [12], 2dSpAn-Auto [13], Spot Spine [14]) toward deep neural network architectures (RESPAN [5], DeepD3 [4], VG-LAB [15]). User intervention varies significantly, ranging from fully automated pipelines with no manual editing (VSOT [6], 2dSpAn-Auto) to highly interactive platforms featuring manual voxel and mesh curation (VG-LAB, Braintown [14], Spot Spine). Overall, existing tools reflect a trade-off between AI-driven automation and manual curation precision, whereas SpineTool 2 bridges these approaches by combining full-frame neural segmentation of complex dendritic branching with high-precision 3D reconstruction.

Table S2: Comparison of software tools and platforms for dendritic spine analysis and morphometrics.

Tool	Platform / GUI & Input Data	Processing Stages & Tunable Parameters	Manual Annotation / Correction & Output	Core Functionality
RESPAN	Standalone Windows (.exe), Python; 2D/3D TIFF stacks	4 stages: Restoration, Axial enhancement, Segmentation (nnU-Net), Stats; <i>Tunable</i> : Resolution, pre-trained models, spine neck thresholds, GPU selection, model fine-tuning	Fine-tuning via manually corrected data; dedicated Validation tab	Restoration, multi-compartment spine segmentation, tracking, and quantitative metrics.
DeepD3	Standalone GUI (Python/PyQt); TIF stacks	3 stages: Dendrite/spine segmentation, prediction cleaning, 2D/3D ROI construction; <i>Tunable</i> : Real-time 3D ROI segmentation hyperparameters	Interactive ROI tuning and cleaning; Exports to TIFF, ImageJ ROIs, and text centroids	Automated semantic segmentation of dendrites and spines trained on multi-expert data.
VSOT	Code / CLI; Ocean Pre-processed EM reconstructions	2-level segmentation (shaft/spine, head/neck), quantitative analysis; <i>Tunable</i> : Code/config optimization parameters (surface vs. volume metrics)	No manual correction; <i>Output</i> : Compartmental segmentation and morphometrics	Precise compartmental spine segmentation (spine, head, neck).

Continued on next page

Table S2: Comparison of software tools and platforms for dendritic spine analysis and morphometrics.

Tool	Platform / GUI & Input Data	Processing Stages & Tunable Parameters	Manual Annotation / Correction & Output	Core Functionality
Smirnov et al. / Braintown	MATLAB desktop GUI; Single images or 2-photon Z-stacks	3D projection, drift correction, plugin pipeline; <i>Tunable:</i> Scale (pixels/ μm), saveable/shareable plugin configs	Manual ROI drawing, click-to-select spines, brightness tracking, model training; <i>Output:</i> Temporal volume changes, dataset exports	Real-time automated spine identification for live imaging (> 90% accuracy).
2dSpAn-Auto	Desktop app (Windows, C++); 2D MIP images	3 modules: Denoising/binarization, skeletonization, morphometry; <i>Tunable:</i> Denoising/binarization selection, blob thresholds, skeletal path criteria	Fully automated batch processing without point-and-click; <i>Output:</i> Metrics tables, global stats, color-coded visualisations	Automated dendrite/spine segmentation and morphometry (binary/fuzzy modes).
VG-LAB tool	Desktop app (Python/Qt/VTK); 3D Z-stacks	CNN segmentation (3 models) $\rightarrow$ spine identification $\rightarrow$ post-processing; <i>Tunable:</i> Noise thresholds, watershed splitting, component stitching parameters	Voxel relabeling, fragment stitching, watershed splitting, ROI selection; <i>Output:</i> Operation history, 3D meshes, geometric metrics	Automated CNN dendritic and spine segmentation with manual curation toolset.
PiPrA	Standalone GUI (Python/PyQt5); TIFF stacks	Manual annotation stage only; <i>Tunable:</i> Customizable brush/foreground colors and presets	4 manual modes (brush, flood fill, contour outline, GrabCut foreground estimation); <i>Output:</i> Binary masks (HDF5), file lists, TIFF/MP4 exports	Pixel-precise ground truth mask generation and manual ROI drawing.
Spot Spine	ImageJ/Fiji plugin; 3D stacks + SWC traced dendrite model	3 steps: Local maxima detection, head segmentation, neck tracing; <i>Tunable:</i> Maxima detection and intensity thresholds	Manual add/re-move/relocate maxima, contour editing, neck path deletion; <i>Output:</i> Quantitative metrics (head volume/surface area, neck length)	Morphometric quantification for various staining and visualization techniques.

Continued on next page

Table S2: Comparison of software tools and platforms for dendritic spine analysis and morphometrics.

Tool	Platform / GUI & Input Data	Processing & Tunable Parameters	Stages	Manual Annotation / Correction & Output	Core Functionality
SpineJ	ImageJ/Fiji plugin; 2D projections (STED, confocal)	3 stages: binarization, auto reconnection, skeletonization → morphometry; <i>Tunable</i> : Real-time binarization thresholds	Wavelet semi-	Visual verification and disconnection fixing at each step; Output quality depends on filter tuning	Nanometer-precise measurement of neck width/length and head area.
SpineTool	Python GUI pipeline; 3D confocal images	5 steps: binarization, mesh reconstruction, skeletonization, auto-labeling; <i>Tunable</i> : Adaptive binarization parameters	Viewing, surface	Manual verification/correction of spine-dendrite candidate links, ML shape classification; <i>Output</i> : 3D spine mesh dataset, CLDH morphometric metrics	3D spine mesh generation, CLDH morphometrics, and ML shape classification.

3 Proposed AI segmentation framework

3.1 Neural network more detailed

All four stages share the same 3D U-Net architecture [16] with four downsampling levels, a bottleneck, and skip connections. Convolutional blocks use $3 \times 3 \times 3$ kernels with batch normalization and Swish activation function, $\text{swish}(x) = x/(1 + e^{-x})$; outputs are mapped to probabilities by sigmoid. All stages were trained with Adam [17] at a learning rate of 0.005, with the parameters recommended by the authors ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$).

I is the acquired volume, I_s its version rescaled to 0.1 μm per pixel in XY, and I_c the preprocessed volume; $\|X\|$ denotes min-max normalization and X_p the region of X corresponding to patch p . Patch sizes, overlaps, filter counts and losses are given per stage in Table S3.

Stage 1 dendrite segmentation The network receives three channels,

$$\|I_c\|_p, \quad \|p\|, \quad \frac{\text{Otsu}(I_c, 4)_p + \text{Otsu}(p, 4)}{6}, \quad (4)$$

that is, the preprocessed volume normalized over the whole image and restricted to patch p ; the same patch normalized on its own, local contrast is seen without losing global context; and a four-class multi-Otsu map [18] computed at both scales, its summed labels (0–6) rescaled to $[0, 1]$. Trained with the mean squared error, it outputs the dendrite probability L_{dendrite} .

Stage 2 neck reconstruction Subtracting 70% of the mean intensity from I_s and clipping negatives retains the near-background signal in which necks lie; normalizing and scaling to 0–255 gives I_{interim} , from which 25 masks $M_j = [I_{\text{interim}} \geq 8j]$, $j = 2, \dots, 26$, are formed. The stage-1 map is thresholded at 0.5 and its connected components grouped by reachability through M_j ; the number of groups is non-decreasing in j , and j_0 is the largest j before it grows. Intensities

are stretched piecewise-linearly about $j_0 - [0, 8(j_0-1)] \rightarrow [0, 0.4]$, $[8(j_0-1), 8(j_0+1)] \rightarrow [0.4, 0.8]$, remainder $\rightarrow (0.8, 1.0]$ giving I_{si} , the fourth channel alongside $\|I_c\|_p$, $\|p\|$ and the stage-1 map.

Stage 3 resolution restoration Naive upscaling of the stage-2 mask produces boundary artefacts. The probability maps of stages 1 and 2 are instead brought to native resolution and Gaussian-smoothed, then network refined them from four channels: these two maps, the normalized original-resolution image, and I_{si} processed the same way. The loss is $\mathcal{L}_{\text{Dice}}$.

Stage 4 spine segmentation 4 stage applies to any voxel model of a dendrite. And refines the a segmentation produced by VSOT [6]. VSOT requires an unbranched dendrite, thats why the skeleton is extracted first, branches shorter than 6 μm are discarded as spine protrusions, and the remaining branches split the dendrite into unbranched segments. On each of which VSOT is run. Then the network receives three channels: the VSOT segmentation and two normalized distance maps, one to the shaft skeleton and one to the spine axes - segments drawn from each surface point farthest from the skeleton within 0.5 μm to the nearest skeleton point. Only patches containing both shaft voxels and either background or spine voxels are processed. Where a spine head falls outside every processed patch it keeps its original shaft label while its neck is relabelled as spine, leaving a small shaft-labelled component detached from the shaft itself, such components are relabelled as spines.

3.2 Training details

All four segmentation stages were trained one by one for each cross-validation fold using PyTorch Lightning [19]. The random seed was fixed at 42, and optimization was performed on a GPU. No automatic early-stopping criterion was used. The batch sizes and training durations were stage dependent, details of which are provided in the “Training” section of Table S3.

Validation was performed after every epoch. At each validation pass, the validation loss and the stage-specific accuracy metrics were calculated. Control points were saved based on the validation loss: after each epoch, the two control points with the lowest validation loss value recorded up to that point were saved, along with the control point from the most recent epoch. During training, TensorBoard raw statistics were recorded at intervals specified in the PyTorch Lightning logger, equal to 50 optimizer steps. In the metric plots shown in Figure S2C, all training loss and accuracy records corresponding to the same epoch were averaged, reporting a single value for each metric and each epoch.

Thus, the number of training epochs for each stage was fixed across all folds and did not depend on accuracy metrics. Indeed, the minimum validation loss was often achieved before the end of the last epoch. The model with the best validation loss value was defined as a target model and used to generate predictions on the test images.

4 Interactive segmentation platform

4.1 Human-in-the-Loop Annotation Workflow

Ground-truth annotation. Ex vivo and in vitro three-dimensional image stacks were processed by a few independent experts using the semi-automated mode of SpineTool 2.0. Using several annotators reduces the influence of individual bias and expands the annotated dataset. The semi-automated procedure relies on the same algorithmic core as SpineTool [20] - surface reconstruction, skeletonization and distance-based separation of spines from the shaft, and now wrapped in a standalone application built on PyQt5 with a napari/vispy-derived viewer [21]. Relative to the original version, it adds an area-of-interest stage, an automatic method for recovering faint spine necks, brush-based manual correction, and user-adjustable parameters at every stage. Excluding image import, in which voxel dimensions are specified, annotation proceeds through

eight stages. At the area of interest stage a segment of a secondary dendrite up to its branching point is selected (criteria for selecting the region: segments free of overlapping dendrites and crossing spines, and only where individual spines could be resolved reliably); voxels outside the selection are set to zero, since the downstream algorithms assume a single unbranched shaft. Each segment is then processed independently through the remaining stages. At the binarization stage the dendrite is separated from the background using global and adaptive thresholding. At the find points stage the operator marks attachment sites at the spine head-neck junction and spines whose necks were too thin to be recovered during binarization. These point annotations drive automatic reconstruction of the necks at the neck restoration stage. Spines are then separated from the shaft on the voxel model, after which a polygonal model is built at the mesh stage and divided into shaft and spines at the mesh segmentation stage. The voxelization stage converts the polygonal model back to voxels for component-wise inspection, and the final segmentation stage produces the segmented dendrite, exported as an .off file. Every stage is under operator control and can be adjusted either through its parameters or manually with the built-in brush tools. Importantly, this is not voxel-wise manual delineation: object boundaries are computed algorithmically - adaptive thresholding for the foreground, connected-component analysis and A*-based path search for the necks, and marching cubes for the surface - while the operator supplies the decisions the algorithms cannot make. Because the human contributes judgement, the same expert decisions reproduce the same boundaries. Manual delineation offers no such guarantee: inter-rater reliability among seven experts annotating the same volume voxel-wise has been reported at 82.2 ± 6.4 [4], a level of disagreement that limits its value as ground truth for evaluating automated methods. Binary segmentation masks were extracted from each stage of SpineTool 2.0 with custom Python code and used as input for training the segmentation networks.

4.2 Segmentation preprocessing

4.2.1 AI segmentation preprocessing

The microscopy image stacks were passed to the train and validation dataset-generation scripts in their original form, without the use of third-party software - that is, without prior deconvolution, denoising, or other image-enhancement procedures. When working with test images to generate a prediction or when working with a finished segmentation model in the SpineTool2 software, the input image can also be passed in its original form, without any such processing. The only transformations were those performed inside the scripts to construct the model-specific input tensors: spatial resampling where required, intensity normalization, calculation of the additional input channels described with the network architecture, region-of-interest masking when available, padding, patch extraction, and training augmentation.

The Gaussian smoothing with $\sigma = 2$ —used by the Stage 1 and Stage 2 channel generation functions, as well as for the Stage 3 auxiliary prediction channels—was applied as part of the tensor preparation process to transfer them to the segmentation model.

The first two stages used a common lateral sampling interval of $0.1 \mu\text{m}$ per pixel. For an input volume with lateral voxel sizes Δ_y and Δ_x and dimensions $N_y \times N_x$, the resampled dimensions were

$$N'_y = \text{round}\left(N_y \frac{\Delta_y}{0.1 \mu\text{m}}\right), \quad N'_x = \text{round}\left(N_x \frac{\Delta_x}{0.1 \mu\text{m}}\right).$$

Continuous-valued image channels were resized slice by slice using area interpolation, whereas discrete annotation masks were resized by nearest-neighbour interpolation. At Stage 3, the probability maps generated by Stages 1 and 2 were resized to the native $Y \times X$ dimensions of the original volume and combined with the native-resolution image. Stage 4 likewise operated on the native voxel grid; the physical voxel dimensions stored for each image were used explicitly in skeletonization and distance calculations. Skeleton extraction, VSOT segmentation, and con-

struction of the distance channels are described separately in the Supplementary Methods section on 2.1.

All image stacks were divided into fixed-size three-dimensional fragments that overlap one another to enable segmentation along tile boundaries. The fragments and overlap sizes are listed in the "Inference" section of Table S3.

Spatial augmentation was applied only to the training sets. For every training volume, three angles were sampled from $\{0^\circ, 10^\circ, \dots, 350^\circ\}$, and all image, auxiliary, and target channels were rotated jointly around the Z axis. Continuous-valued channels were interpolated linearly, while class masks and validity masks used nearest-neighbour interpolation. Rotations increased the variety of orientations and boundary contexts without changing the axial resolution. Limiting the expansion to rotations in the plane was reasonable, since the orientation of the dendrites in the visualization plane is arbitrary, whereas rotations affected by the anisotropically discretized Z -axis would have led to significantly more artifacts of interpolation. The rotations in Stage 4 were applied to enlarged regions of size $64 \times 184 \times 184$, after which central cropping was performed to a size of $64 \times 128 \times 128$, which prevented empty corners.

4.2.2 Semi-auto segmentation preprocessing

During the semi-automatic labeling process using algorithmic segmentation methods, image stacks were prepared to improve the accuracy of the algorithms and simplify the manual correction process performed by experts. Noise removal algorithms and the Richardson-Lucy deconvolution algorithm were used [22, 23].

4.3 Segmentation postprocessing

4.3.1 Processing of stage 4 segmentation results

The neural network pipeline makes predictions on tiles of the input z-stack image. The model predicts separate sigmoid probability maps for the dendritic shaft and the dendritic spines. If the VSOT input data for a tile did not contain shaft voxels, the fragment represents an isolated spine or spine head without sufficient dendritic context. In this case, during post-processing, we summed the predicted probabilities for the shaft and the dendritic spine, and set the shaft channel to zero. The tiles predictions were then placed at their original coordinates. Tiles were merged by averaging across voxels in overlapping regions, as in all other pipeline models.

After stitching all the tiles together and receiving a full-volume forecast, a second correction was applied. The preliminary shaft mask was defined by condition $P_{\text{shaft}} > P_{\text{spine}}$ and divided into components with the 26-fold connectivity. The largest component was assumed to represent the main dendritic shaft and was always preserved. Every other component containing at most 50,000 voxels was treated as a patch-context error and reassigned to the spine class. For every corrected voxel x , the probabilities were updated as

$$P'_{\text{spine}}(x) = \max(P_{\text{spine}}(x), P_{\text{shaft}}(x)), \quad P'_{\text{shaft}}(x) = 0.$$

The final class label $L(x) \in \{0, 1, 2\}$ for each voxel x was selected by

$$L(x) = \arg \max_{k \in \{0, 1, 2\}} (S_0(x), S_1(x), S_2(x)), \quad (S_0, S_1, S_2) = (0.1, P'_{\text{shaft}}, P'_{\text{spine}}),$$

where labels 0, 1, and 2 denote background, shaft, and spine, respectively. Thus, a voxel was classified as background when neither corrected foreground probability exceeded 0.1; otherwise, it was assigned to the foreground class with the larger probability. For voxels belonging to a corrected small shaft component, $P'_{\text{shaft}}(x) = 0$ and $P'_{\text{spine}}(x) \geq P_{\text{shaft}}(x)$. Consequently, such voxels could be classified only as spine or background, but not as shaft. This removed isolated shaft-labelled fragments caused by patches containing a spine head or a complete spine without sufficient shaft context.

4.3.2 False positive spines filtration

In the context of evaluating dendrite morphology in normal and abnormal conditions in a model of Alzheimer’s disease, the new segmentation algorithm was compared with the Spinetool. Spinetool involves manual selection of true positives and false positives dendritic spines as part of the segmentation process. To provide a comprehensive review of the new algorithm, the same post-processing of the segmentation results was also performed on the predictions from the neural network pipeline.

The dendritic spines from the resulting segmentations were reviewed by three experts; false-positive spines were removed from the segmentation and marked as part of the dendritic trunk, and morphological analysis was performed only on the filtered dendritic spines. The following incorrectly segmented regions were excluded from the analysis: 1 - a portion of the dendrite trunk that is not part of a spine was segmented; 2 - less than 50% of a dendritic spine was segmented; 3 - a dendritic spine was segmented without its head.

4.3.3 Dendrite component meshes extraction

The predicted multiclass volume was converted into surface meshes before calculating spine morphology with SpineTool [20]. Foreground voxels (shaft label 1 and spine label 2) were first partitioned into connected components using 26-neighbour connectivity. A component was retained only if it occupied at least 1% of the complete image volume and contained both shaft and spine voxels. Each retained component was processed independently and stored in a separate directory.

For each component, the shaft and spine labels were merged into a binary dendrite mask, and its principal connected component was selected. Further, the algorithm for building a mesh based on a voxel mask, which we described as mesh processing method, was applied to the current portion of the segmented dendrite.

The surface of individual dendritic spines is not watertight at the site of the spine’s attachment to the dendrite since it is part of the overall surface. Therefore, the spine mesh is not applicable for closed-surface SpineTool morphological measurements. So each spine mesh was divided into face-connected components. Components containing fewer than 10 faces were removed as noise. For each remaining component, the border half-edges were traversed to obtain each closed boundary loop. Algorithm first filled the loop with a polygonal face, and a new vertex was inserted at the centre of this face. Connecting the new vertex to every consecutive pair of boundary vertices replaced the polygon by a triangular fan. For a boundary loop $B = (v_1, \dots, v_n)$ and the inserted centre c , the added faces were

$$T_i = (v_i, v_{i+1}, c), \quad i = 1, \dots, n, \quad v_{n+1} = v_1.$$

Thus, the spine-to-shaft attachment was filled while preserving the original spine surface, resulting in a closed triangular mesh for subsequent SpineTool measurements.

4.4 Mesh processing methods

Polygonal surface models were generated from voxel segmentations. Connected components of the combined dendritic shaft and spine mask were identified using 26-connectivity. Components occupying less than 1% of the image volume were discarded as noise, while the remaining components were treated as separate dendritic segments and processed independently.

For each dendritic segment, the largest connected object was treated as the dendritic shaft. To smooth the surface without shrinking it, an opening (erosion followed by dilation, 6-connected structuring element) was combined with the original mask by union. After padding the mask with a zero layer to close the surface, an isosurface was extracted at level 0.5 by marching cubes [24], with spacing defined by the anisotropic voxel dimensions. Surfaces with fewer than 10 000 vertices were rejected, and small disconnected fragments were removed.

For each mesh vertex v , its voxel-space coordinates were rounded to the nearest voxel and clipped to the image bounds, giving $p(v)$. A neighbourhood $N(v)$ was then formed from all valid voxels within three voxels of $p(v)$ along each axis which is equivalent to a $7 \times 7 \times 7$ window away from the image boundaries. Let $n_{\text{shaft}}(v)$ be the number of voxels in $N(v)$ labeled as a dendritic shaft, and let $n_{\text{fg}}(v)$ be the total number of voxels in the considered window. The vertex was classified by the rule:

$$v \in \begin{cases} \text{spine candidate,} & \frac{n_{\text{shaft}}(v)}{n_{\text{fg}}(v)} < 0.75, \\ \text{shaft surface,} & \frac{n_{\text{shaft}}(v)}{n_{\text{fg}}(v)} \geq 0.75. \end{cases}$$

In other words, a surface vertex was marked as part of a spine when fewer than three of every four nearby dendrite voxels belonged to the shaft. Connected candidate regions were extracted from the surface and saved as separate spines. As a result, one complete dendrite mesh and a set of individual spine meshes are extracted for each processed dendrite component.

The surface of a dendritic spine is composed of the vertices of the dendrite surface passing the criterion; Connected regions of spine vertices formed individual spines, each exported as a separate .off file together with the full surface model.

Supplementary Figures

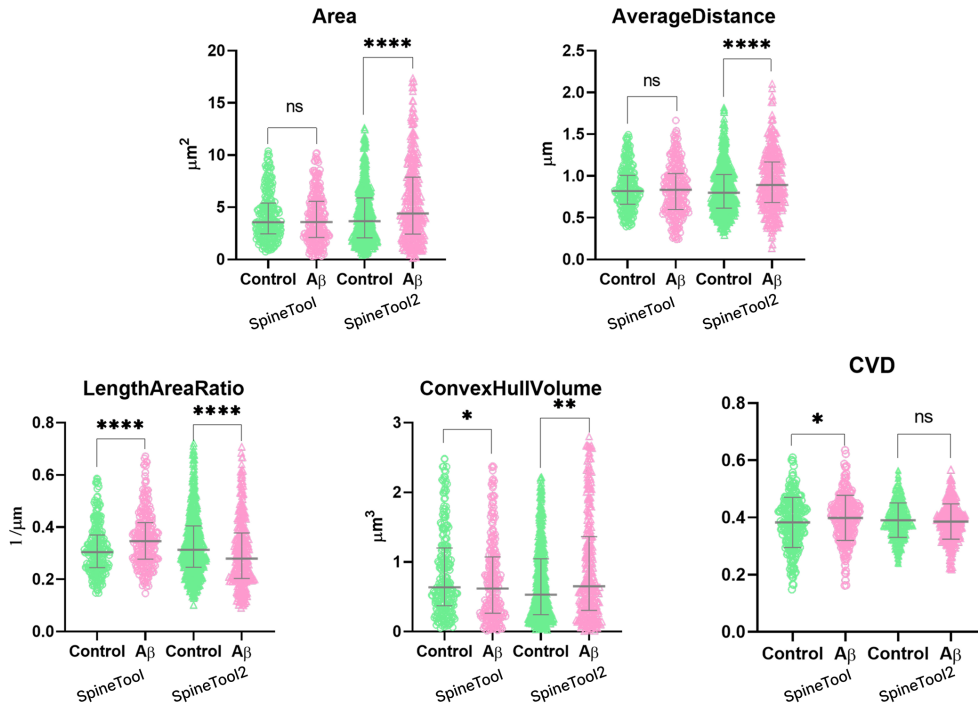


Figure S1: **Comparative morphometric analysis of the primary hippocampal neuron cultures morphology dataset [25] processing in SpineTool and SpineTool2.** Quantitative comparison between Control and pathological Aβ groups, including the total number of spines detected by SpineTool and SpineTool2. For all graphs data are presented as median with interquartile range [Q1; Q3] (Mann–Whitney test). * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$, **** $p < 0.0001$, ns = not significant.

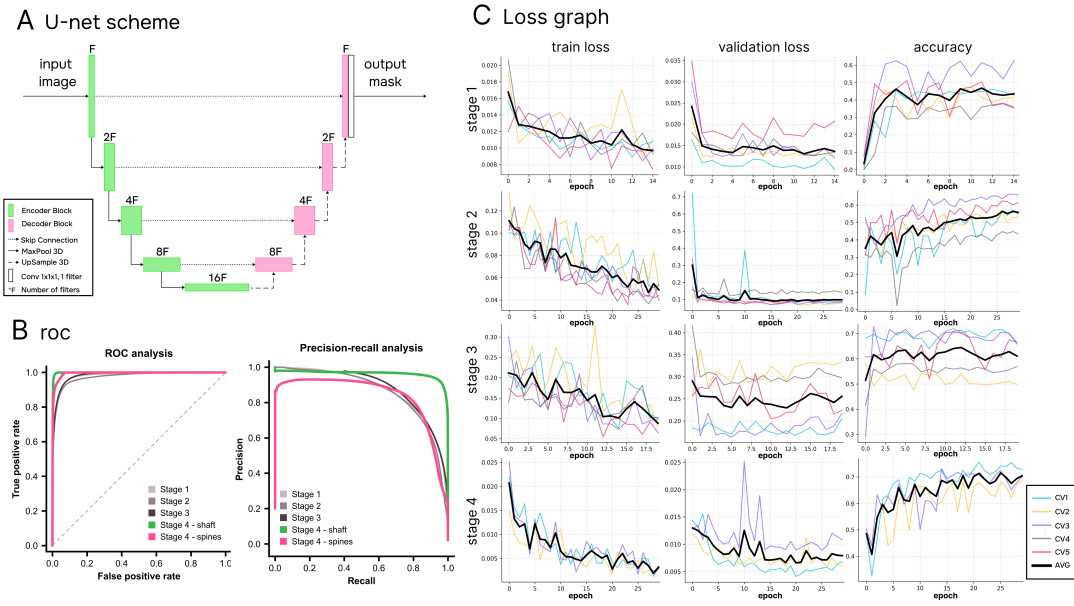


Figure S2: **Segmentation neural networks cascade.** A. schematic representation of the u-net architecture used by each cascade model. B. ROC plot of model sensitivity to the selected segmentation threshold and the relationship between segmentation precision and recall. C. Training and validation plots for cascade models, showing the loss function on the training and validation sets, and the model accuracy plots on the validation set: IOU based on a binary mask for stages 1–3, and IOU based on spines for stage 4

Table S3: Configuration of the proposed neural network models.

Parameter	Stage 1	Stage 2	Stage 3	Stage 4
Pipeline				
Task	Dendrite segmentation	Neck reconstruction	Image enhancement	Spine segmentation
Input	3 channels	4 channels	4 channels	3 channels
Output	Dendrite mask	Neck mask	Enhanced image	2-channel: dendrite shaft mask and dendritic spine mask
Network architecture				
Architecture	3D U-Net	3D U-Net	3D U-Net	3D U-Net
Encoder-decoder levels	5	5	5	5
Initial feature maps	32	64	64	64
Training				
Loss	MSE	$0.2L_{\text{Dice}} + 0.8L_{\text{Skeleton}}$	Dice	Masked BCEWithLogits
Optimizer	Adam	Adam	Adam	Adam
Learning rate	0.005	0.005	0.005	0.005
LR scheduler	–	ReduceLROnPlateau	ReduceLROnPlateau	–
Max. epochs	15	30	20	30
Batch size	16	4	3	3
Mixed precision	No	No	bf16	bf16
Gradient accumulation	1	1	4	4
Inference				
Patch size	$64 \times 64 \times 64$	$64 \times 64 \times 64$	$64 \times 128 \times 128$	$64 \times 128 \times 128$
Overlap	32	32	$32 \times 64 \times 64$	$48 \times 96 \times 96$
Strategy	Sliding window	Sliding window	Sliding window	Sliding window

Supplementary references

- [1] Anita Ustinova, Ekaterina Volkova, Anastasiya Rakovskaya, Daria Smirnova, Olesya Korovina, and Ekaterina Pchitskaya. Generate and analyze three-dimensional dendritic spine morphology datasets with spinetool software. *Current Protocols*, 4:e70061, 12 2024.
- [2] Suyu Sun, Hua Zhang, Jie Liu, Elena Popugaeva, Nan-Jie Xu, Stefan Feske, Charles L. White III, and Ilya Bezprozvanny. Reduced synaptic stim2 expression and impaired store-operated calcium entry cause destabilization of mature spines in mutant presenilin mice. *Neuron*, 82:79–93, 4 2014.
- [3] Elena Popugaeva, Ekaterina Pchitskaya, Anastasiya Speshilova, Sergey Alexandrov, Hua Zhang, Olga Vlasova, and Ilya Bezprozvanny. Stim2 protects hippocampal mushroom spines from amyloid synaptotoxicity. *Molecular Neurodegeneration*, 10:37, 2015.
- [4] Martin H P Fernholz, Drago A Guggiana Nilo, Tobias Bonhoeffer, and Andreas M Kist. Deepd3, an open framework for automated quantification of dendritic spines. *PLOS Computational Biology*, 20:e1011774–, 2 2024.
- [5] Sergio B Garcia, Alexa P Schlotter, Daniela Pereira, Aleksandra J Recupero, Franck Polleux, and Luke A Hammond. Respan: A deep learning pipeline for accurate and automated restoration, segmentation, and quantification of dendritic spines. *bioRxiv*, page 2024.06.06.597812, 1 2025.
- [6] Boyu Lyu, Jiangxiong Wang, William Christopher Risher, and Guoqiang Yu. Vsot: volume-surface optimization for accurate ultrastructure analysis of dendritic spines. *Bioinformatics*, 41:btaf220, 5 2025.
- [7] GNU Octave Developers. GNU Octave, 2026.
- [8] M. Sato, I. Bitter, M. A. Bender, A. E. Kaufman, and M. Nakajima. Teasar: Tree-structure extraction algorithm for accurate and robust skeletons. In *Proceedings - Pacific Conference on Computer Graphics and Applications*, volume 2000-January. IEEE Computer Society, 2000.
- [9] Lena Maier-Hein, Annika Reinke, Patrick Godau, Minu D. Tizabi, Florian Buettner, Evangelia Christodoulou, Ben Glocker, Fabian Isensee, Jens Kleesiek, Michal Kozubek, Mauricio Reyes, Michael A. Riegler, Manuel Wiesenfarth, A. Emre Kavur, Carole H. Sudre, Michael Baumgartner, Matthias Eisenmann, Doreen Heckmann-Nötzl, Tim Radsch, Laura Acion, Michela Antonelli, Tal Arbel, Spyridon Bakas, Arriel Benis, Matthew B. Blaschko, M. Jorge Cardoso, Veronika Cheplygina, Beth A. Cimini, Gary S. Collins, Keyvan Farahani, Luciana Ferrer, Adrian Galdran, Bram van Ginneken, Robert Haase, Daniel A. Hashimoto, Michael M. Hoffman, Merel Huisman, Pierre Jannin, Charles E. Kahn, Dagmar Kainmueller, Bernhard Kainz, Alexandros Karargyris, Alan Karthikesalingam, Florian Kofler, Annette Kopp-Schneider, Anna Kreshuk, Tahsin Kurc, Bennett A. Landman, Geert Litjens, Amin Madani, Klaus Maier-Hein, Anne L. Martel, Peter Mattson, Erik Meijering, Bjoern Menze, Karel G.M. Moons, Henning Müller, Brennan Nchyporuk, Felix Nickel, Jens Petersen, Nasir Rajpoot, Nicola Rieke, Julio Saez-Rodriguez, Clara I. Sánchez, Shravya Shetty, Maarten van Smeden, Ronald M. Summers, Abdel A. Taha, Aleksei Tiulpin, Sotirios A. Tsafaris, Ben Van Calster, Gaël Varoquaux, and Paul F. Jäger. Metrics reloaded: recommendations for image analysis validation. *Nature Methods*, 21:195–212, 2 2024.

- [10] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, volume 2, pages 1137–1143, 1995.
- [11] Curtis T. Rueden, Mark C. Hiner, Edward L. Evans, and et al. Pyimagej: A library for integrating imagej and python. *Nature Methods*, 2022.
- [12] Florian Levet, Jan Tønnesen, U Valentin Nägerl, and Jean-Baptiste Sibarita. Spinej: A software tool for quantitative analysis of nanoscale spine morphology. *Methods*, 174:49–55, 2020.
- [13] Subhadip Basu, Dariusz Plewczynski, Satadal Saha, Matylda Roszkowska, Marta Magnowska, Ewa Baczynska, and Jakub Wlodarczyk. 2dspan: Semiautomated 2-d segmentation, classification and analysis of hippocampal dendritic spine plasticity. *Bioinformatics*, 32:2490–2498, 8 2016.
- [14] Michael S Smirnov, Tavita R Garrett, and Ryohei Yasuda. An open-source tool for analysis and automatic identification of dendritic spines using machine learning. *PLOS ONE*, 13:e0199589–, 7 2018.
- [15] Isabel Vidaurre-Gallart, Isabel Fernaud-Espinosa, Nicusor Cosmin-Toader, Lidia Talavera-Martínez, Miguel Martín-Abadal, Ruth Benavides-Piccione, Yolanda Gonzalez-Cid, Luis Pastor, Javier DeFelipe, and Marcos García-Lorenzo. A deep learning-based workflow for dendritic spine segmentation. *Frontiers in Neuroanatomy*, 16:817903, 2022.
- [16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M Wells, and Alejandro F Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer International Publishing, 2015.
- [17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.
- [18] N Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9:62–66, 1979.
- [19] William Falcon and The PyTorch Lightning team. Pytorch lightning, 2019.
- [20] Ekaterina Pchitskaya, Peter Vasiliev, Daria Smirnova, Vyacheslav Chukanov, and Ilya Bezprozvanny. Spinetool is an open-source software for analysis of morphology of dendritic spines. *Scientific Reports*, 13:10561, 2023.
- [21] Nicholas Sofroniew, Talley Lambert, Grzegorz Bokota, Juan Nunez-Iglesias, Peter Sobolewski, et al. napari: a multi-dimensional image viewer for python.
- [22] William Hadley Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62(1):55–59, 1972.
- [23] Leon B. Lucy. An iterative technique for the rectification of observed distributions. *The Astronomical Journal*, 79:745–754, 1974.
- [24] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.
- [25] Daria Smirnova, Anita Ustinova, Viacheslav Chukanov, and Ekaterina Pchitskaya. 3d dendritic spines shape descriptors for efficient classification and morphology analysis in control and alzheimer’s disease modeling neurons. *Bioinformatics*, 42(2):btac025, 02 2026.