Whole-Slide Image Analysis and Quantitative Pathology with QuPath

Thierry Pécot
Research Engineer
Biosit SFR UMS CNRS 3480 – Inserm 018
CZI Imaging Scientist

Whole-Slide Image Size

- A **slide size** is generally **25 mm x 75 mm**

Whole-Slide Image Size

- A **slide size** is generally **25 mm x 75 mm**
- A **20X** magnification gives a pixel width of **0.45 µm**

Whole-Slide Image Size

- A **slide size** is generally **25 mm x 75 mm**
- A **20X** magnification gives a pixel width of **0.45 µm**
- If the **whole-slide image** is **scanned**:
    - (25e-3 * 75e-3) / (0.45e-6)^2 = **9 259 259 259 pixels**

Whole-Slide Image Size

- A **slide size** is generally **25 mm x 75 mm**
- A **20X** magnification gives a pixel width of **0.45 µm**
- If the **whole-slide image** is **scanned**:
    - (25e-3 * 75e-3) / (0.45e-6)^2 = **9 259 259 259 pixels**
- For a **whole-slide image** with **1 channel** of fluorescence:
    - 9 259 259 259 * 1 byte = **9.26 GB**
- For a **H&E(S) whole-slide image** with 3 channels (RGB):
    - 9 259 259 259 * 1 byte * 3 = **27.78 GB**

Whole-Slide Image Size

- A **slide size** is generally **25 mm x 75 mm**
- A **20X** magnification gives a pixel width of **0.45 µm**
- If the **whole-slide image** is **scanned**:
    - (25e-3 * 75e-3) / (0.45e-6)^2 = **9 259 259 259 pixels**
- For a **whole-slide image** with **1 channel** of fluorescence:
    - 9 259 259 259 * 1 byte = **9.26 GB**
- For a **H&E(S) whole-slide image** with 3 channels (RGB):
    - 9 259 259 259 * 1 byte * 3 = **27.78 GB**

not possible to load the whole data in the **RAM memory**

Whole-Slide Image Size

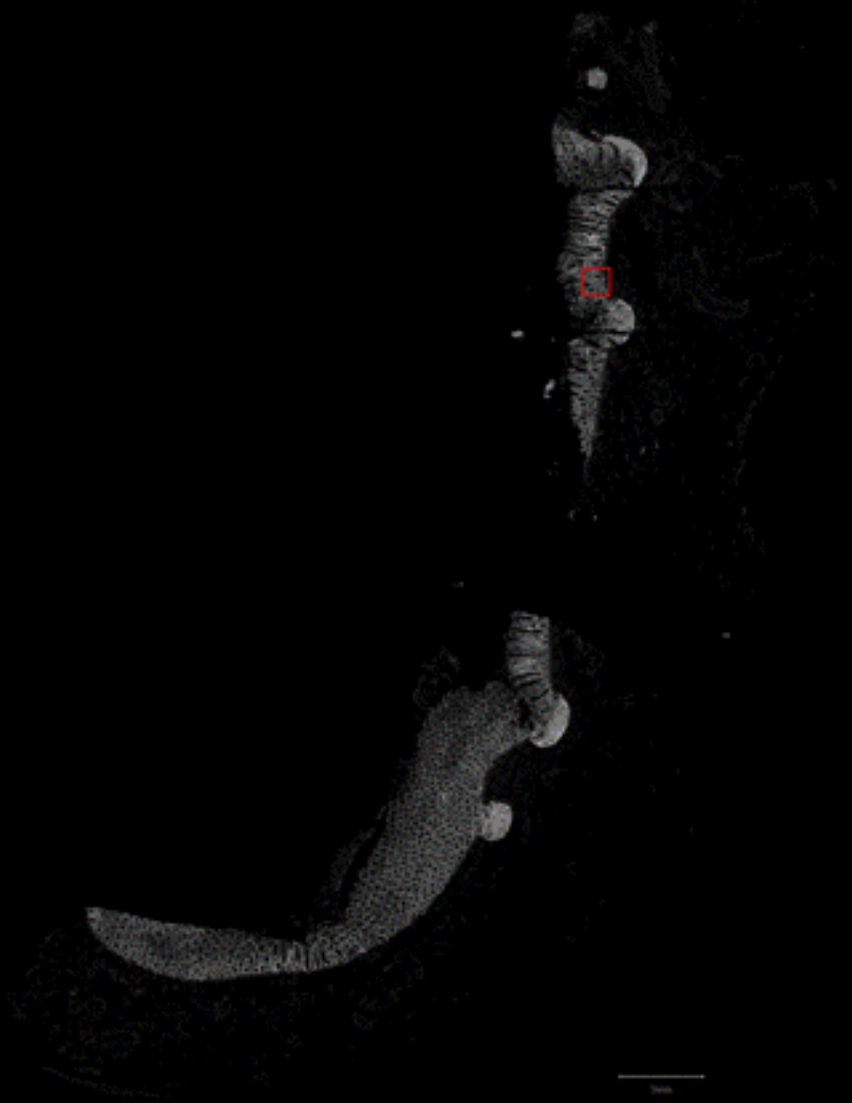
- A **slide size** is generally **25 mm x 75 mm**
- A **20X** magnification gives a pixel width of **0.45 µm**
- If the **whole-slide image** is **scanned**:
    - (25e-3 * 75e-3) / (0.45e-6)^2 = **9 259 259 259 pixels**
- For a **whole-slide image** with **1 channel** of fluorescence:
    - 9 259 259 259 * 1 byte = **9.26 GB**
- For a **H&E(S) whole-slide image** with 3 channels (RGB):
    - 9 259 259 259 * 1 byte * 3 = **27.78 GB**

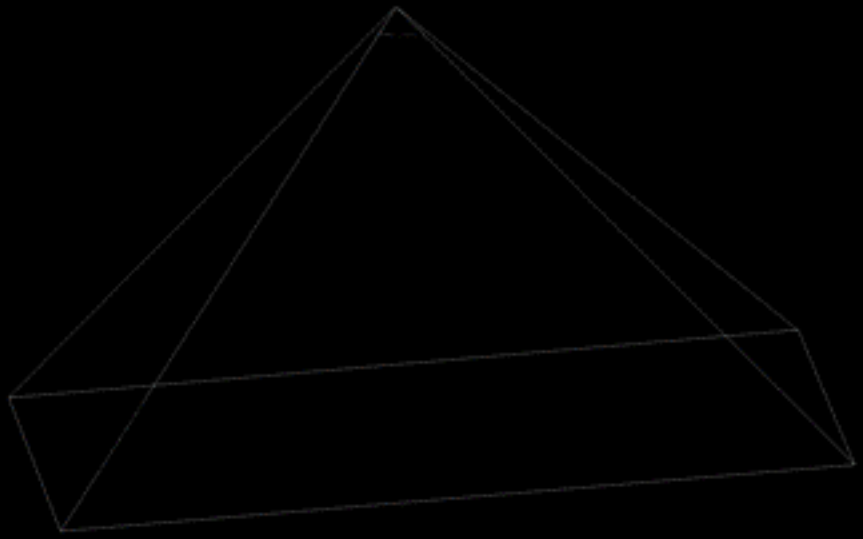
not possible to load the whole data in the **RAM memory**

only load the **required information** in the RAM memory:

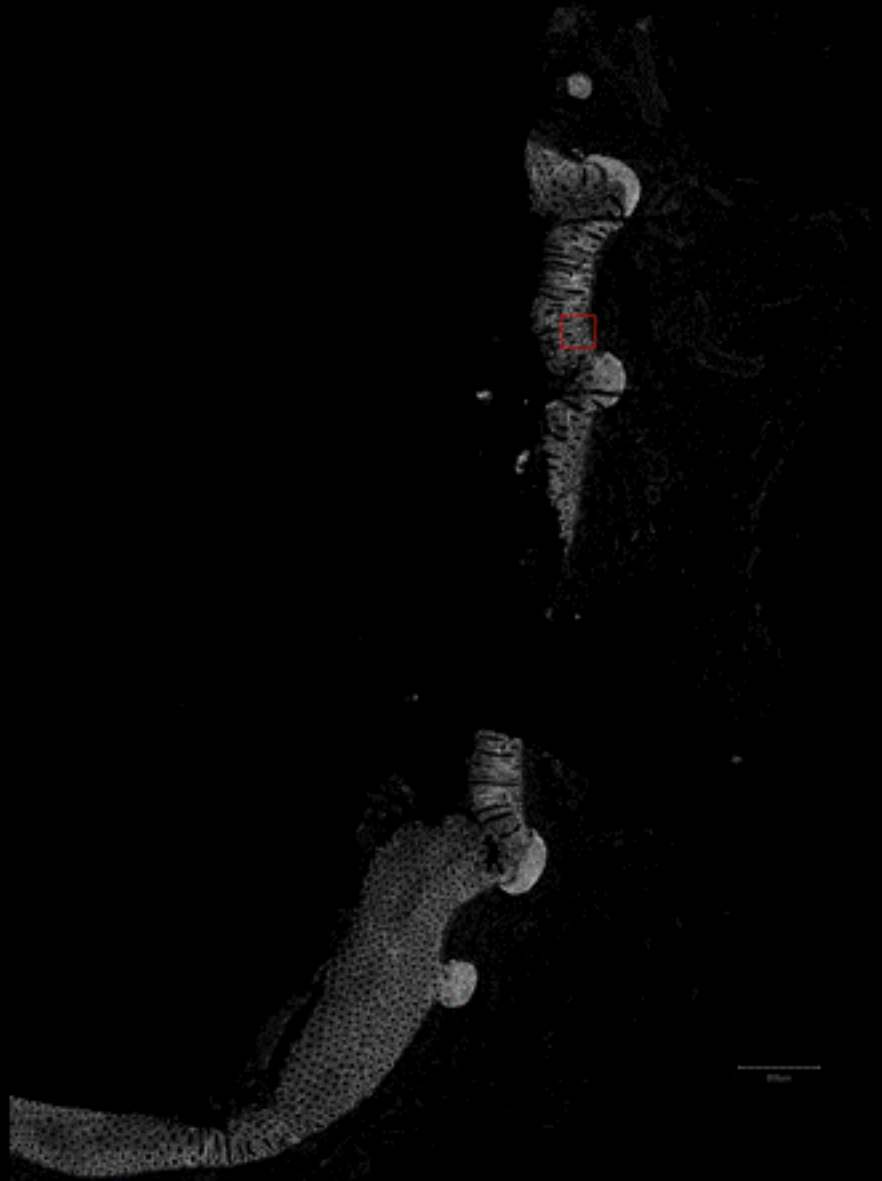
- Define **several resolutions** to create a **pyramidal representation**

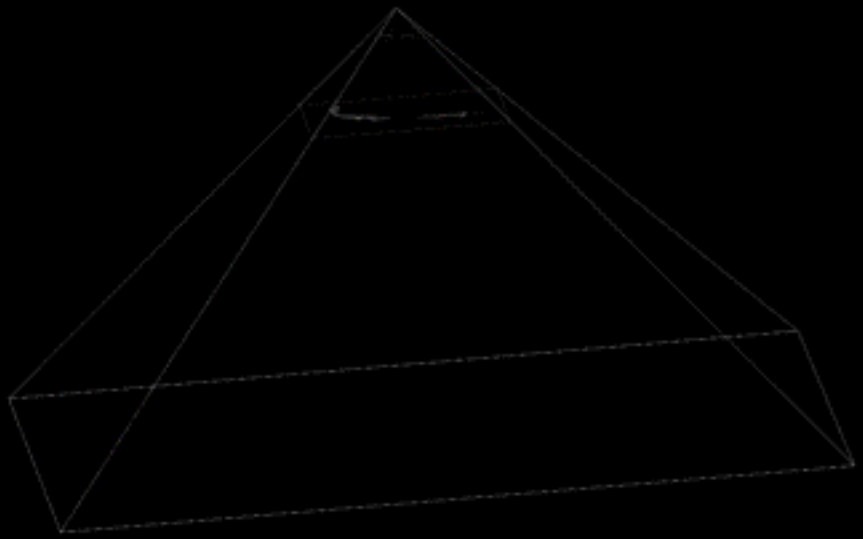
Whole Slide Image – Pyramidal Representation



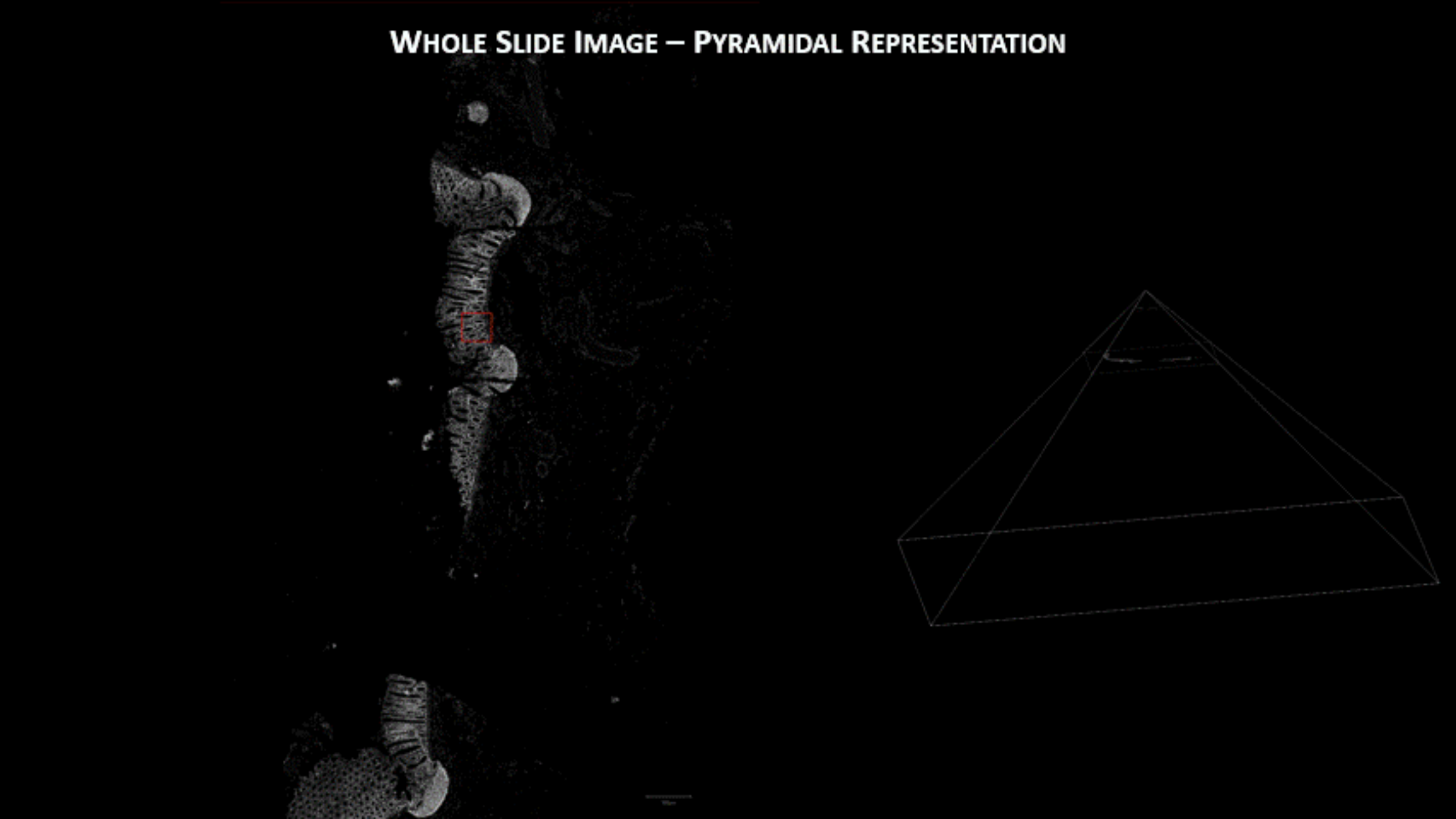


Whole Slide Image – Pyramidal Representation

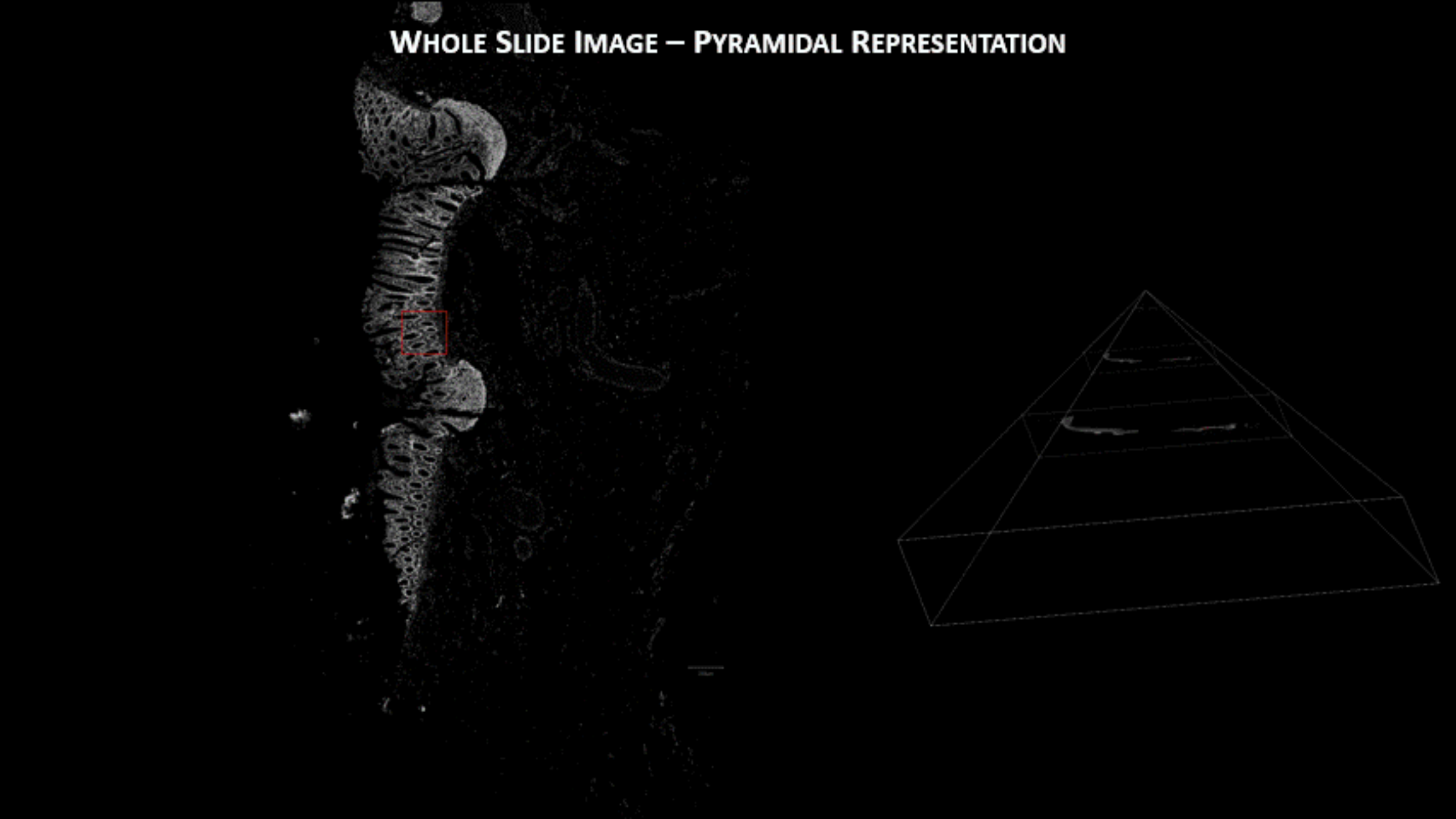




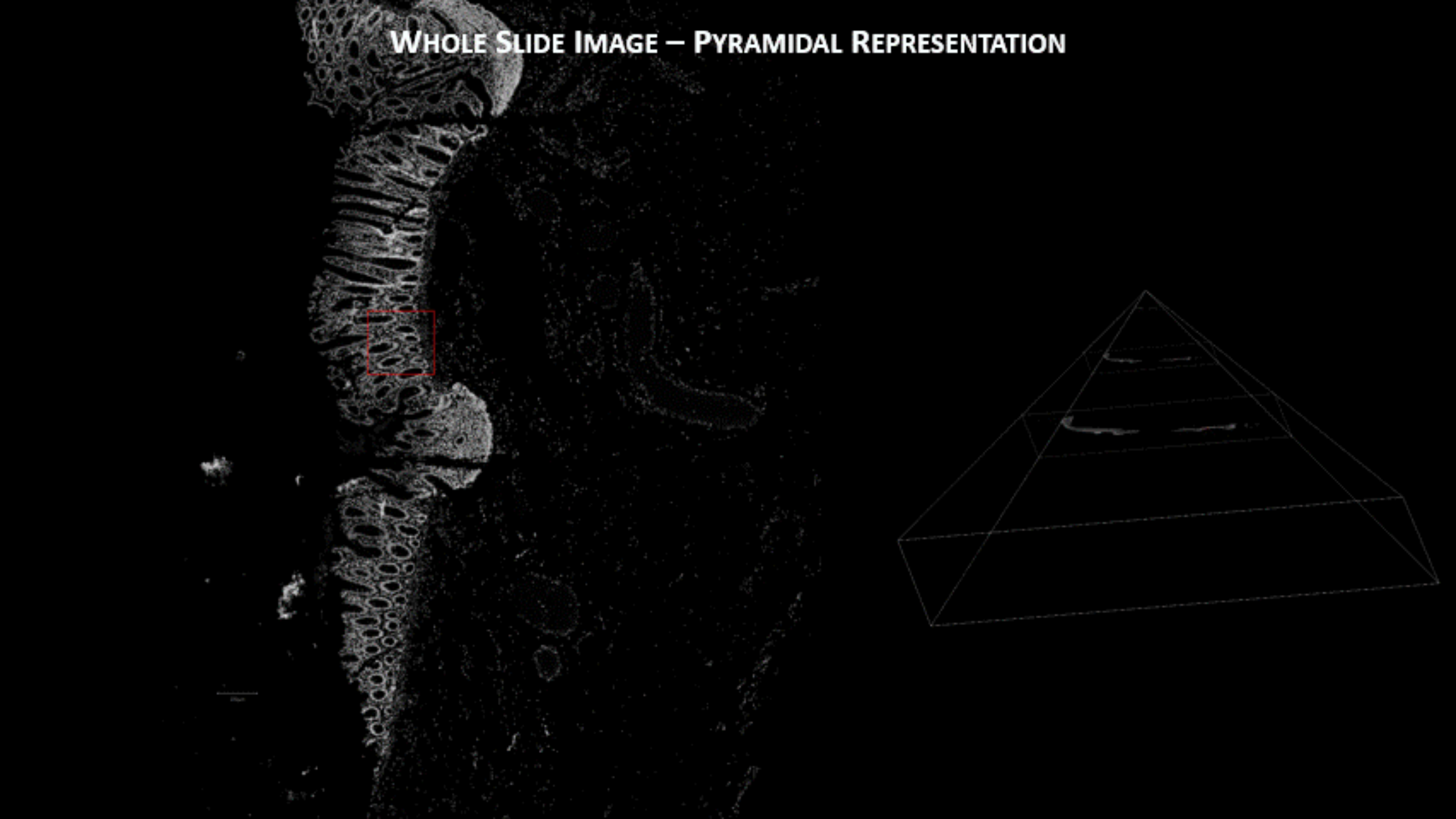
Whole Slide Image – Pyramidal Representation



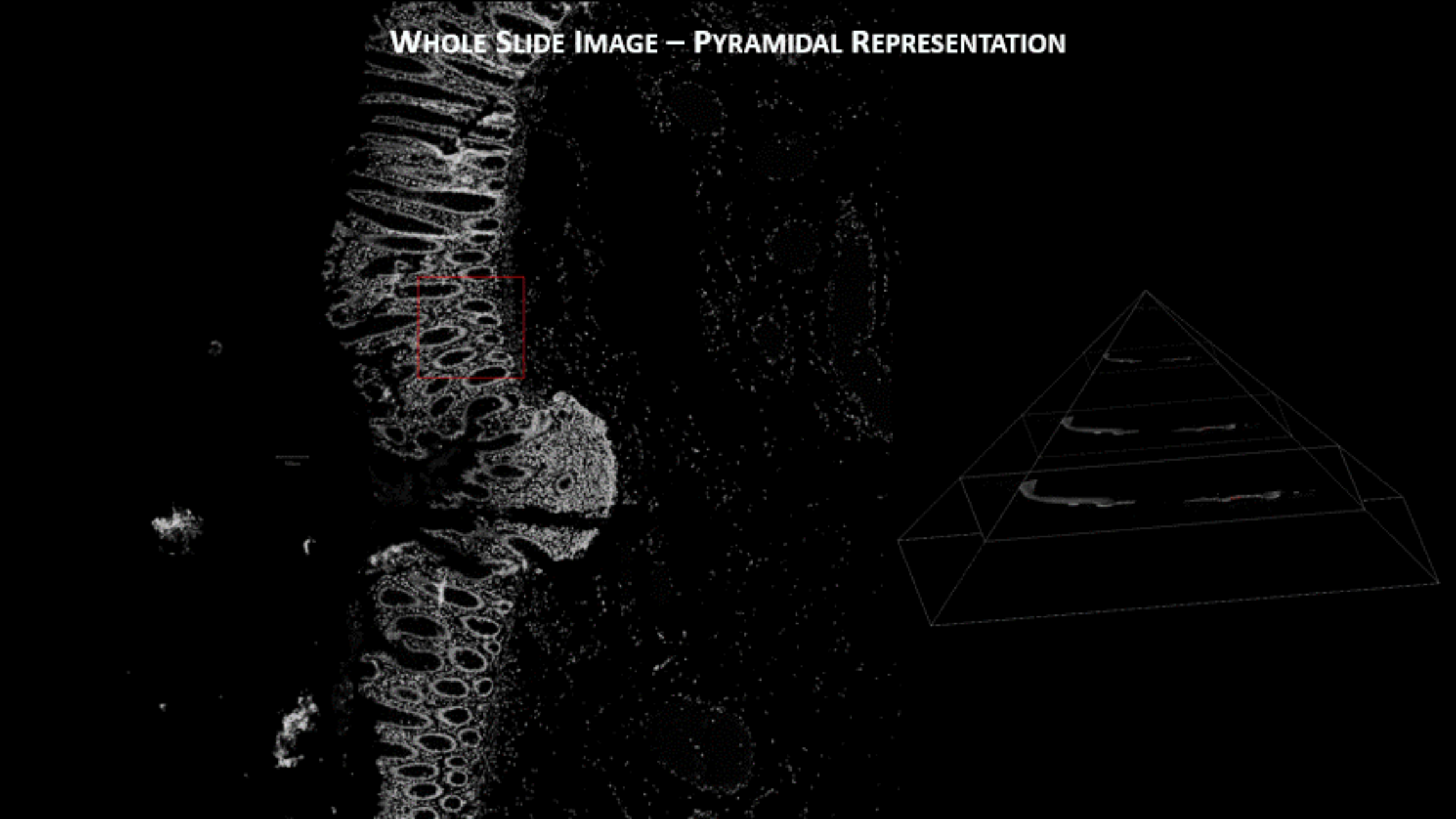
Whole Slide Image – Pyramidal Representation



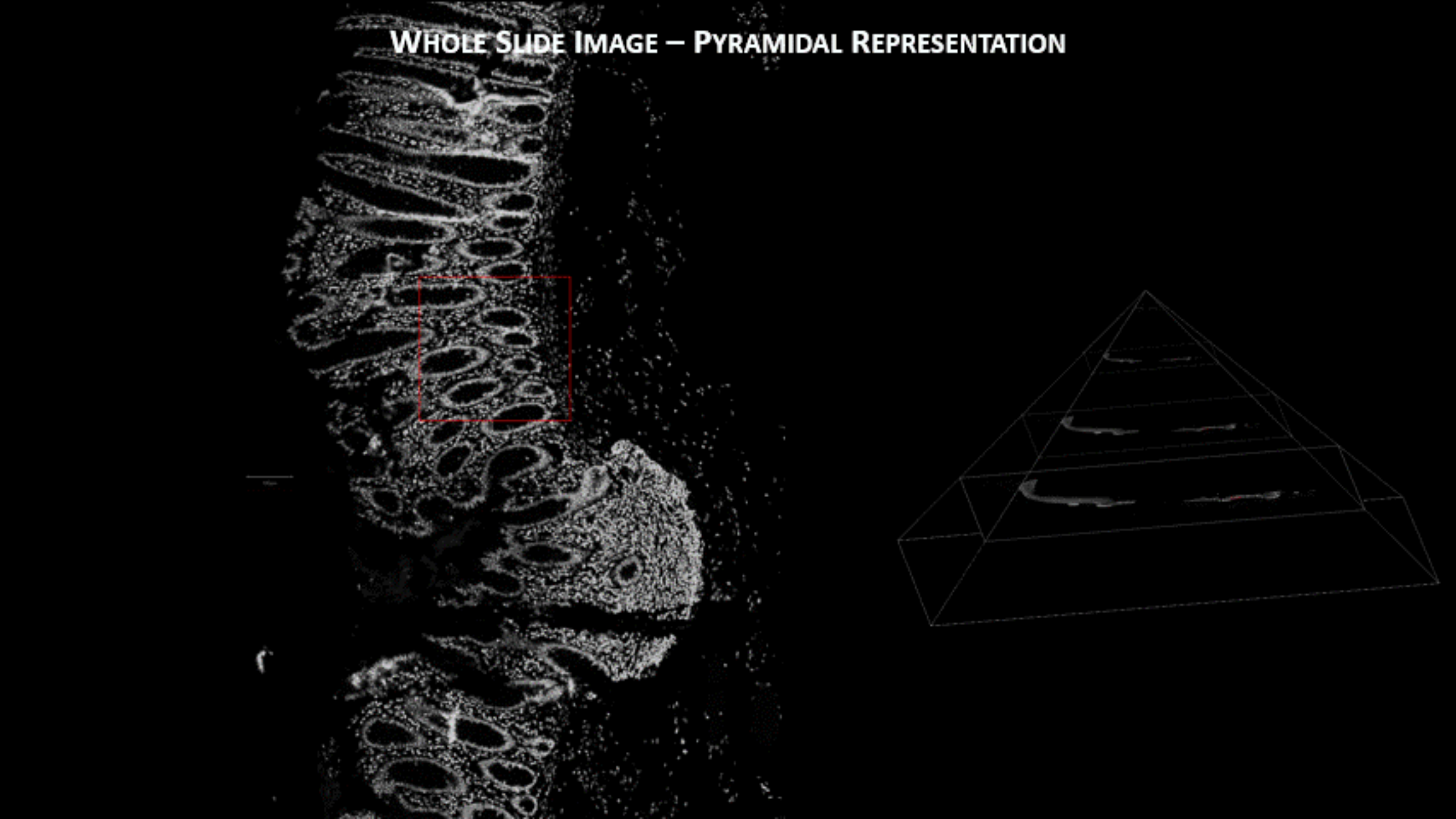
Whole Slide Image – Pyramidal Representation



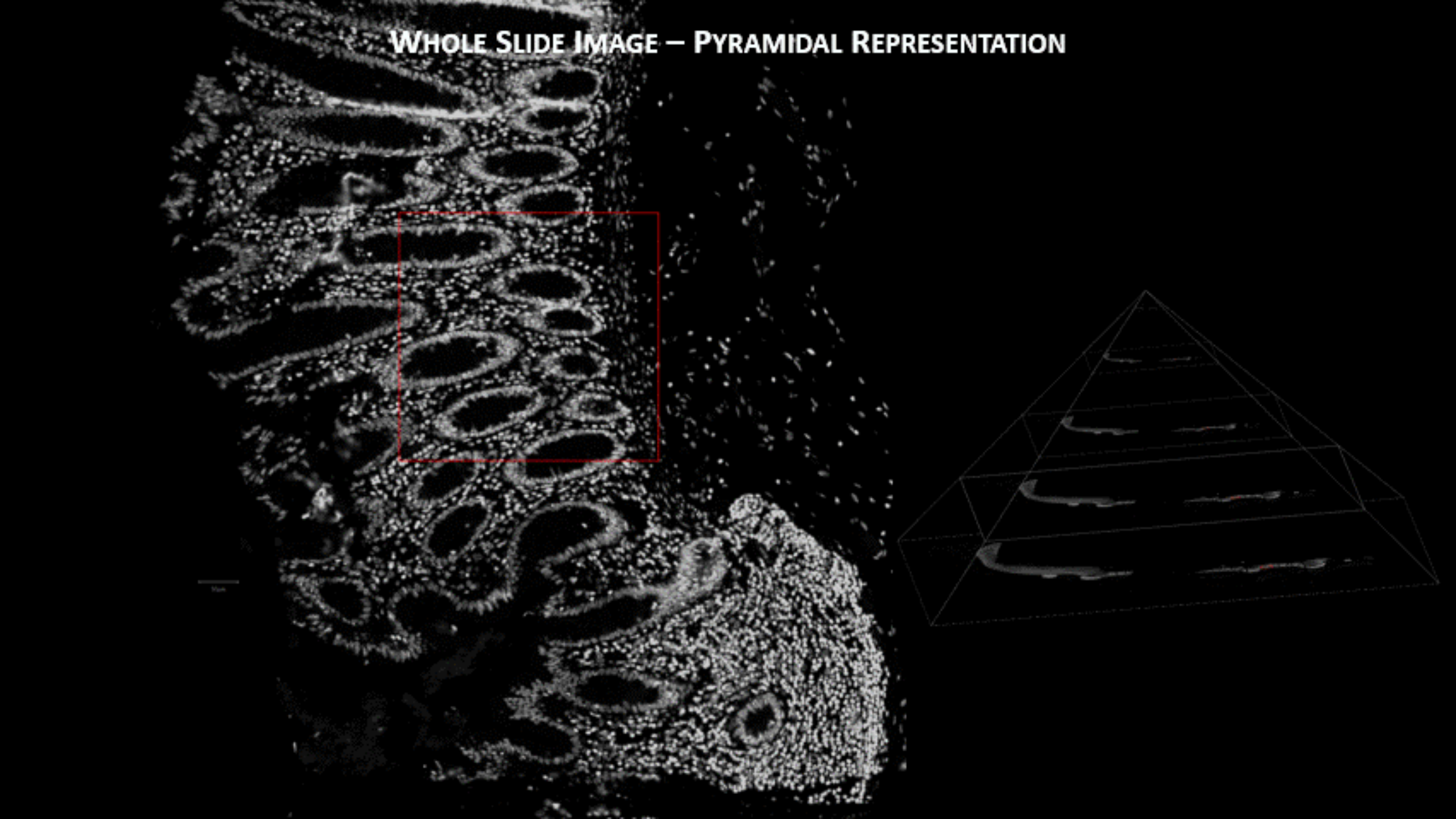
Whole Slide Image – Pyramidal Representation



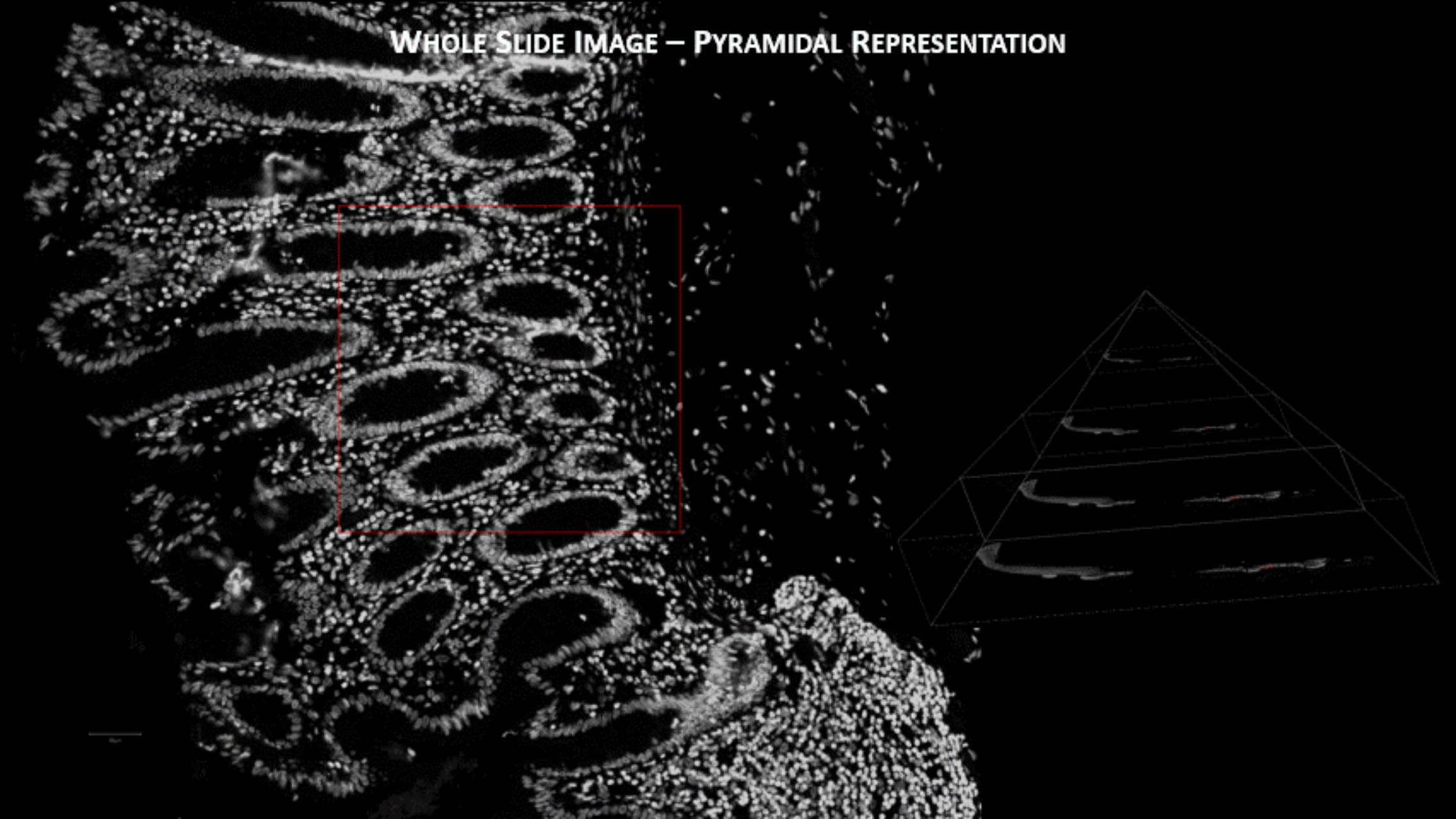
Whole Slide Image – Pyramidal Representation



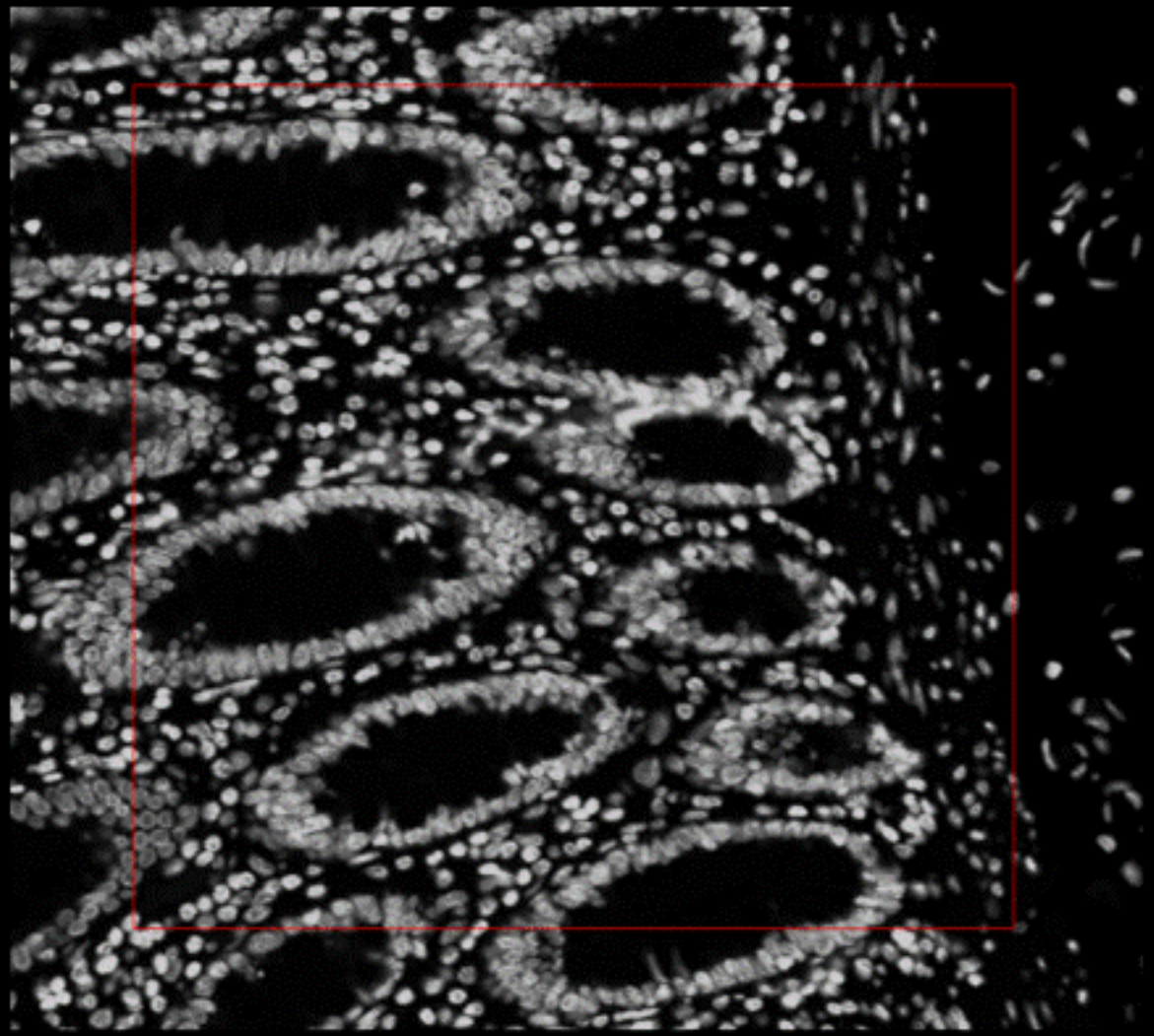
Whole Slide Image – Pyramidal Representation

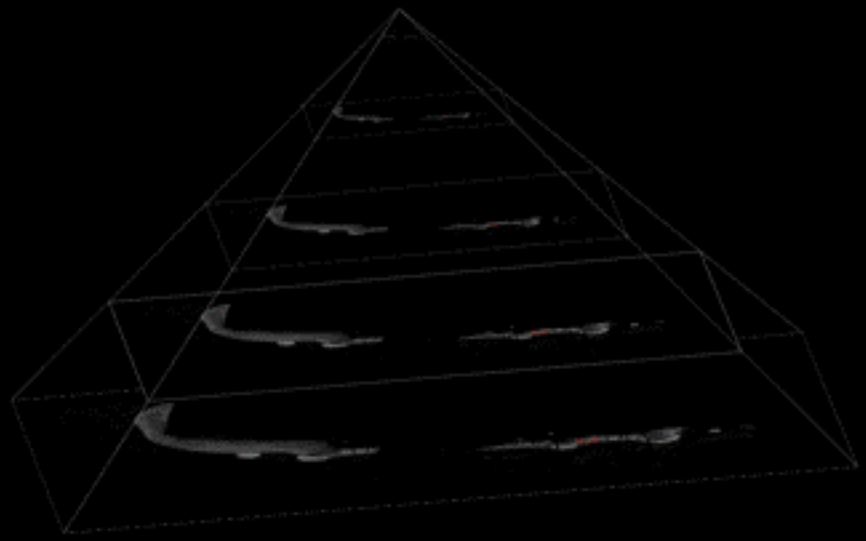


Whole Slide Image – Pyramidal Representation



Whole Slide Image – Pyramidal Representation





Whole-Slide Image Size

- A **full H&E(S)** whole-slide image would be:
  - 9 259 259 259 * 1 byte * 3 = **27.78 GB** for resolution 1
  - 9 259 259 259 * 1 byte * 3 = **6.94 GB** for resolution 2
  - 9 259 259 259 * 1 byte * 3 = **1.74 GB** for resolution 4
  - 9 259 259 259 * 1 byte * 3 = **434 MB** for resolution 8
  - 9 259 259 259 * 1 byte * 3 = **109 MB** for resolution 16
  - 9 259 259 259 * 1 byte * 3 = **27 MB** for resolution 32
  - 9 259 259 259 * 1 byte * 3 = **7 MB** for resolution 64
  - 9 259 259 259 * 1 byte * 3 = **1.7 MB** for resolution 128
  - 9 259 259 259 * 1 byte * 3 = **424 KB** for resolution 256

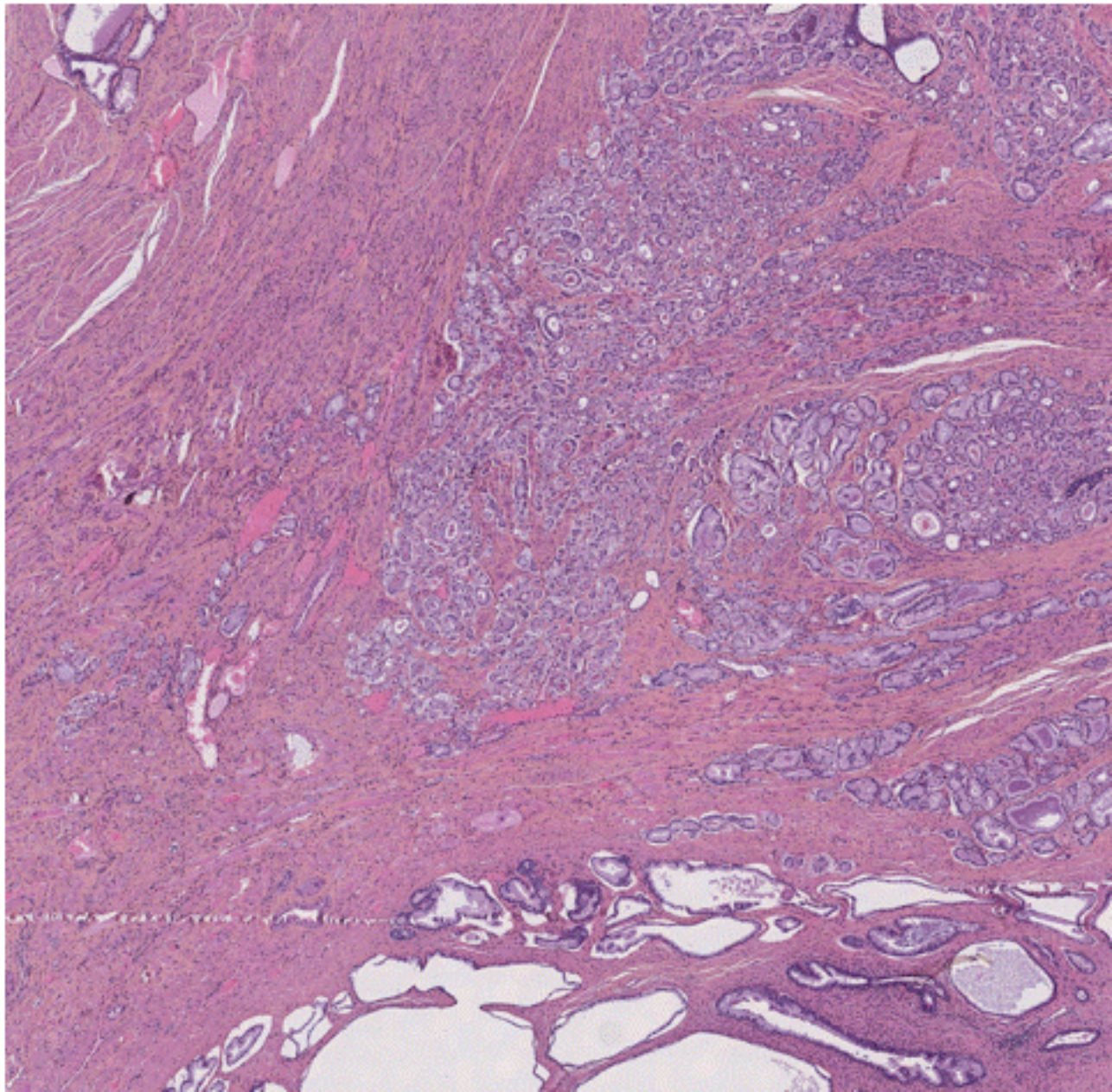
➡ Total of **37 GB uncompressed data**

ANNOTATIONS

- Allow to **add information** to specific regions or entire images
- Lots of **features/measurements** can then be extracted from these regions
- **Powerful** and **storage-efficient** way to process images
- Can be **manually** defined or **automatically** estimated
- Can be enriched by **adding classes**

ANNOTATIONS

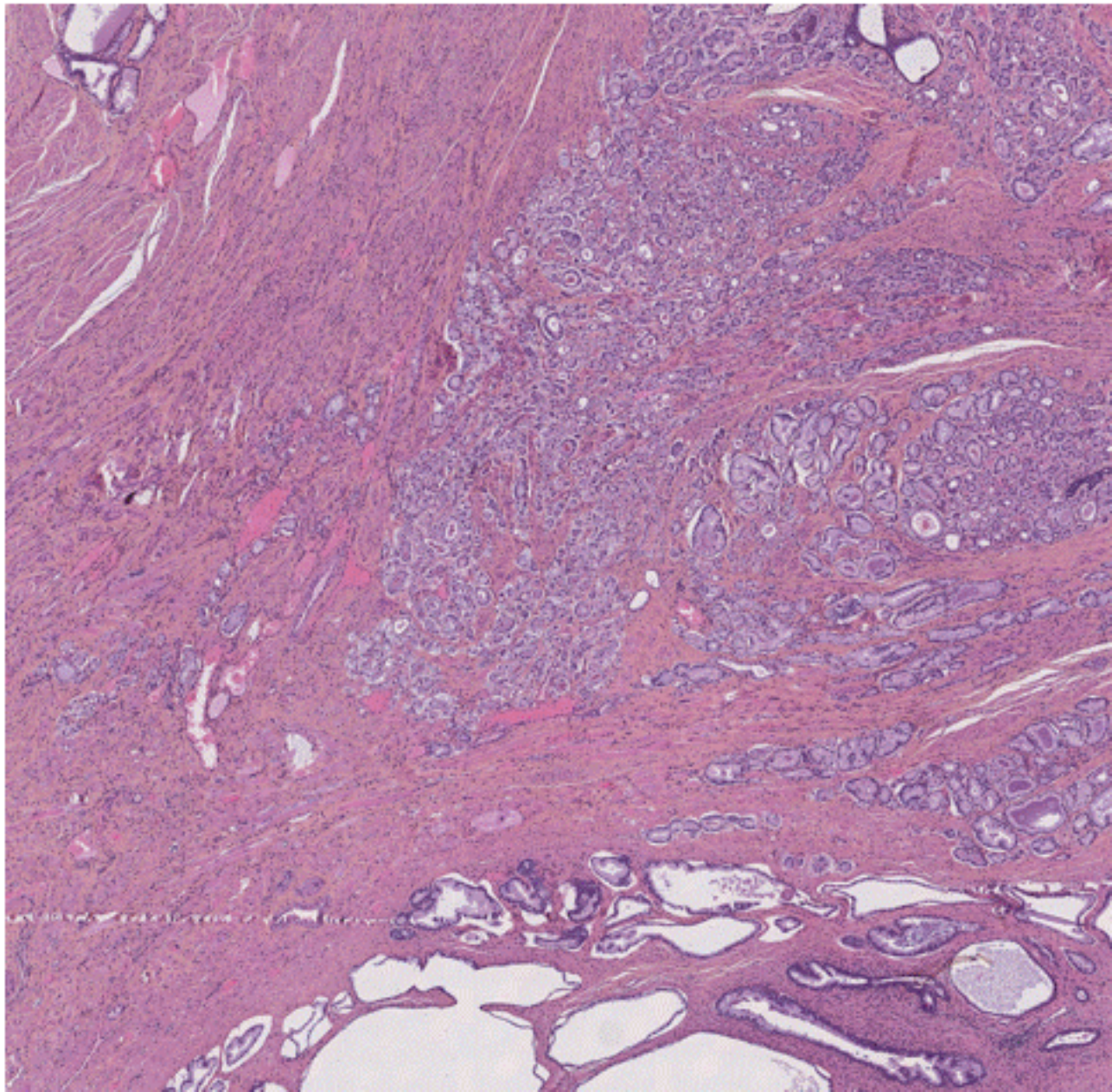
- Allow to **add information** to specific regions or entire images
- Lots of **features/measurements** can then be extracted from these regions
- **Powerful** and **storage-efficient** way to process images
- Can be **manually** defined or **automatically** estimated
- Can be enriched by **adding classes**



ANNOTATIONS

- Allow to **add information** to specific regions or entire images
- Lots of **features/measurements** can then be extracted from these regions
- **Powerful** and **storage-efficient** way to process images
- Can be **manually** defined or **automatically** estimated
- Can be enriched by **adding classes**

- **Open** Prostate1.ome.tif
- Create **different types** of annotations
- Play with **resolution**
- Look at the **measurements** for each **type of annotation**



Stain Estimation

- **Hematoxylin** stains **nuclei** in purple/blue
- **Eosin** stains **extracellular matrix** and **cytoplasm** in pink
- **DAB** is used to stain **antigens** in brown

Stain Estimation

- **Hematoxylin** stains **nuclei** in purple/blue
- **Eosin** stains **extracellular matrix** and **cytoplasm** in pink
- **DAB** is used to stain **antigens** in brown

- Slides are processed with a **brightfield scanner:**
  ➡ A **digital image** with 3 color components (**RGB**) is obtained

Stain Estimation

- **Hematoxylin** stains **nuclei** in purple/blue
- **Eosin** stains **extracellular matrix** and **cytoplasm** in pink
- **DAB** is used to stain **antigens** in brown

- Slides are processed with a **brightfield scanner:**
  ➡ A **digital image** with 3 color components (**RGB**) is obtained

- **Stain estimation** consists in transforming Red-Green-Blue channels to **Hematoxylin-Eosin/DAB-Residue** channels
- It greatly facilitates the **nuclei segmentation** in the **hematoxylin** component, the **DAB** region **characterization** in the DAB component, …

STAIN ESTIMATION

- **Hematoxylin** stains **nuclei** in purple/blue
- **Eosin** stains **extracellular matrix** and **cytoplasm** in pink
- **DAB** is used to stain **antigens** in brown

- Slides are processed with a **brightfield scanner:**
  ➡ A **digital image** with 3 color components (**RGB**) is obtained

- **Stain estimation** consists in transforming Red-Green-Blue channels to **Hematoxylin-Eosin/DAB-Residue** channels
- It greatly facilitates the **nuclei segmentation** in the **hematoxylin** component, the **DAB** region **characterization** in the DAB component, …

- **Automatic** stain estimation in **QuPath** is based on:

Comparative Study › Anal Quant Cytol Histol. 2001 Aug;23(4):291-9.

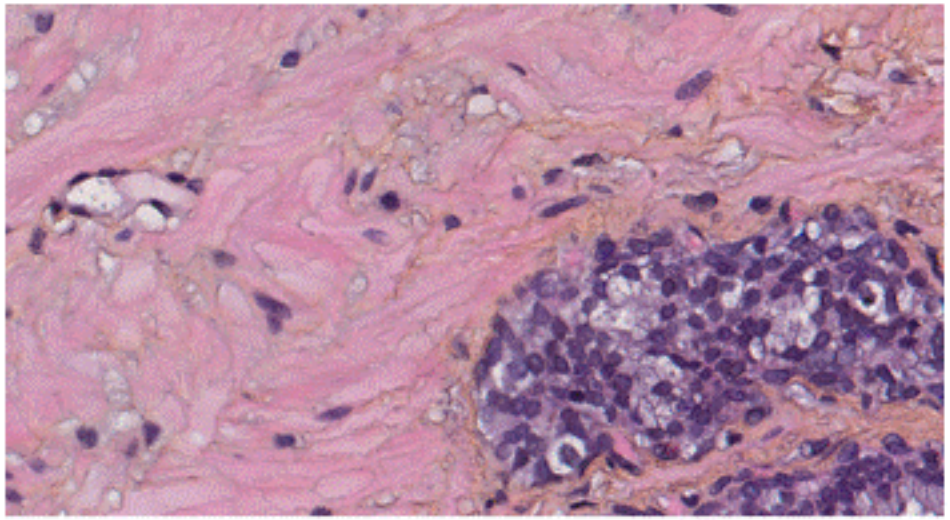
Quantification of histochemical staining by color deconvolution

A C Ruifrok [1], D A Johnston

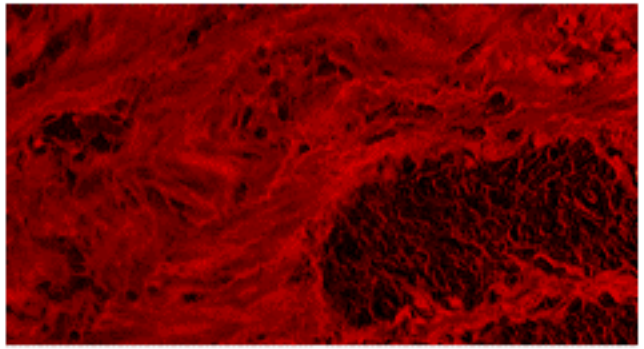
Affiliations + expand

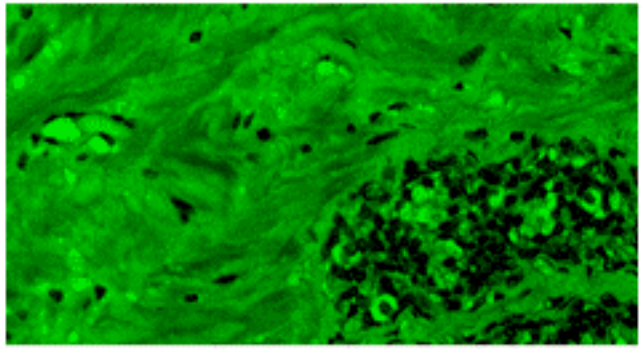
PMID: 11531144

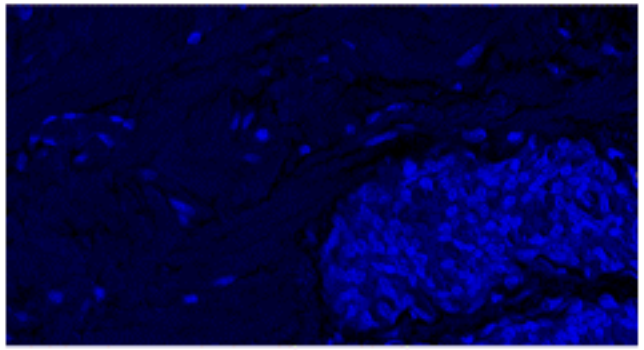
H&E Stain Estimation



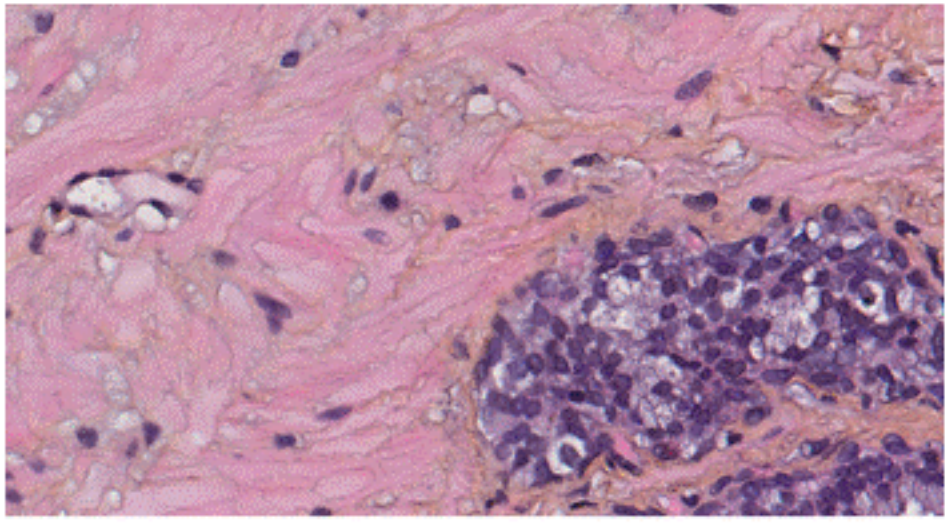
=



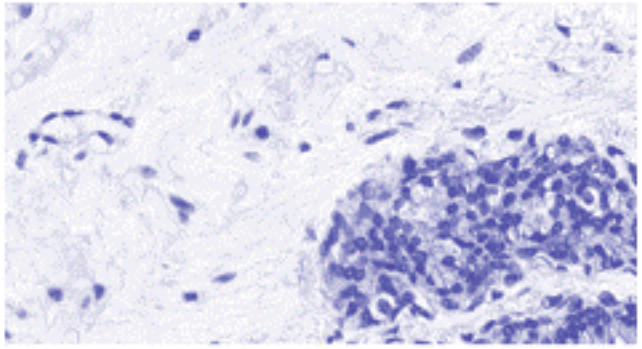


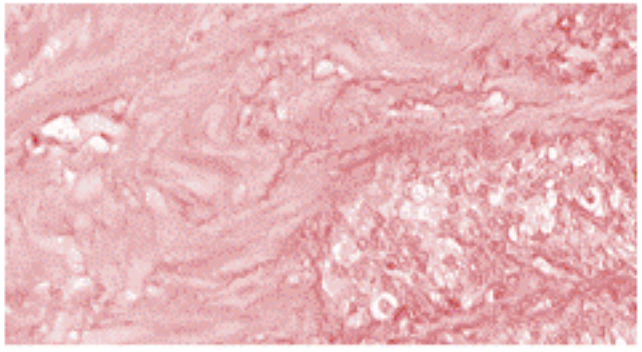


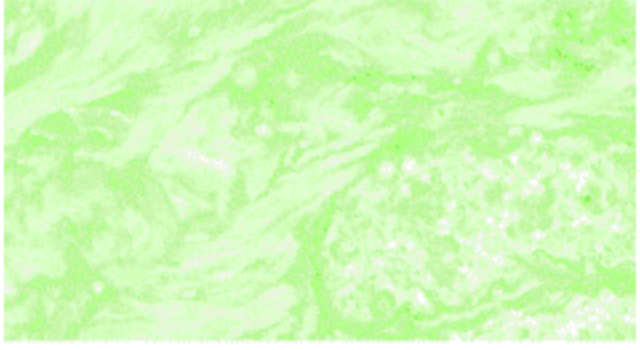
H&E Stain Estimation



=

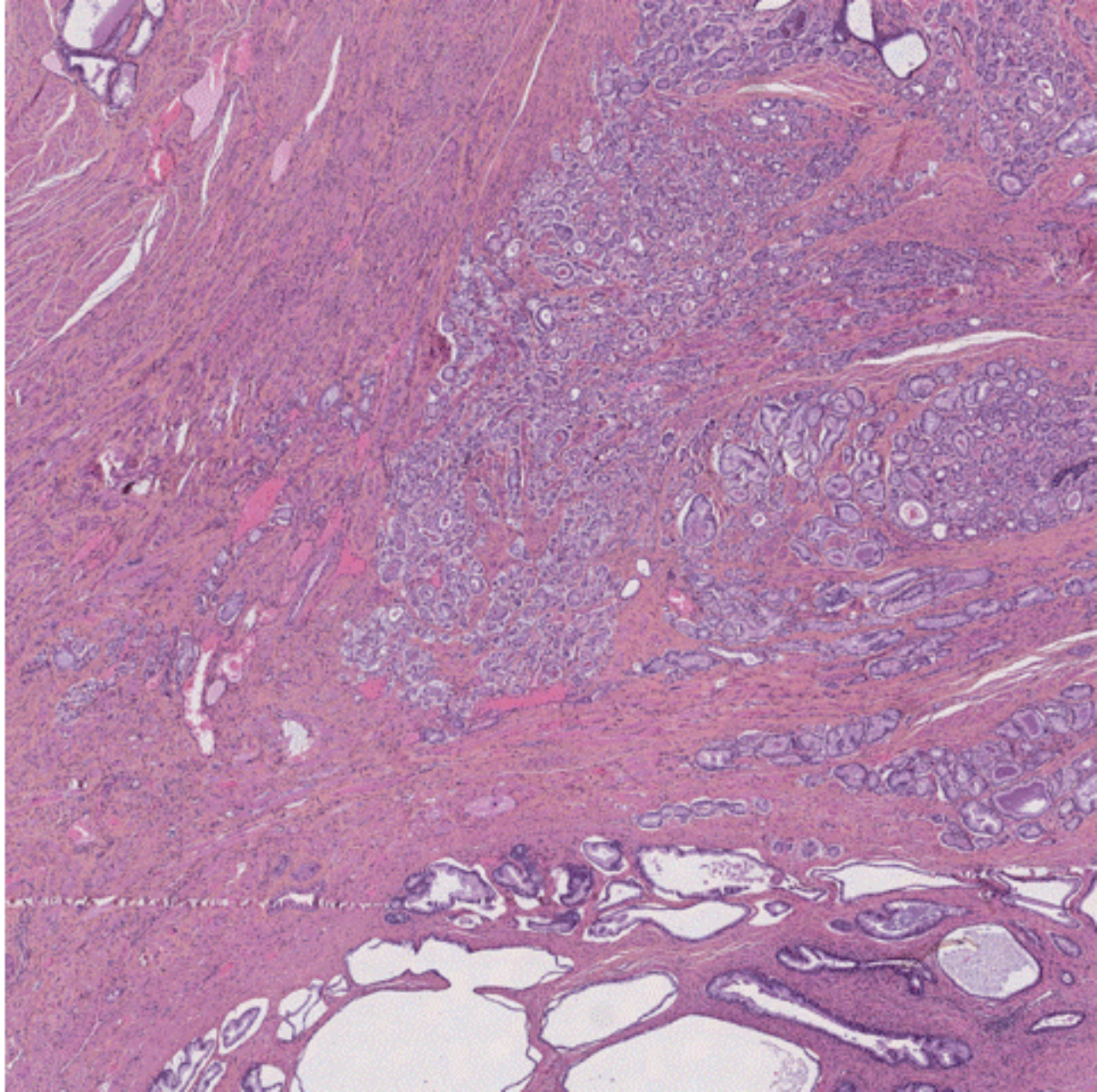




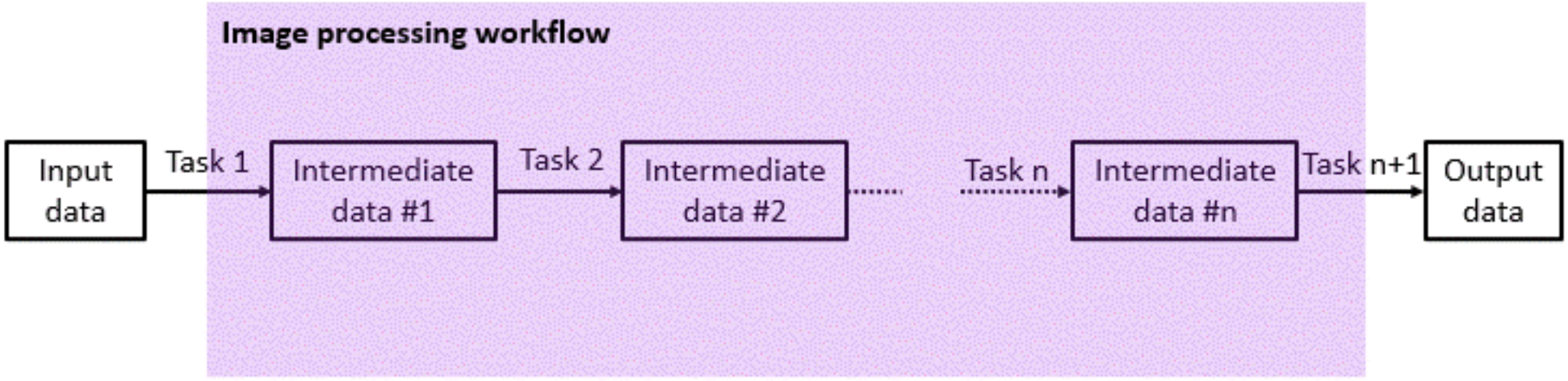


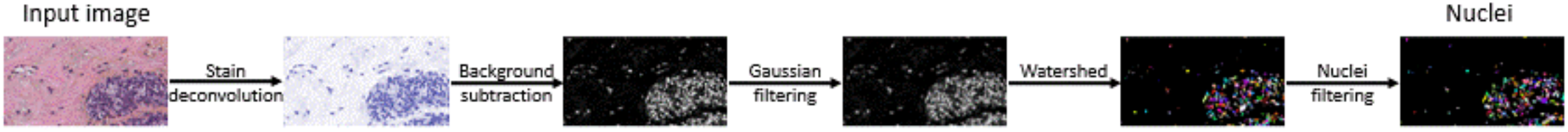
H&E Stain Estimation

- **Open** Prostate_1.ome.tif
- Create a small rectangle annotation and **estimate stain vectors**
- **Manually define Hematoxylin** and **Eosin** components
- **Visualize** the differences

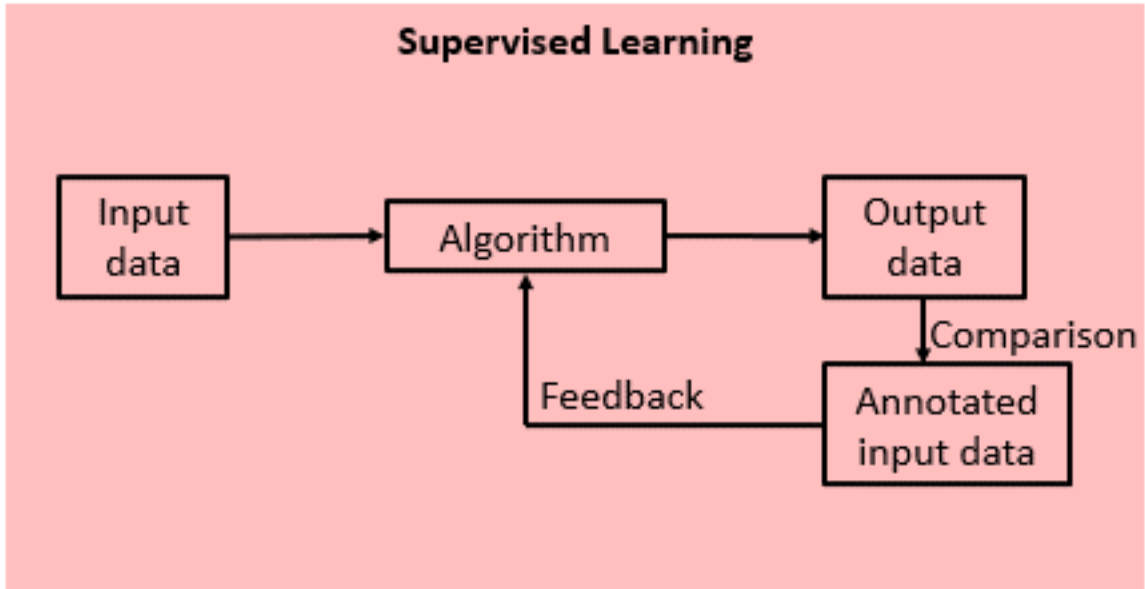


EXPLICIT PROGRAMMING

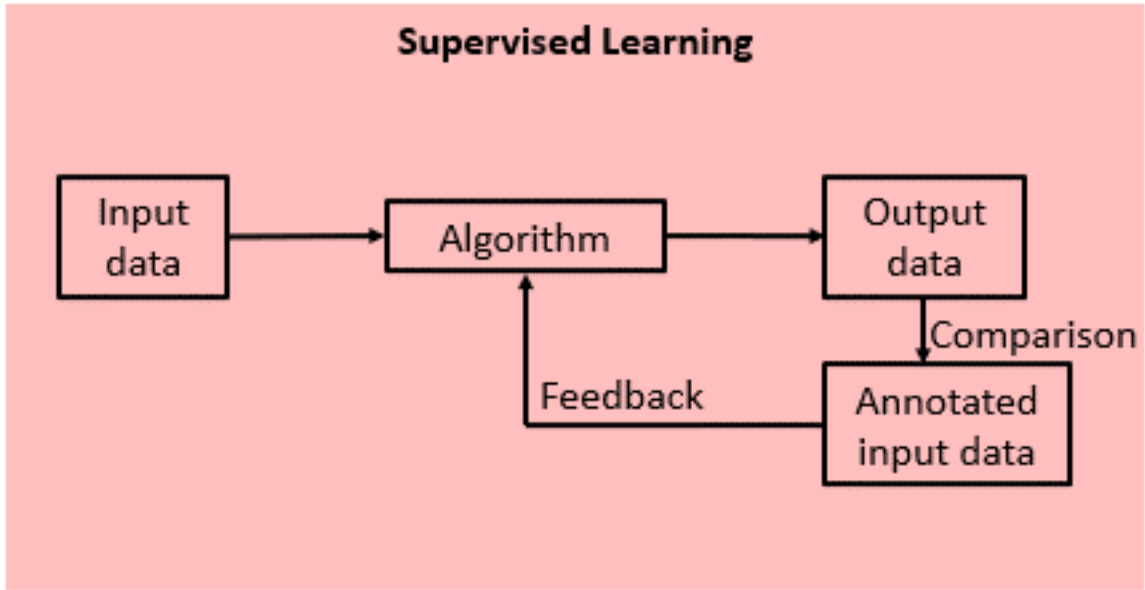


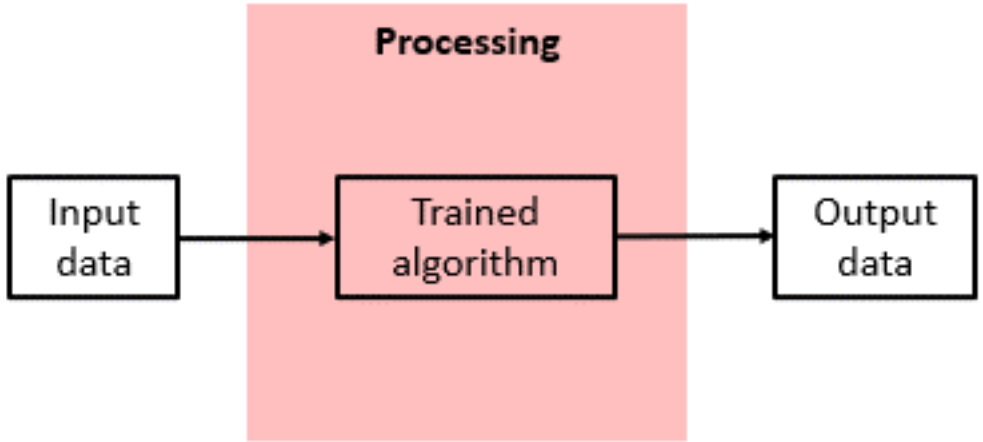


Supervised Machine Learning

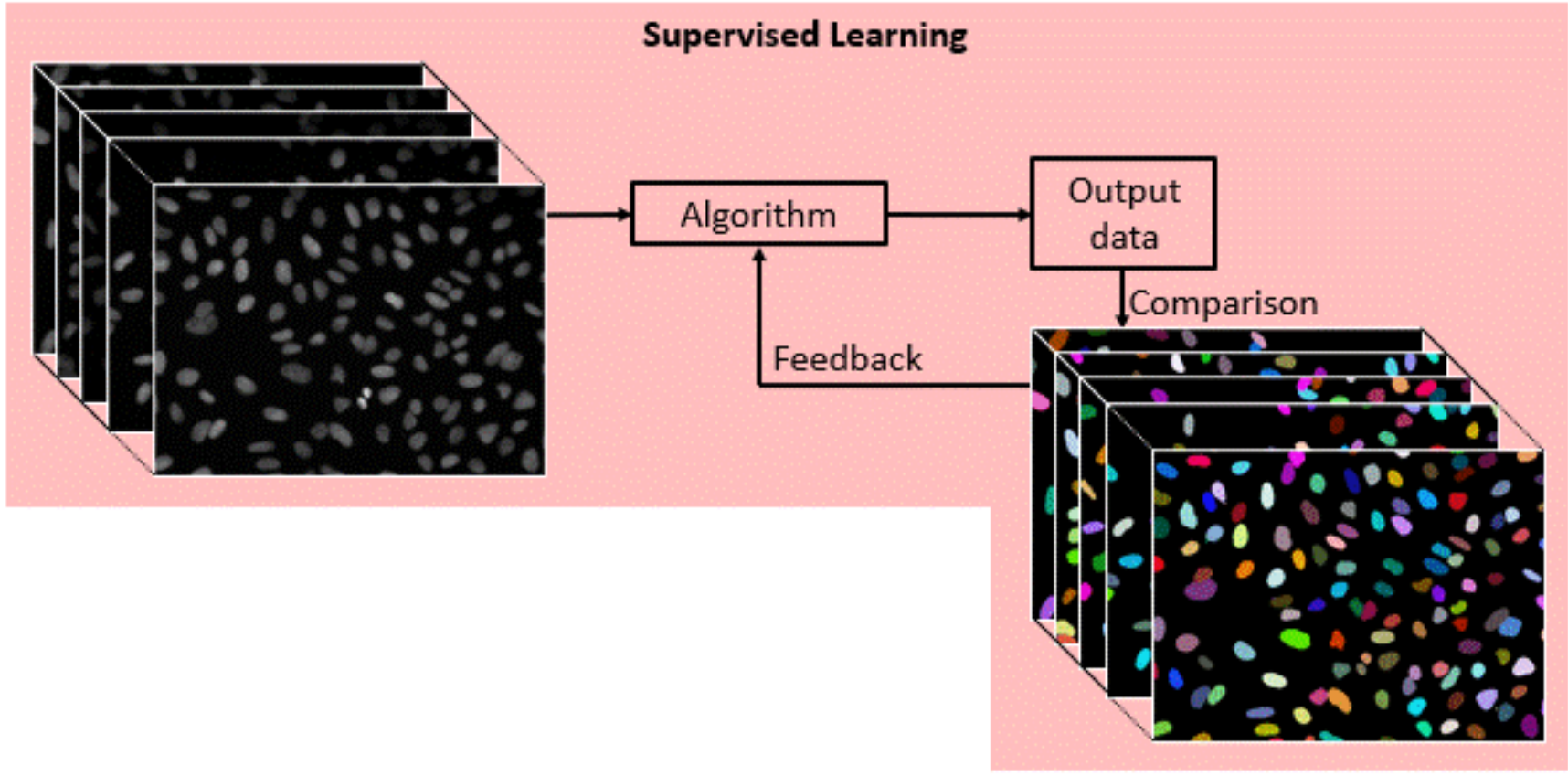


SUPERVISED MACHINE LEARNING

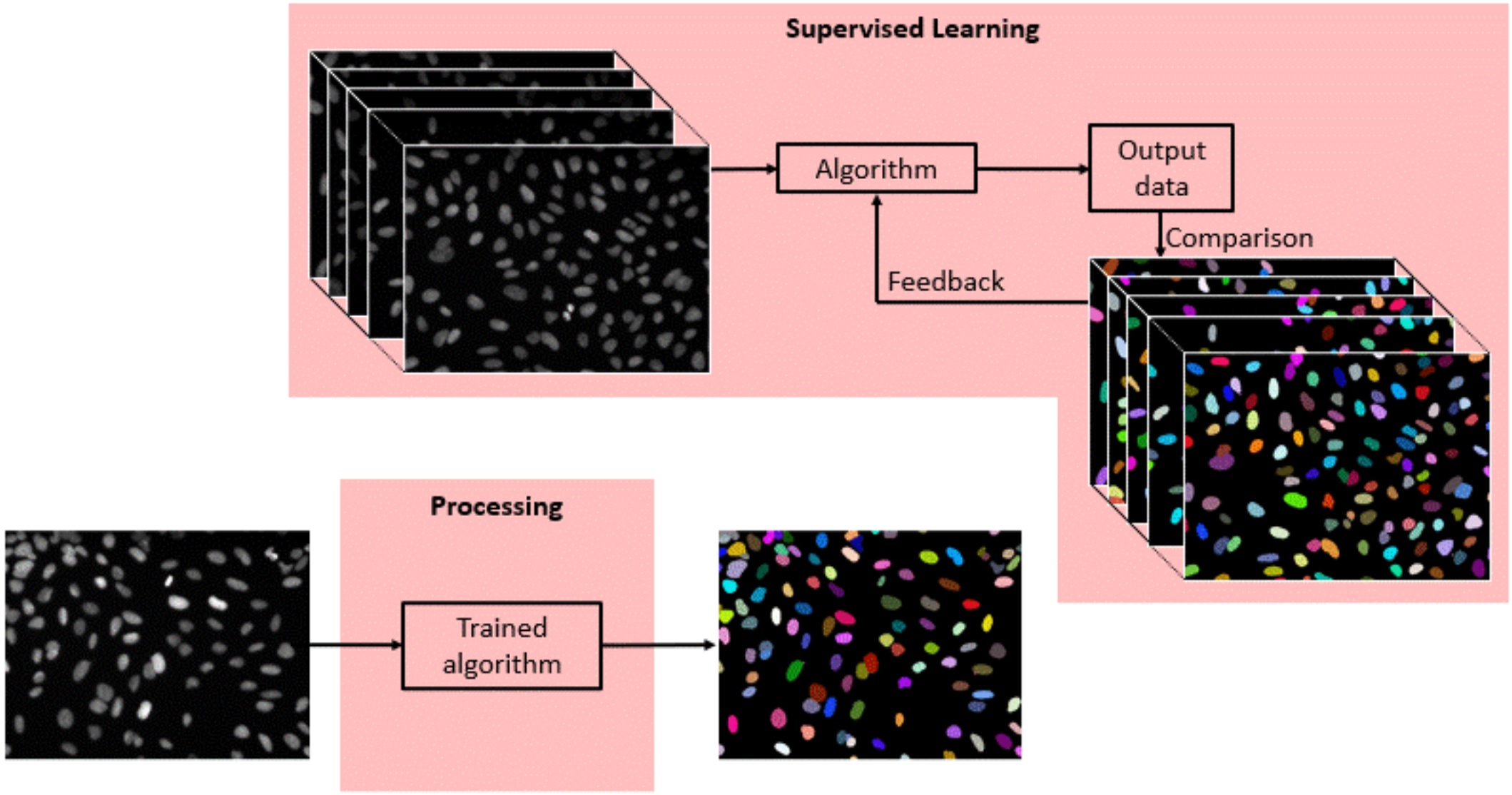




SUPERVISED MACHINE LEARNING



Supervised Machine Learning

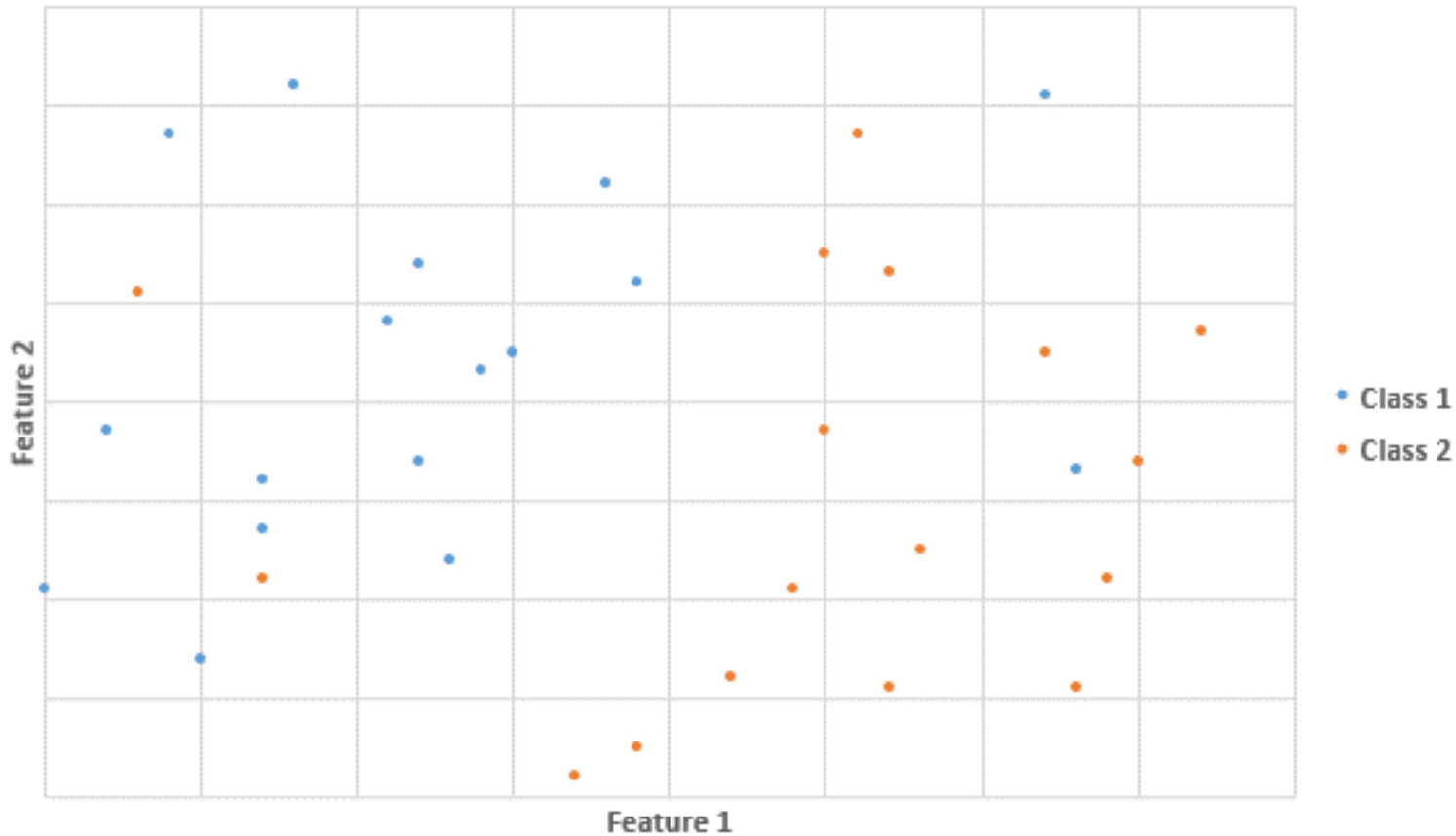


Shallow Machine Learning for Pixel Classification

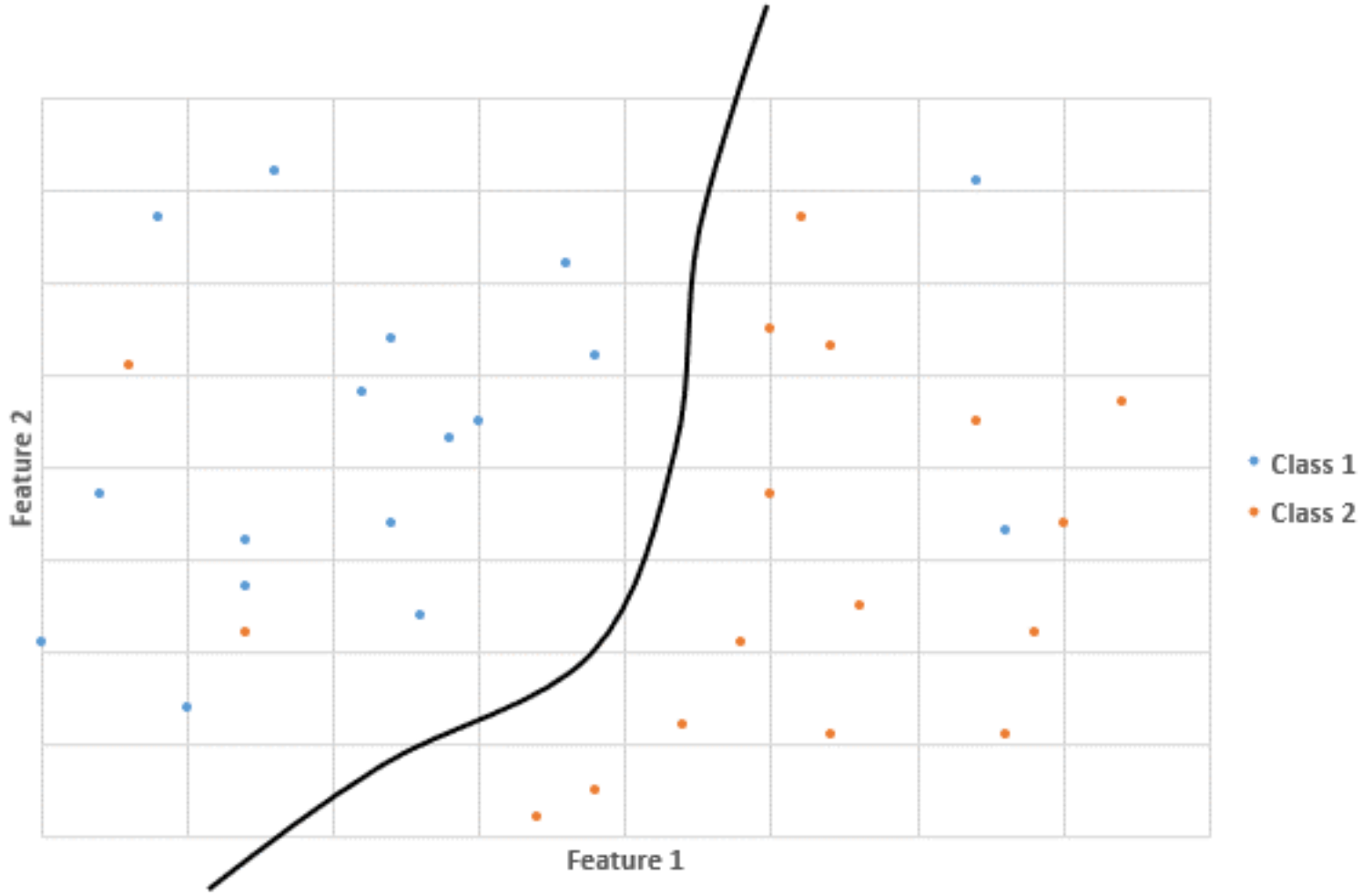
Supervised classification:

- **Examples** of classes are **manually** defined by the user
- A **classifier** is **trained** by using **defined features** with these examples
- Data is then **automatically classified** by using the trained classifier

Shallow Machine Learning for Pixel Classification

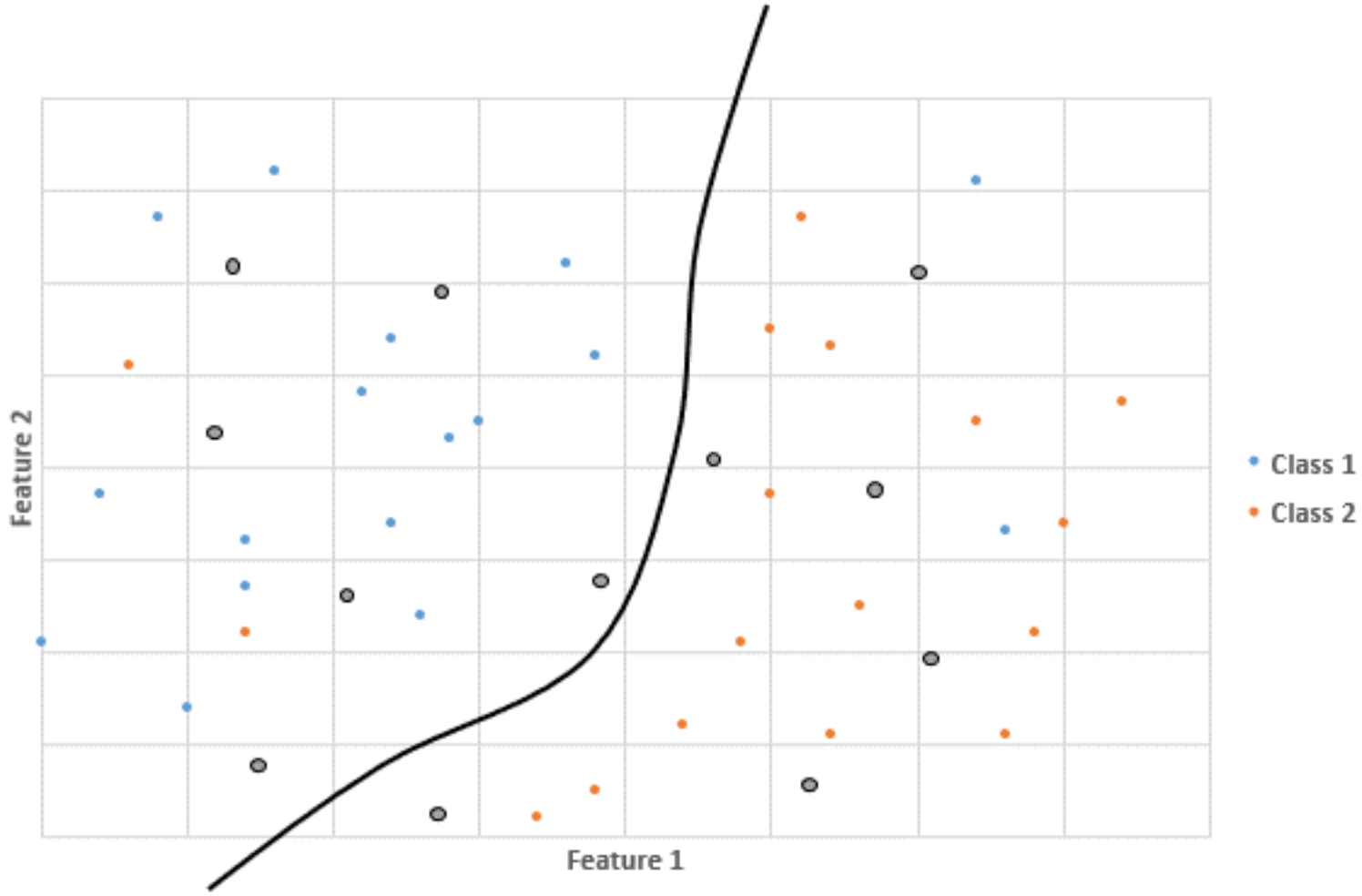


Shallow Machine Learning for Pixel Classification



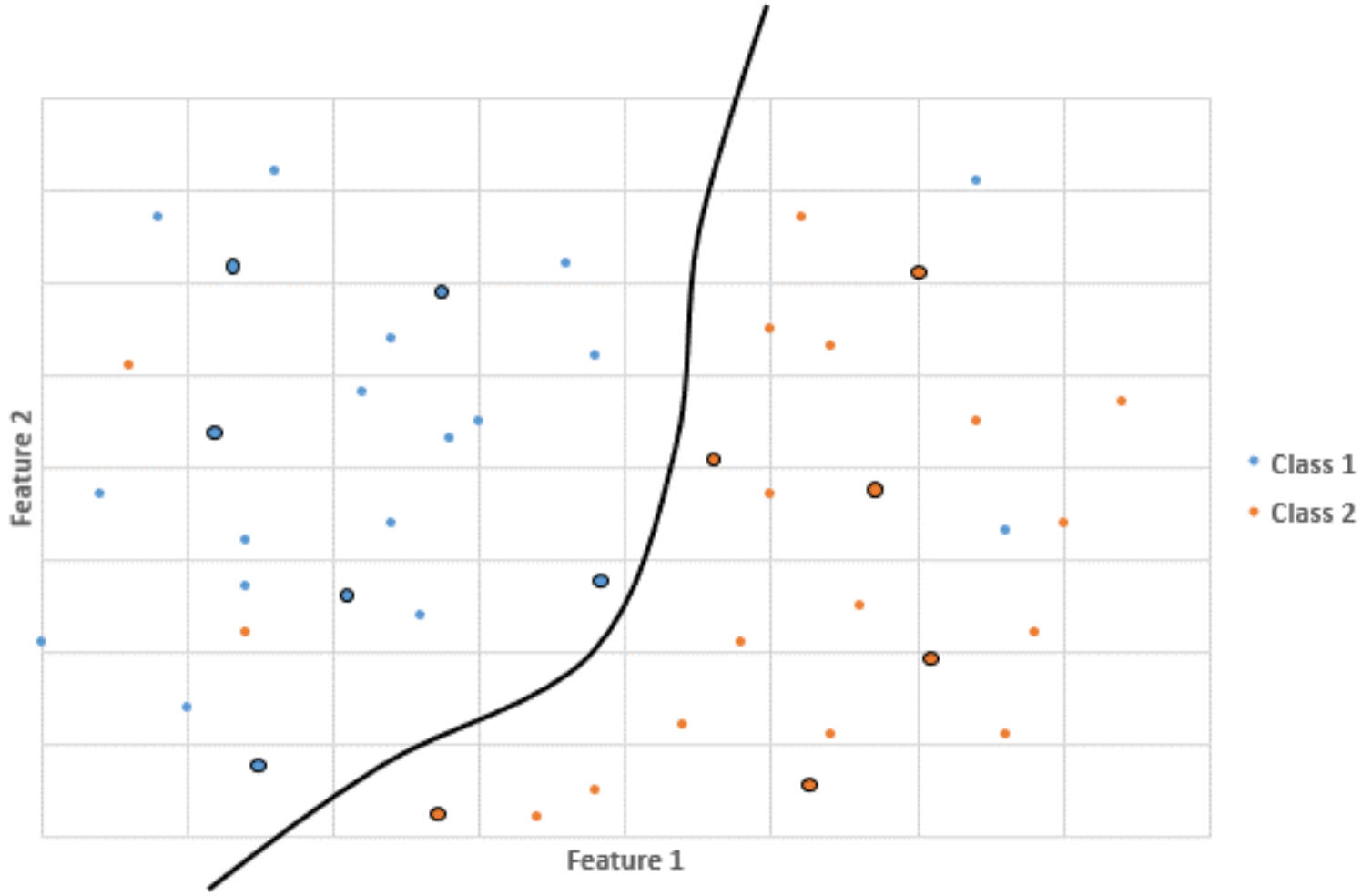
Training a classifier consists in **estimating** the **"best" separation** between classes

Shallow Machine Learning for Pixel Classification



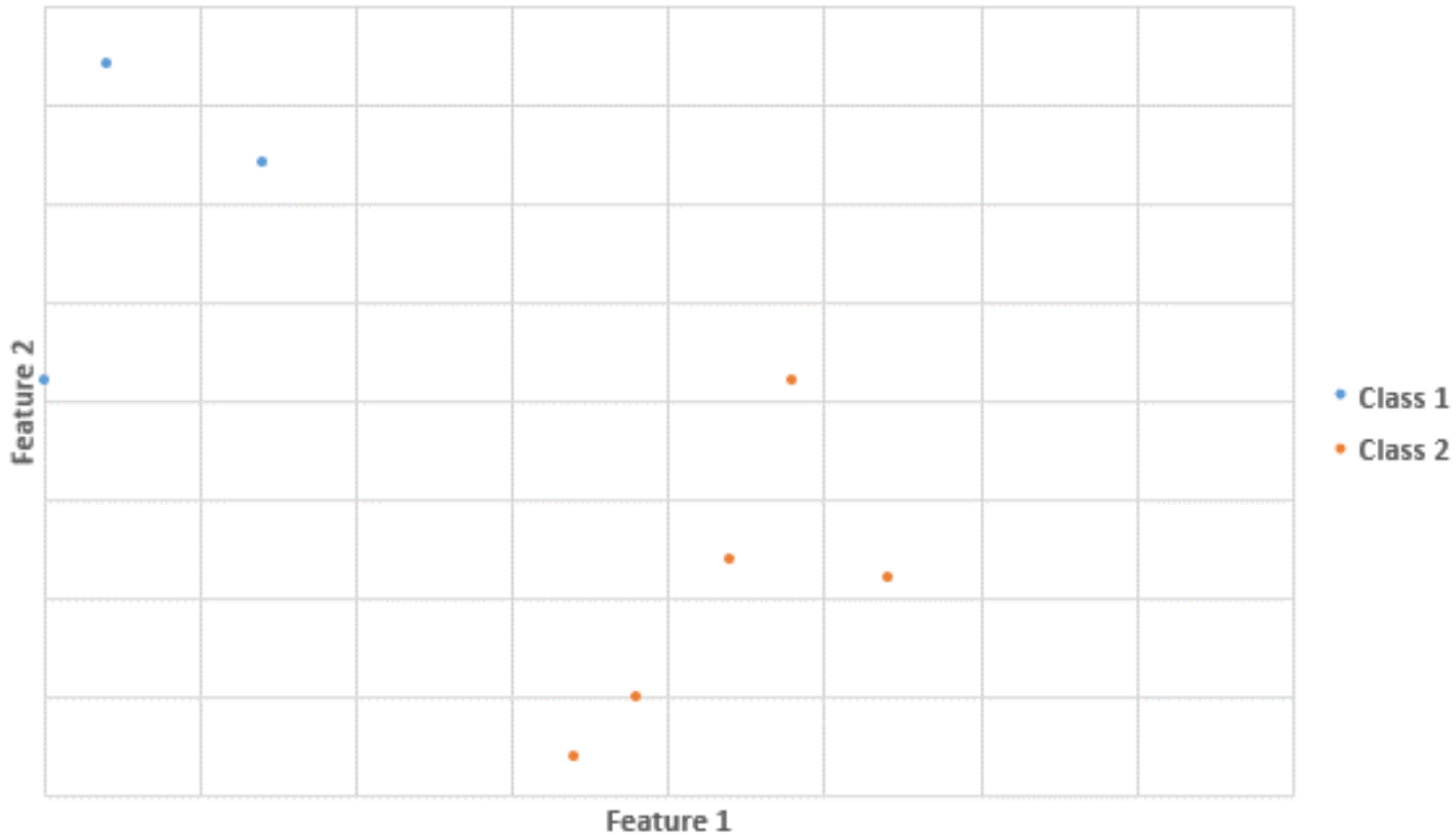
The **estimated class** for new data will be given by the **trained classifier**

Shallow Machine Learning for Pixel Classification



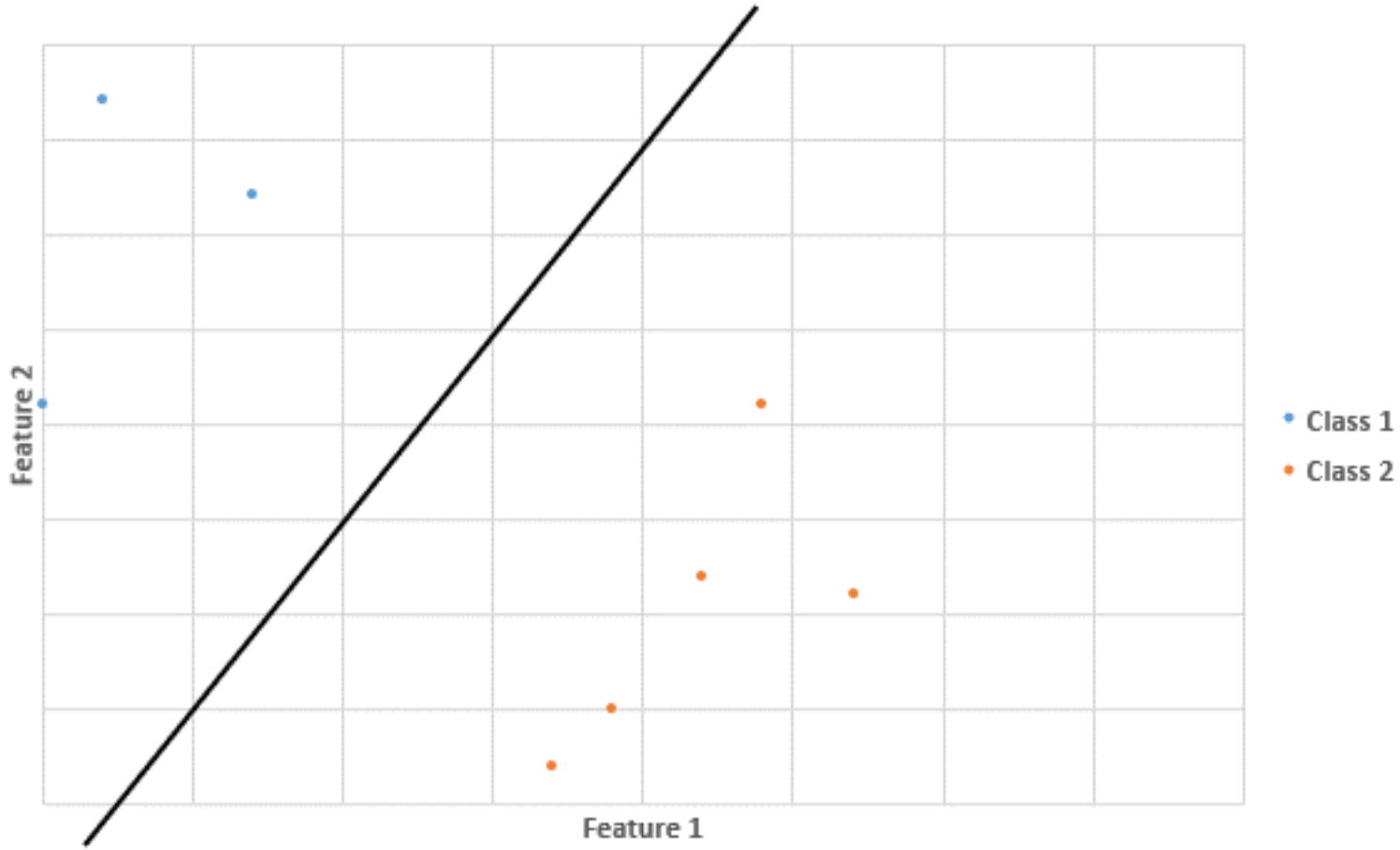
The **estimated class** for new data will be given by the **trained classifier**

Shallow Machine Learning for Pixel Classification



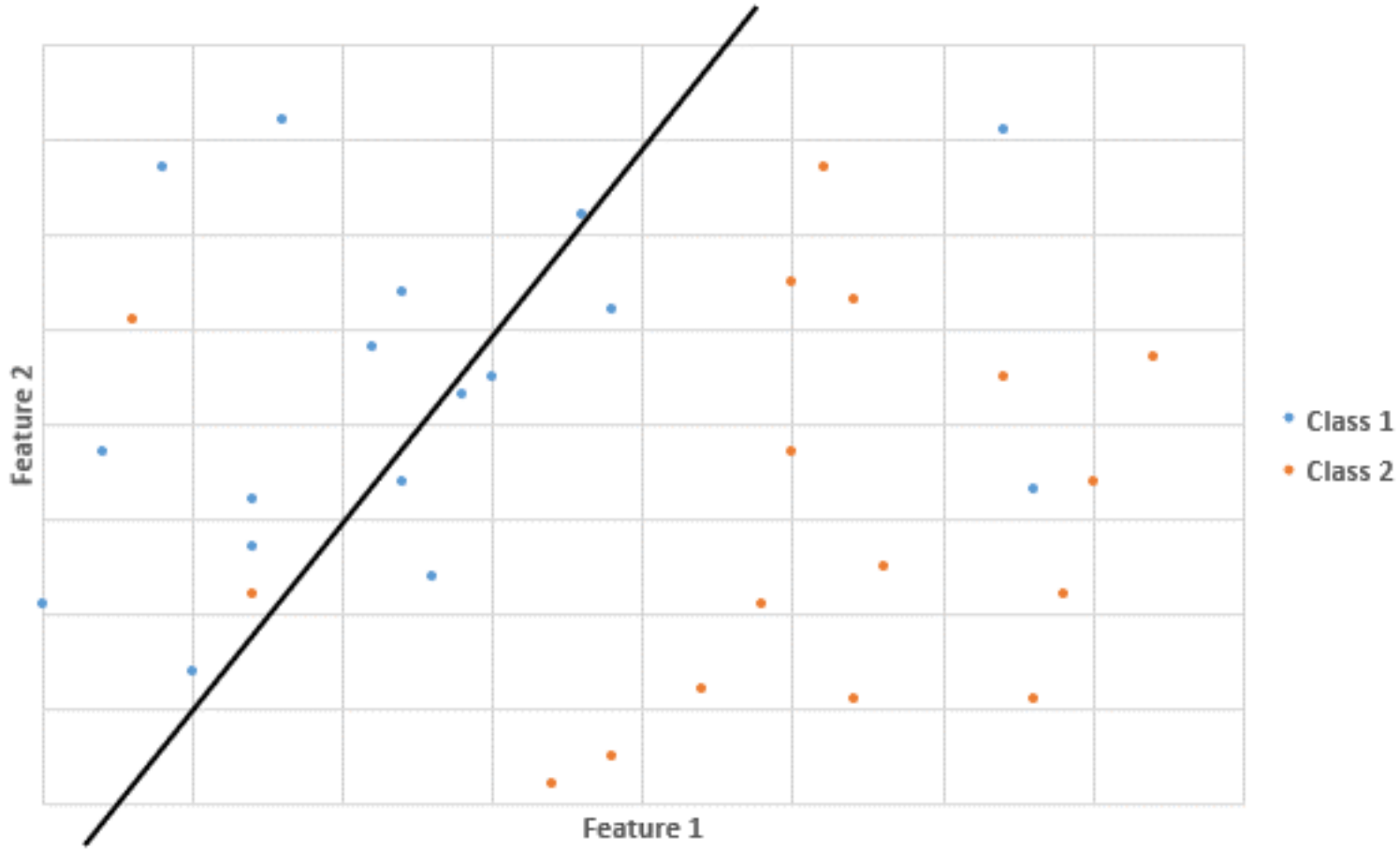
Training will be easier if **enough** and **representative** data is used

Shallow Machine Learning for Pixel Classification



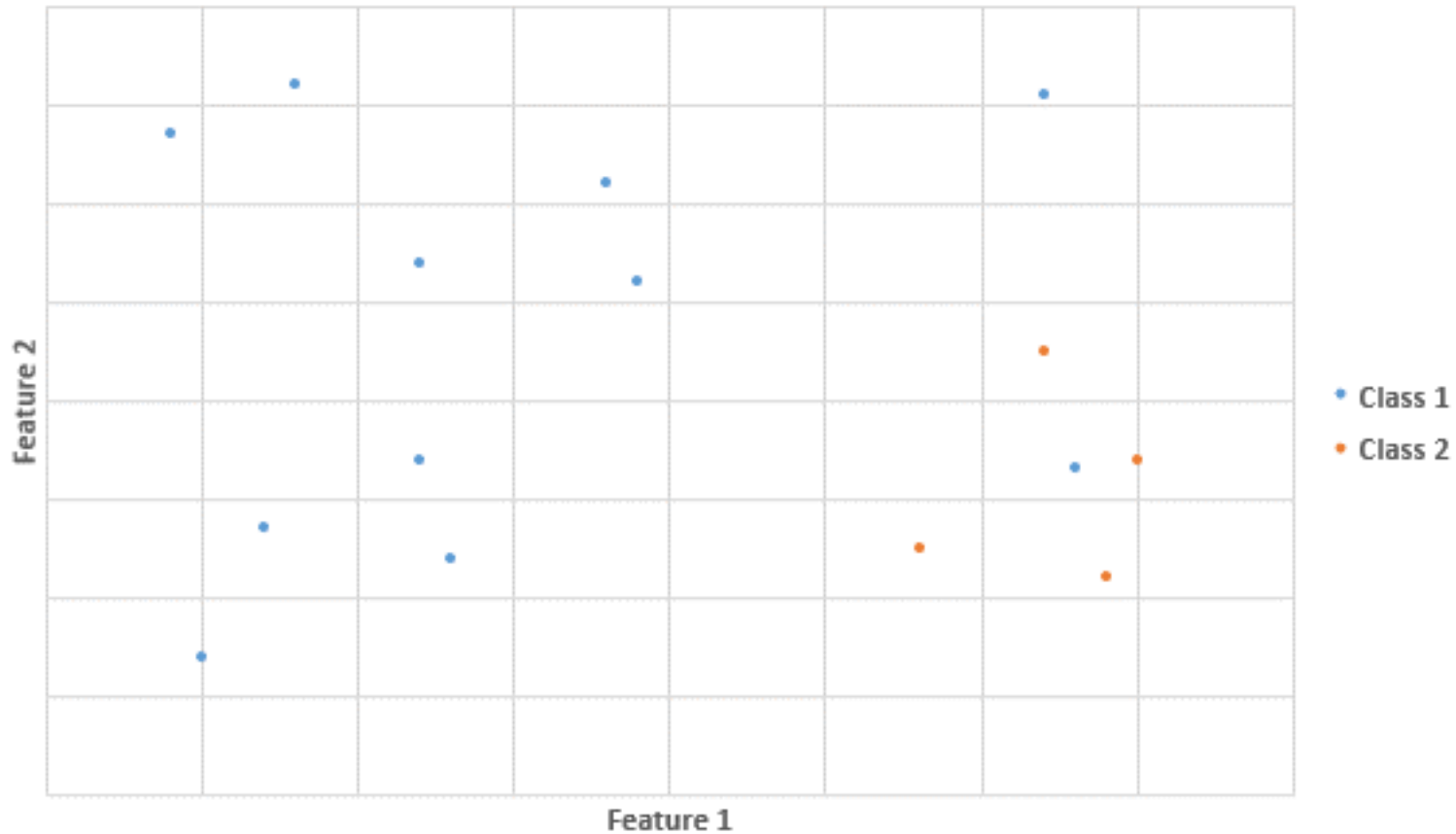
Training will be easier if **enough** and **representative** data is used

Shallow Machine Learning for Pixel Classification



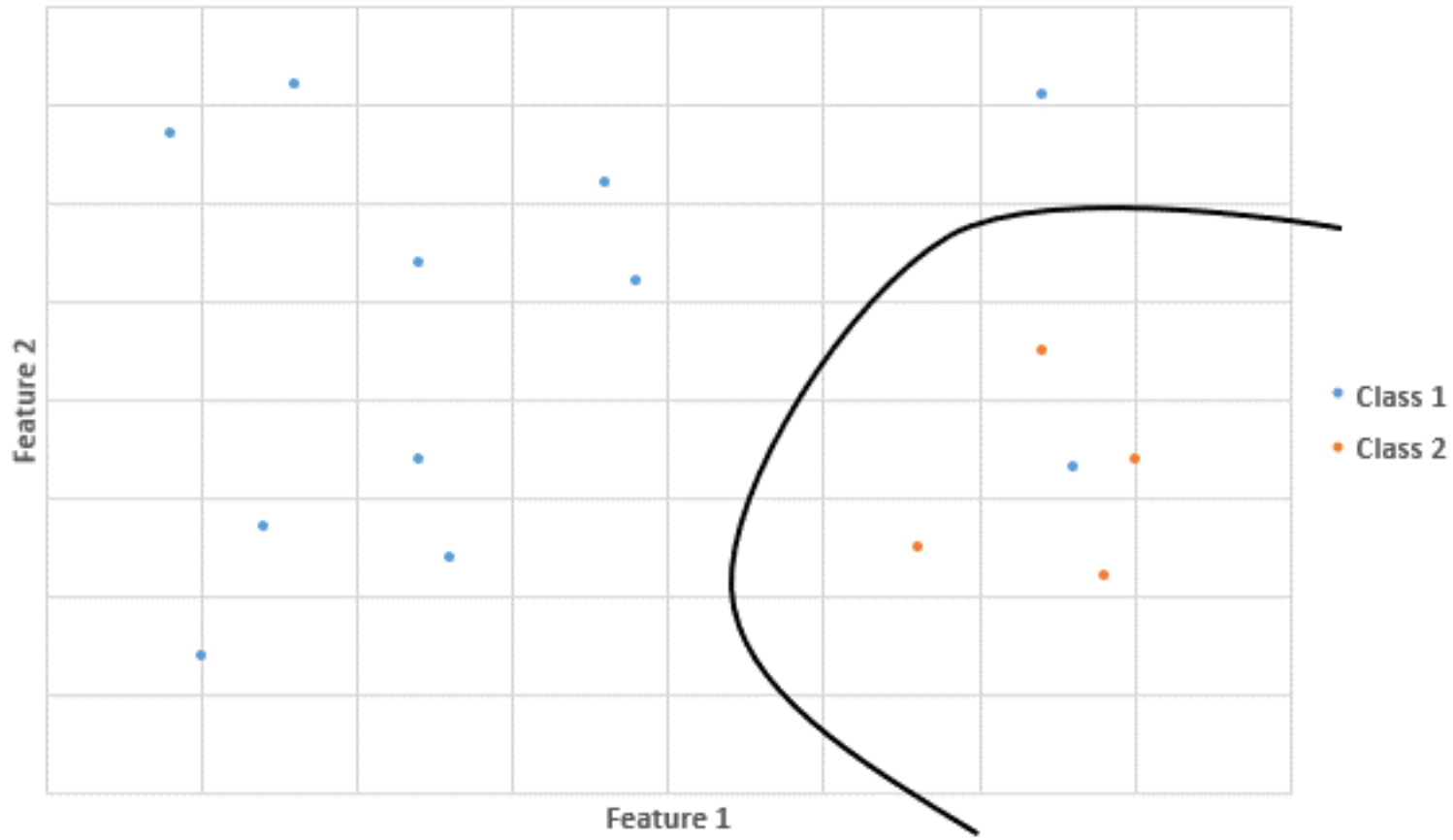
Training will be easier if **enough** and **representative** data is used

Shallow Machine Learning for Pixel Classification



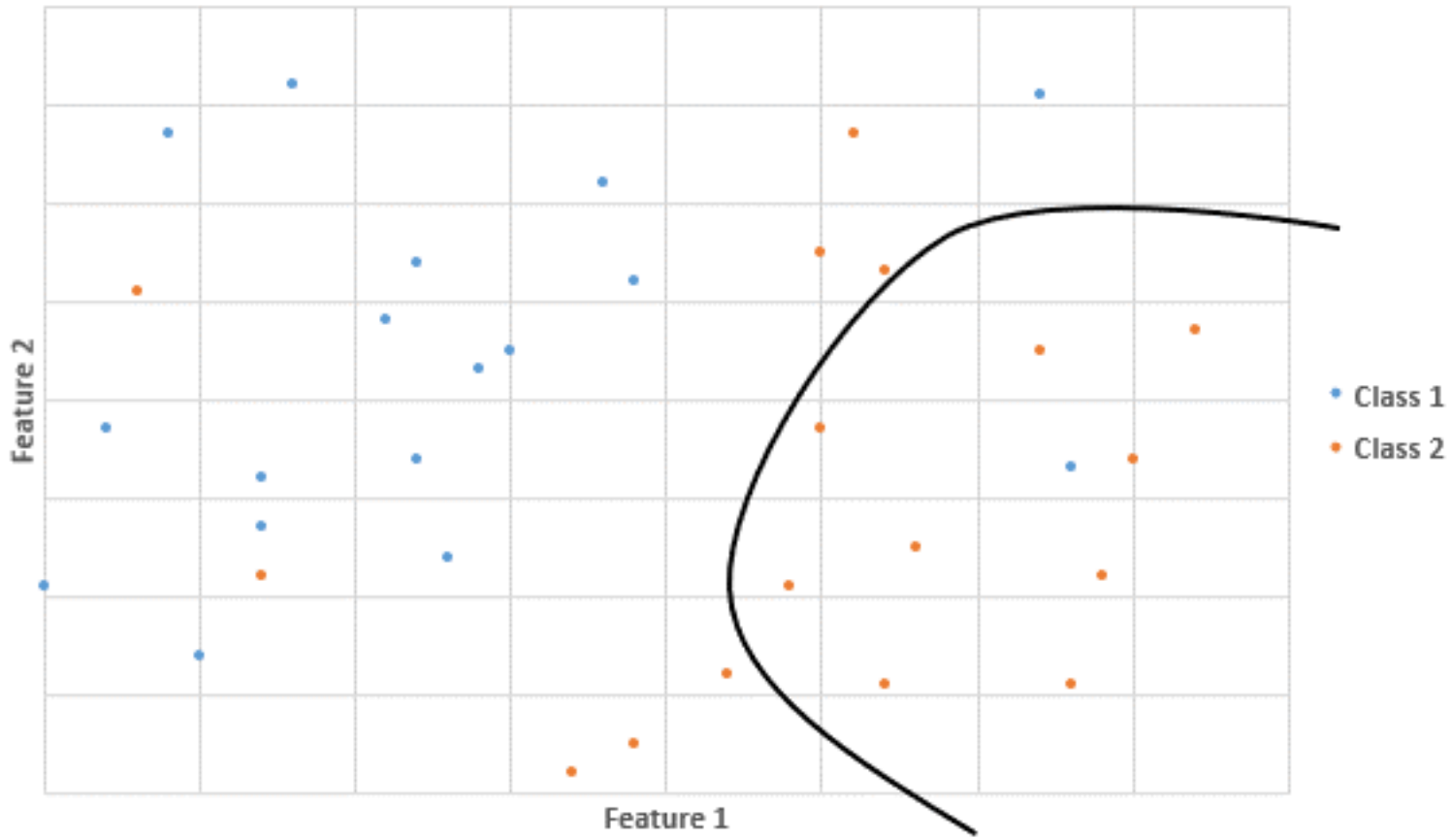
Class imbalance makes the training **more difficult**

Shallow Machine Learning for Pixel Classification



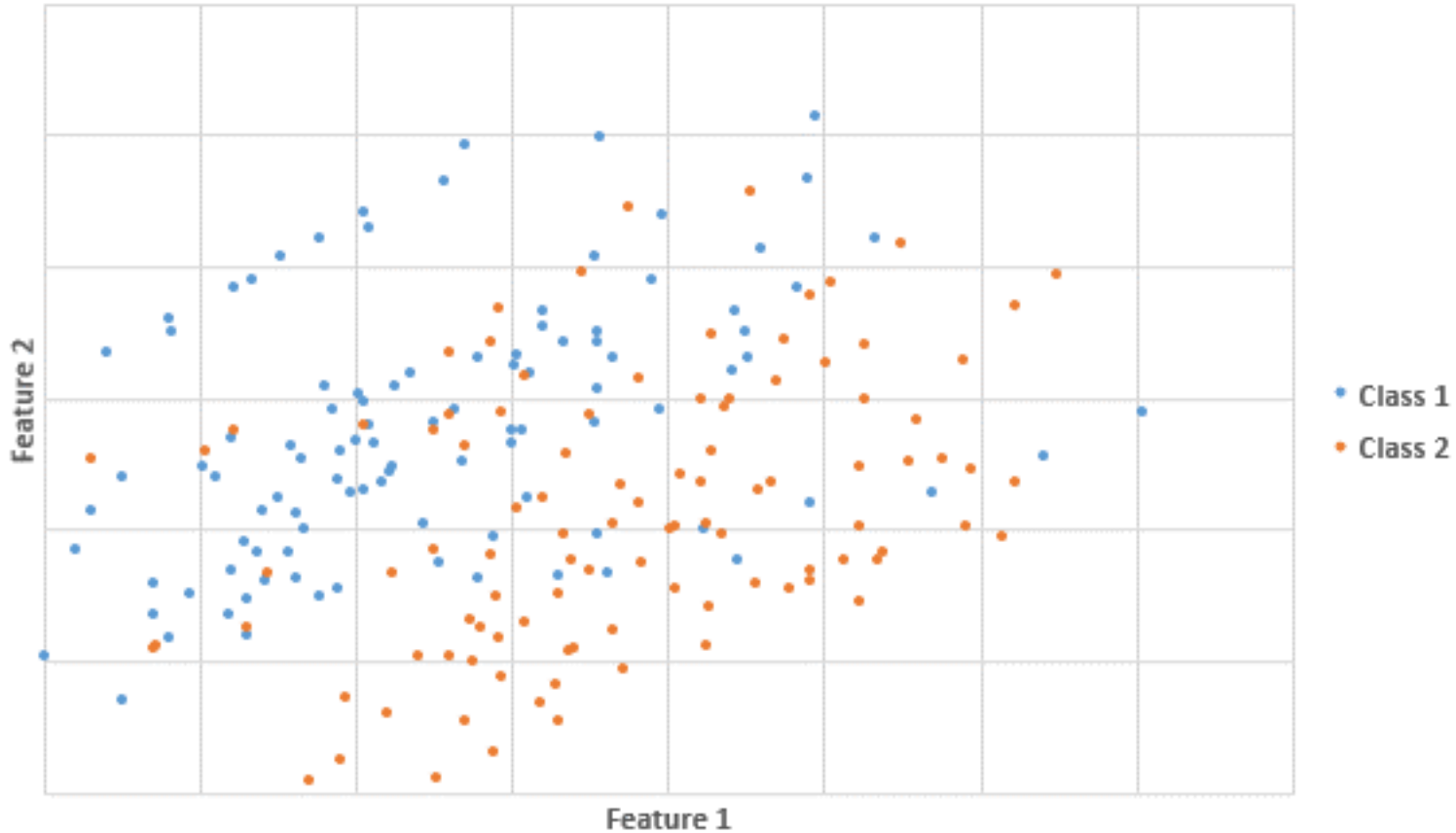
Class imbalance makes the training **more difficult**

Shallow Machine Learning for Pixel Classification



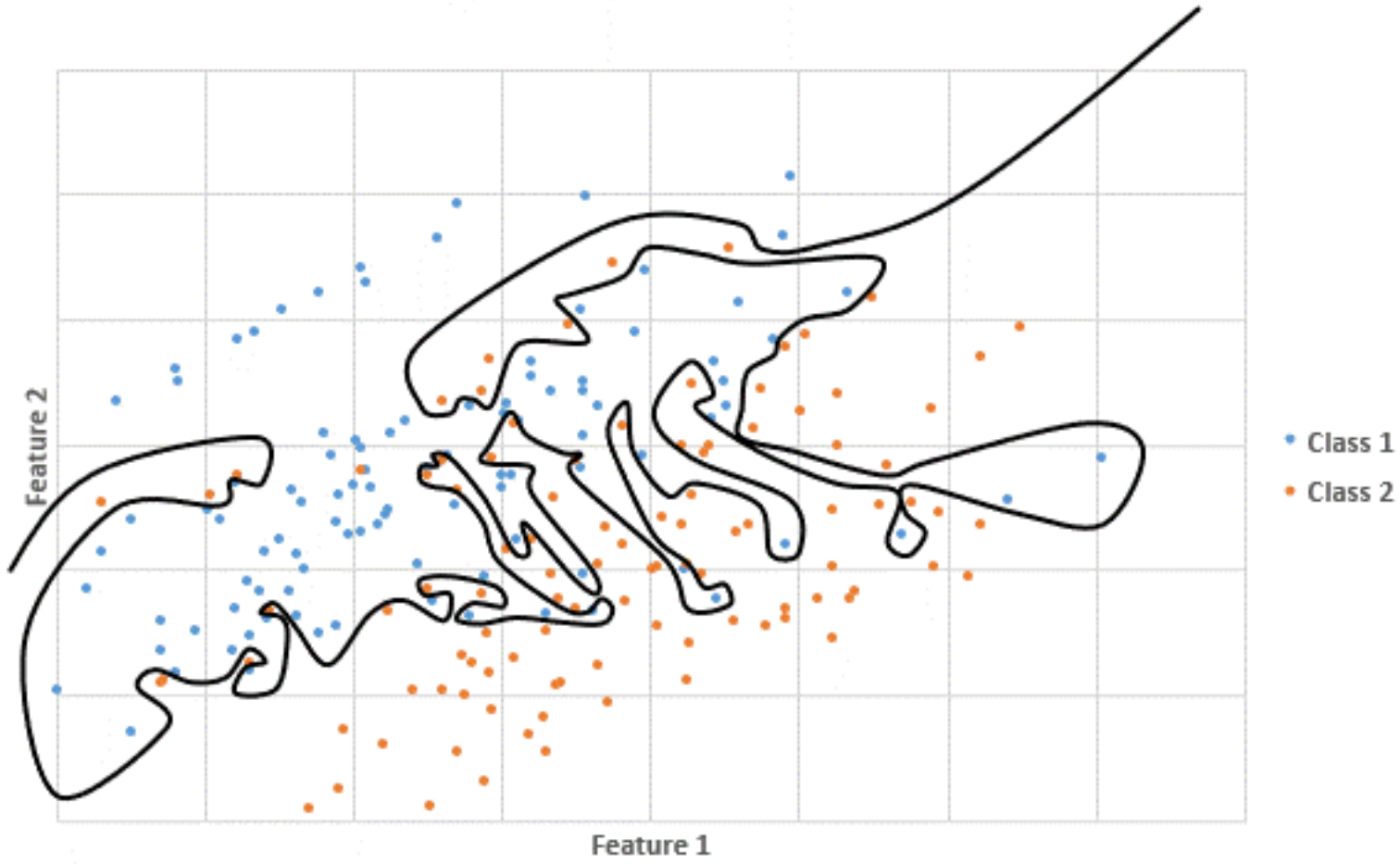
Class imbalance makes the training **more difficult**

Shallow Machine Learning for Pixel Classification



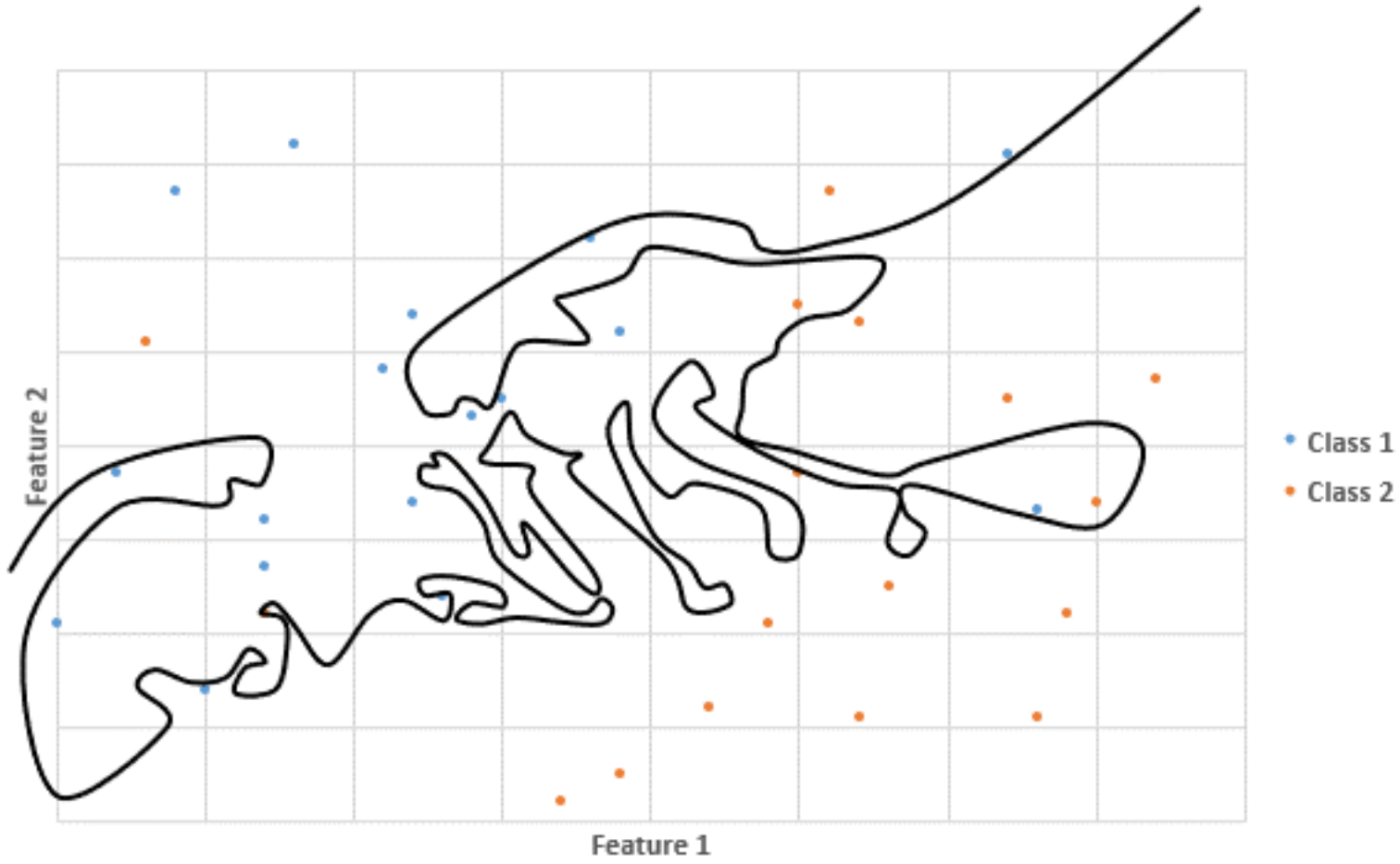
Too many annotations can lead to **over-fitting**: works great on training data but poorly on new data

Shallow Machine Learning for Pixel Classification



Too many annotations can lead to **over-fitting**: works great on training data but poorly on new data

Shallow Machine Learning for Pixel Classification



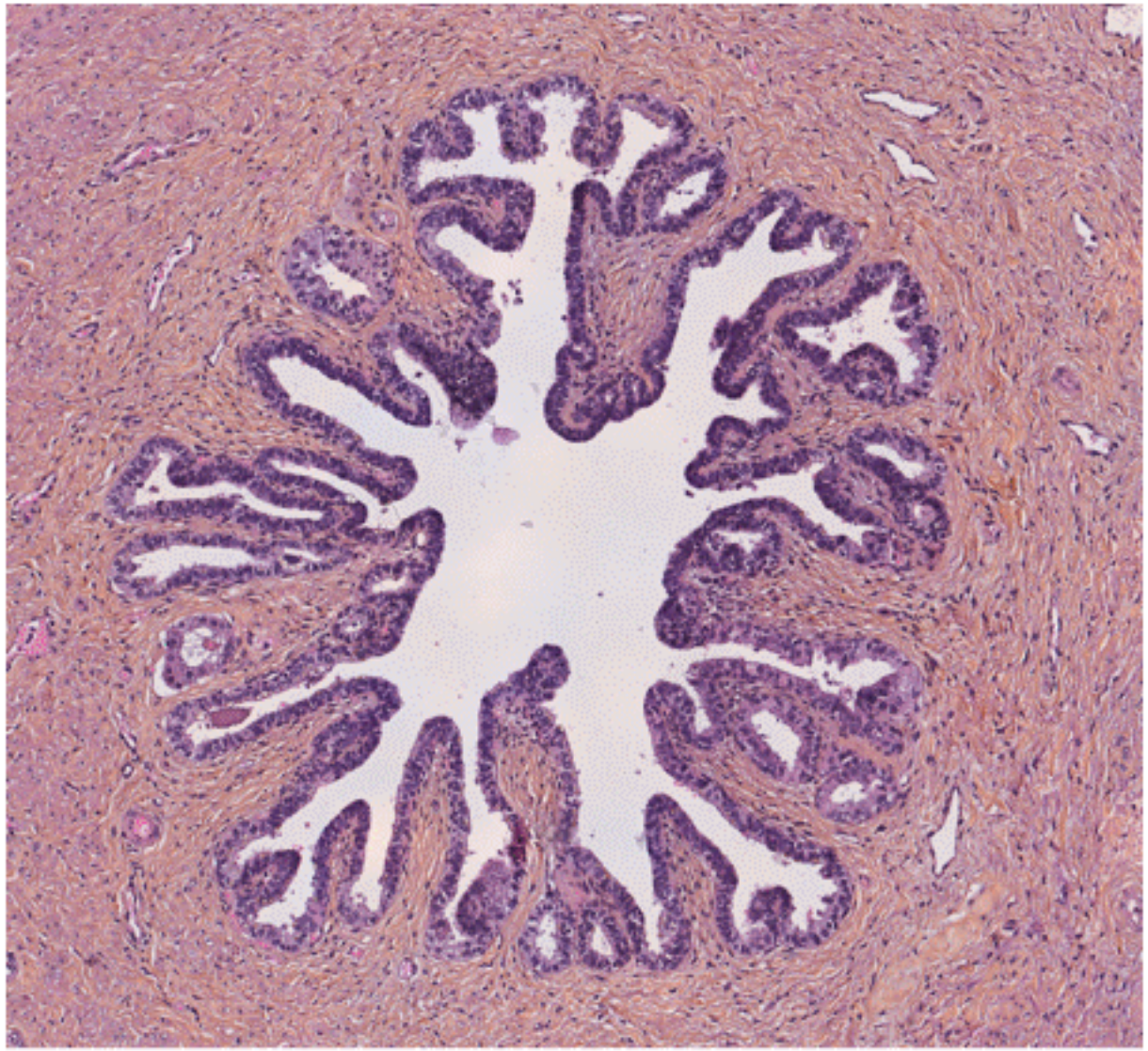
Too many annotations can lead to **over-fitting**: works great on training data but poorly on new data

Shallow Machine Learning for Pixel Classification

- Select **image features appropriate** to the classification problem
- Manually annotate regions/objects that are **representative** of what is seen in images
- Use **V** tool for annotations to **avoid over-representation**
- Define roughly the **same amount** of annotations for **each class**
- **Do not** manually annotate an **entire region of slide** to avoid over-fitting

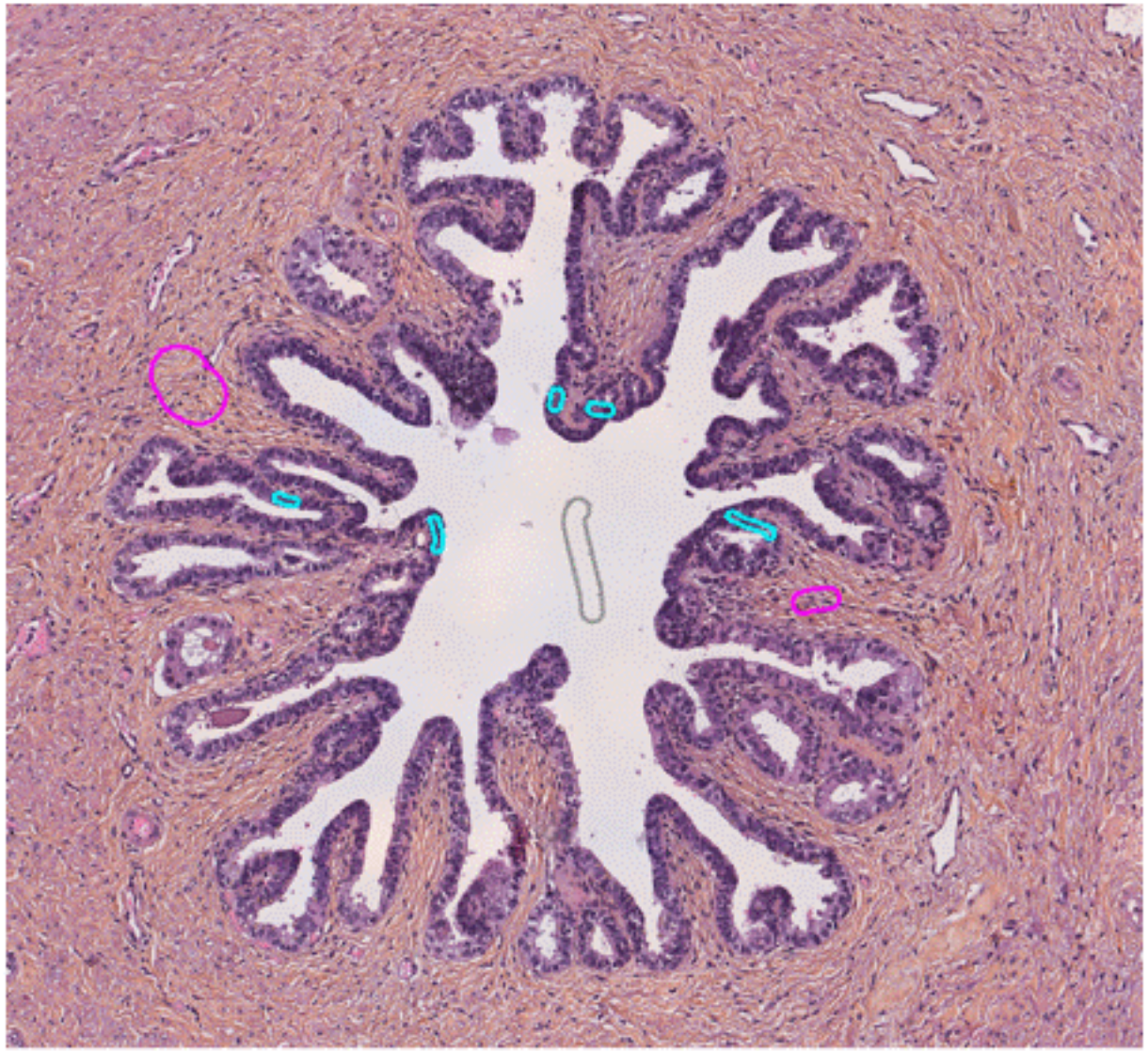
Pixel Classification

Find **regions** corresponding to **epithelium**, **stroma** and **background**

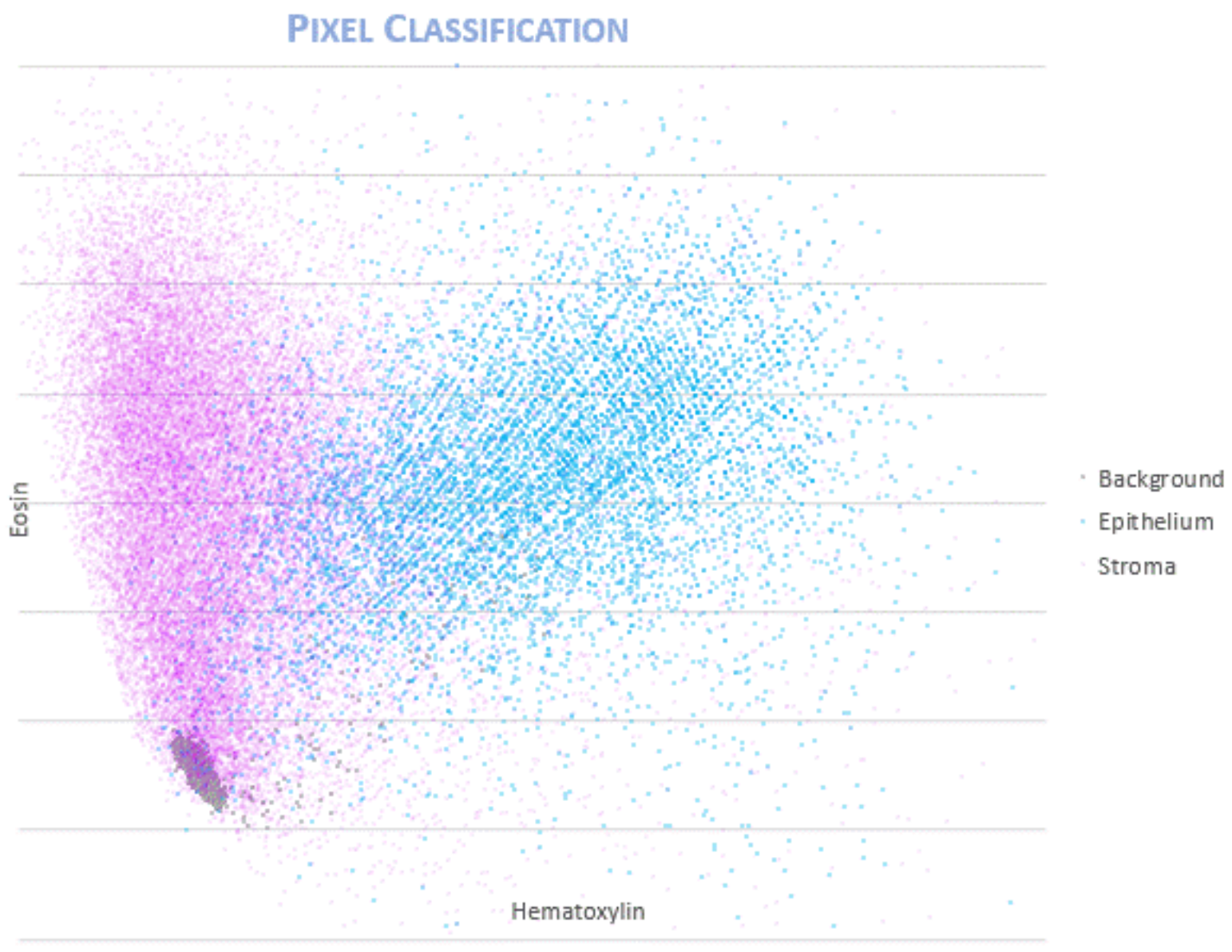


Pixel Classification

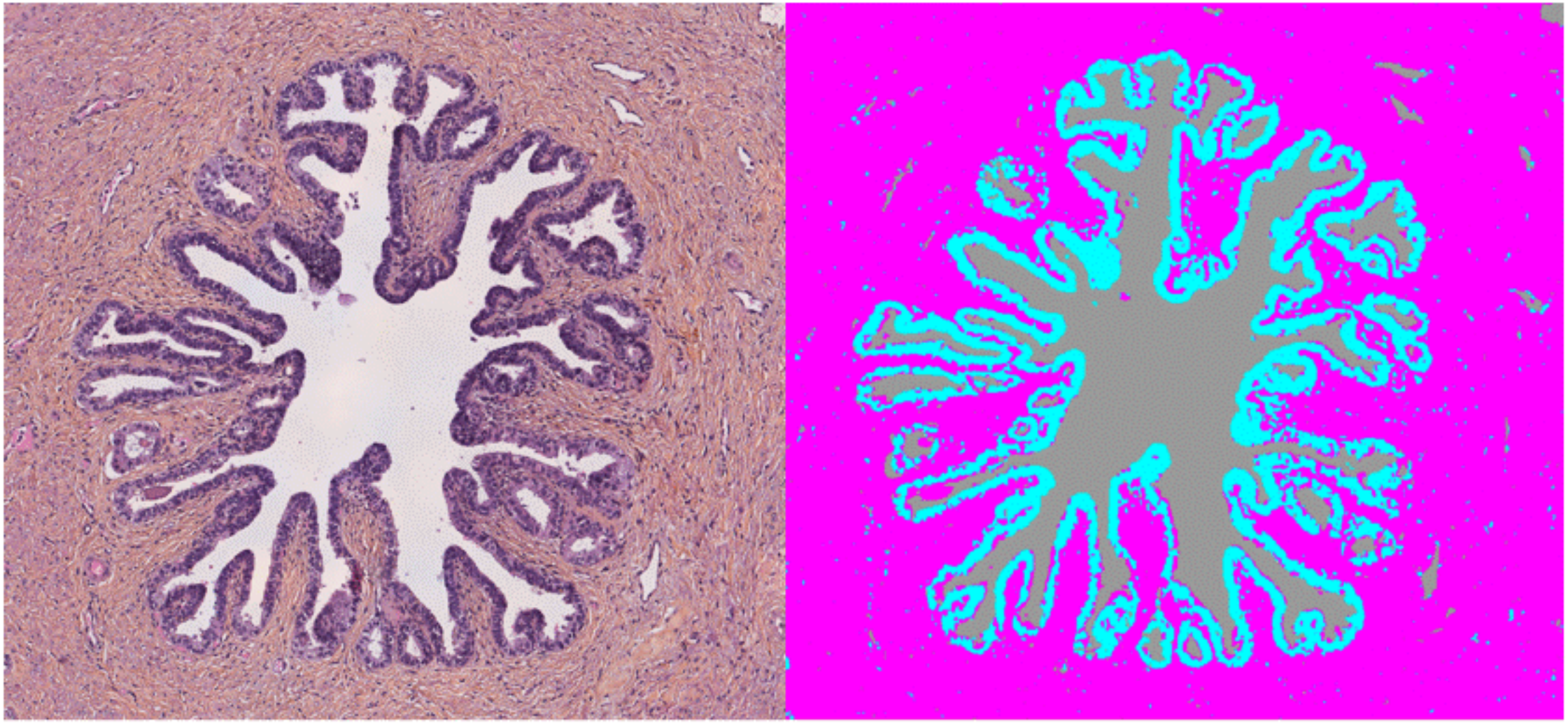
Find **regions** corresponding to **epithelium**, **stroma** and **background**



Find **regions** corresponding to **epithelium**, **stroma** and **background**

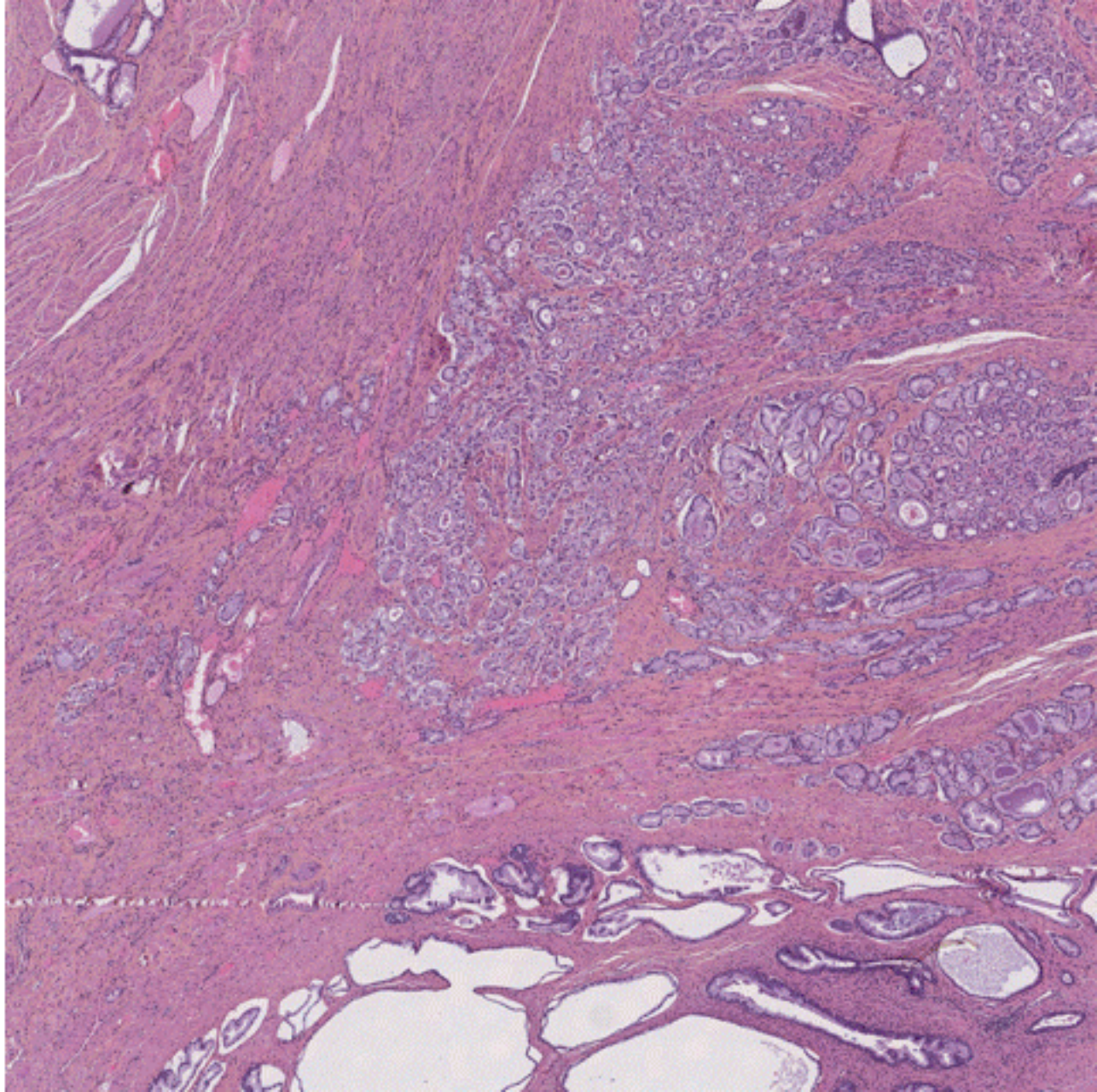


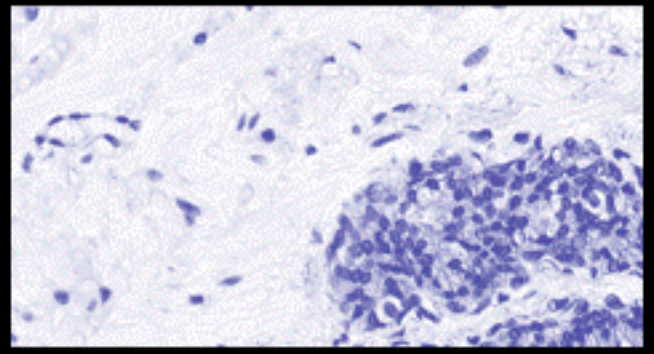
Pixel Classification



Pixel Classification

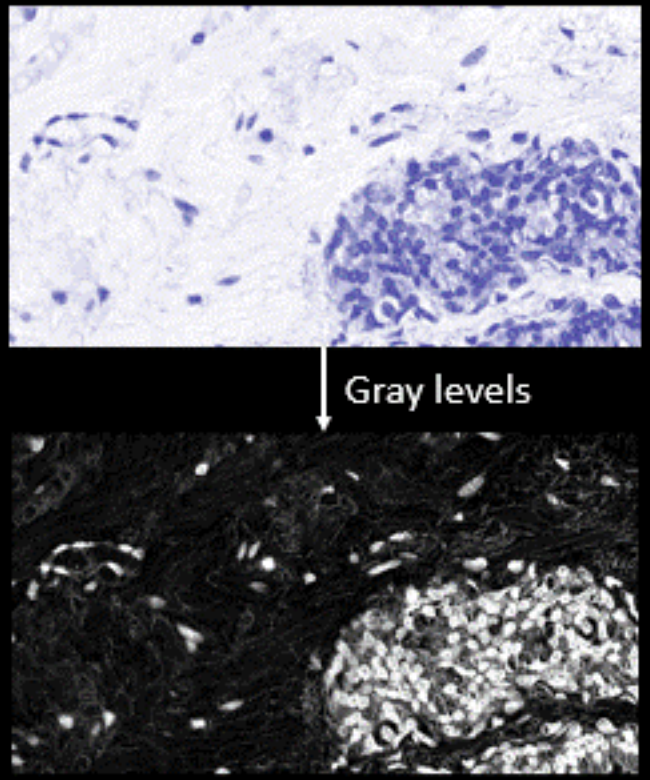
- **Open** prostate_1.ome.tif, prostate_2.ome.tif, prostate_3.ome.tif, prostate_4.ome.tif, prostate_5.ome.tif and prostate_6.ome.tif
- Create **annotations** in each image that recapitulate the **diversity** of the tissue
- Create **regions annotations**
- Open **"Pixel classifier"**
- **Annotate** pixels belonging to **background, epithelium** and **stroma**
- **Save classifier** and apply it to **each image** with a **script** (workflow tab)
- Get **proportions** of **tissues**



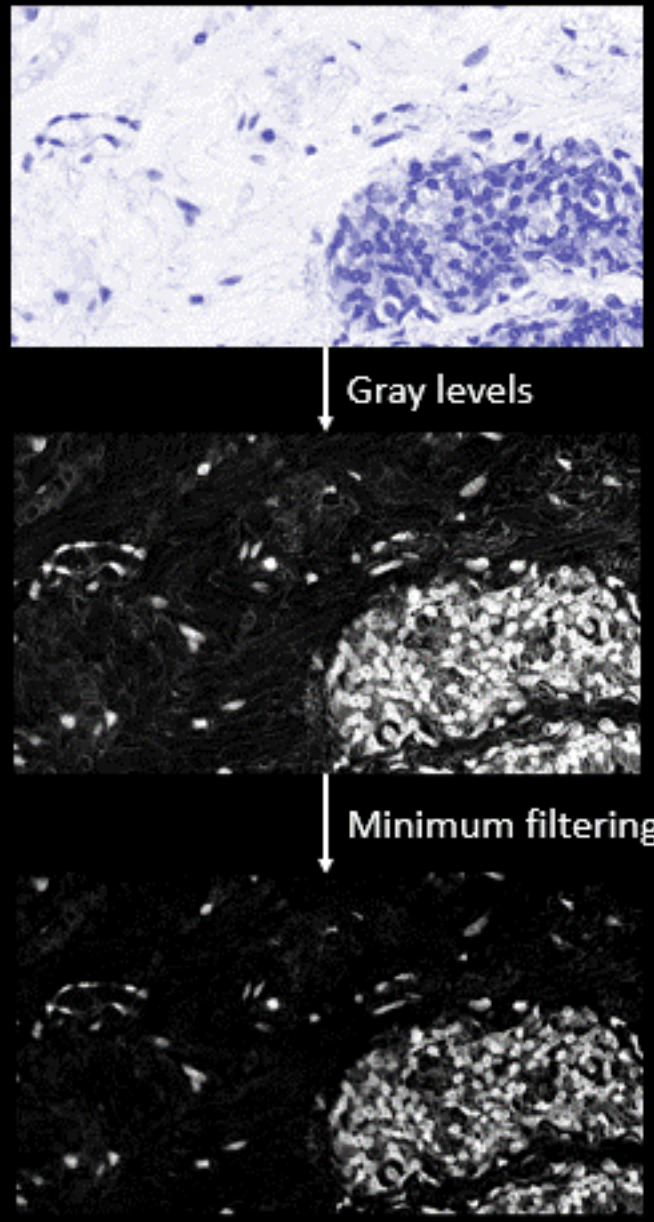


Cell Detection

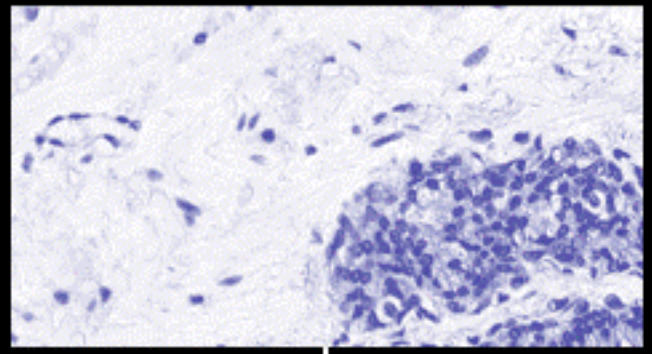
Cell Detection



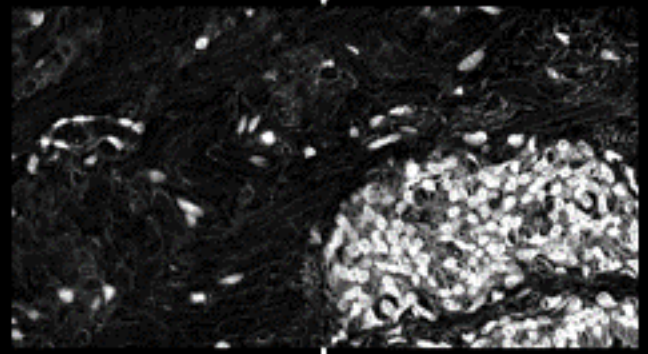
Cell Detection



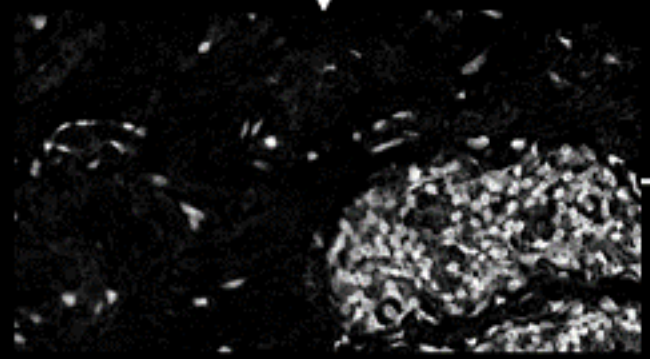
Cell Detection



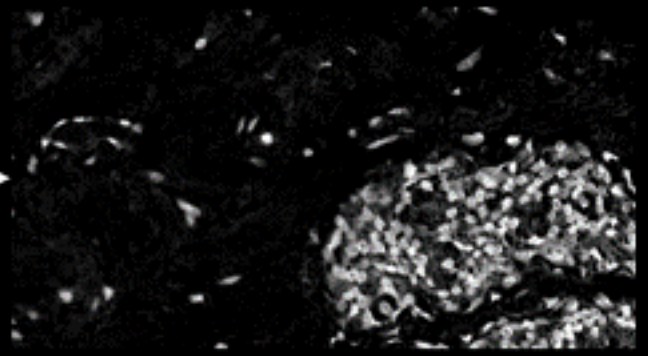
Gray levels



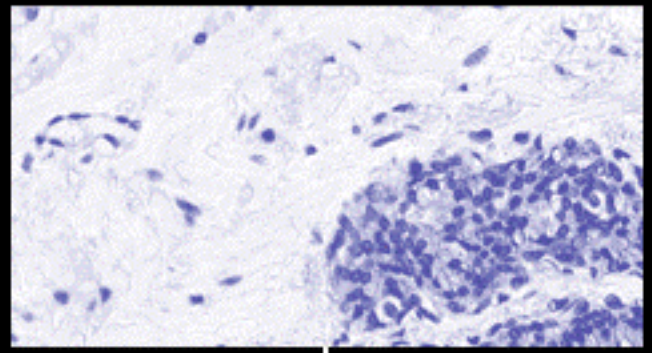
Minimum filtering



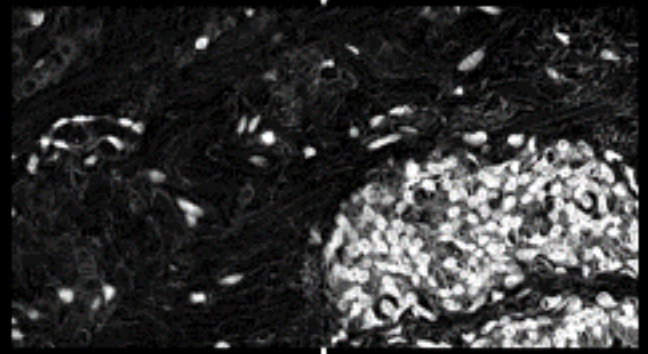
Gaussian blur



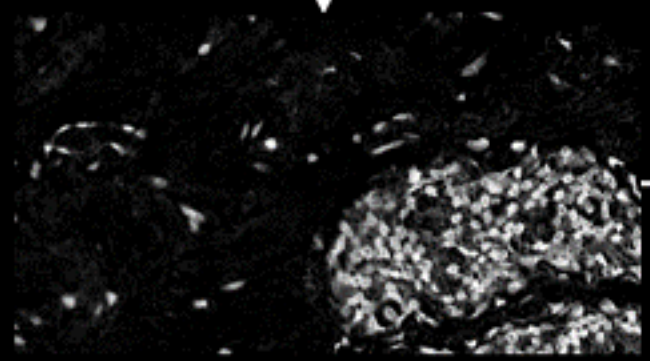
Cell Detection



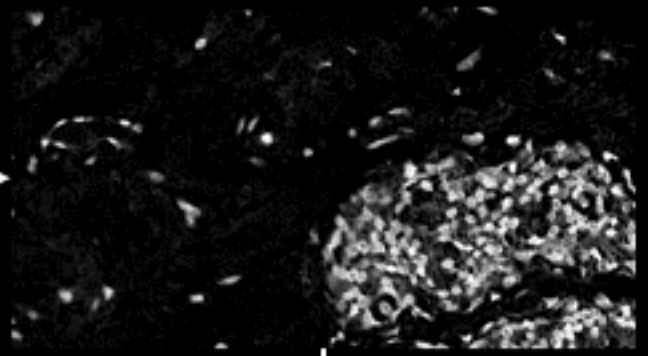
Gray levels



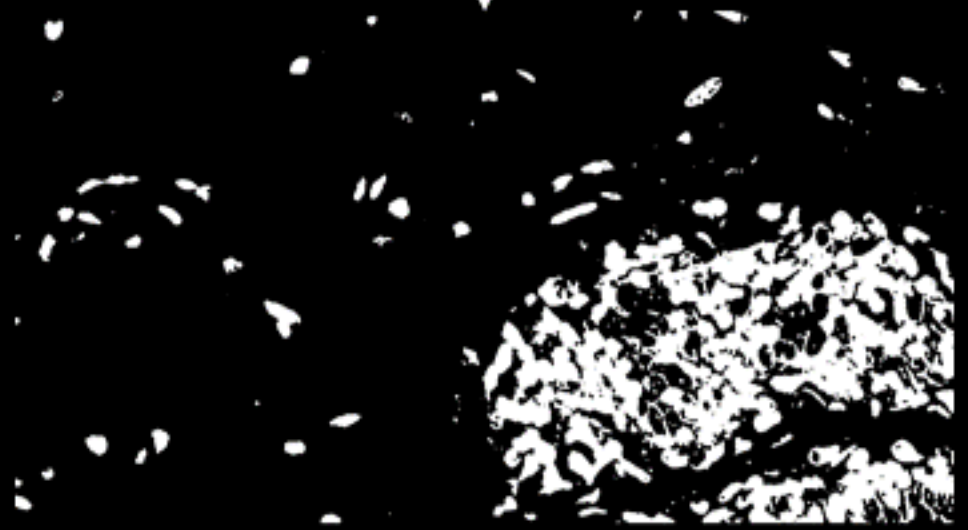
Minimum filtering



Gaussian blur

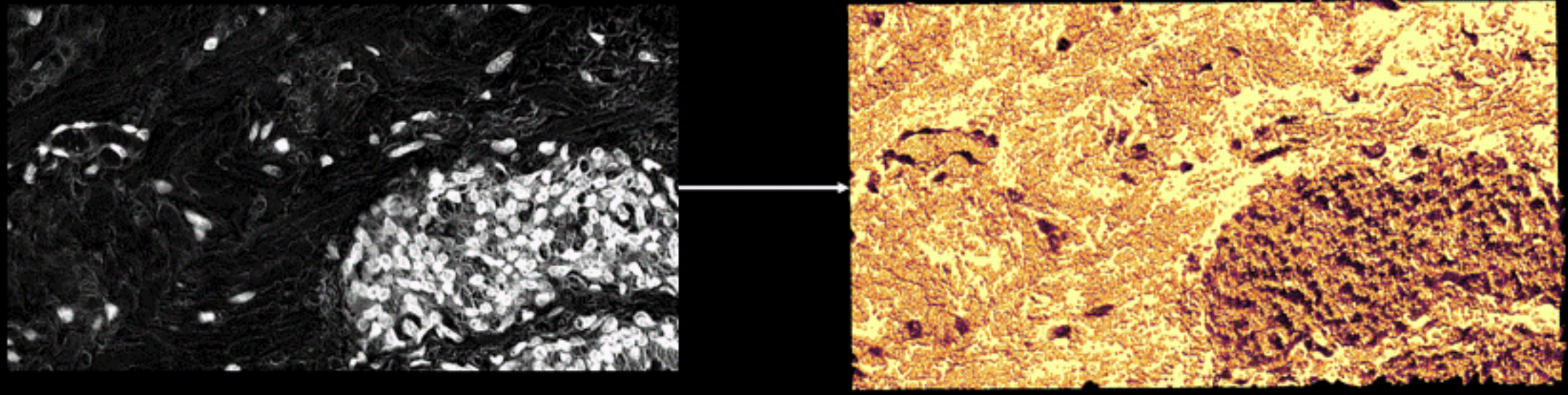


Thresholding



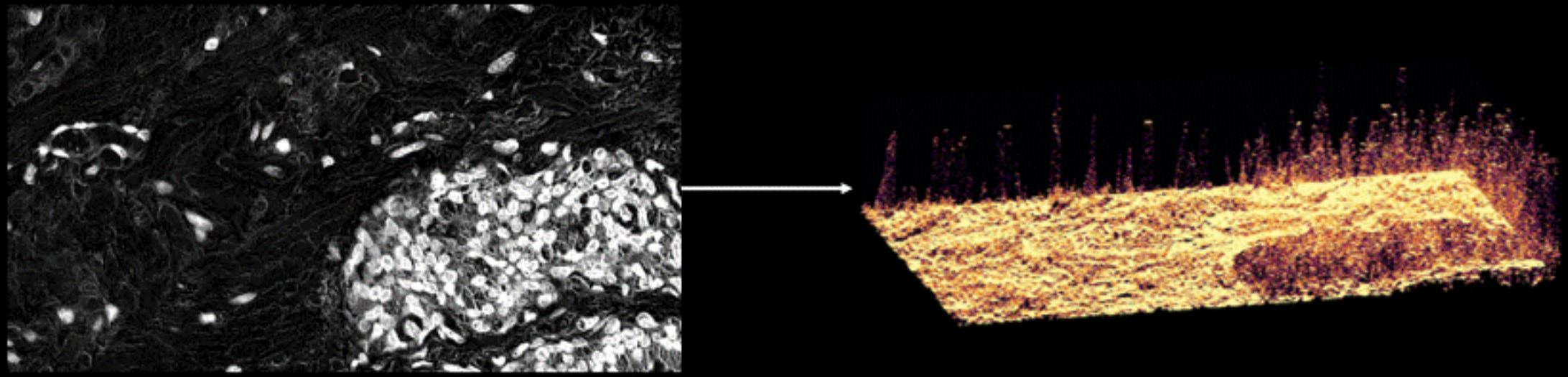
WATERSHED

Transform image so that intensity becomes 3rd dimension



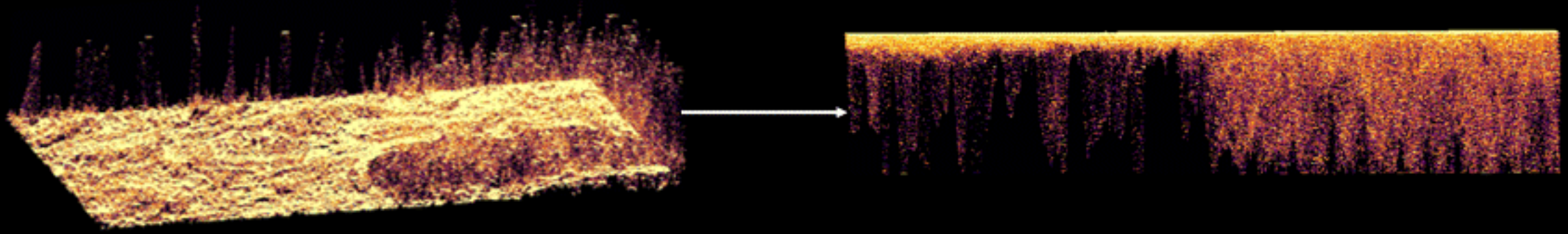
WATERSHED

Transform image so that intensity becomes 3rd dimension



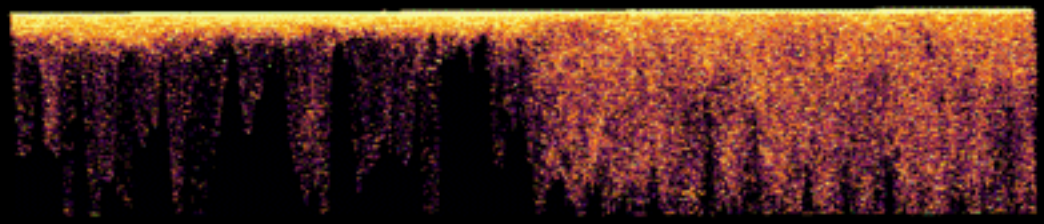
WATERSHED

Reverse intensity so regions with high intensity correspond to valleys



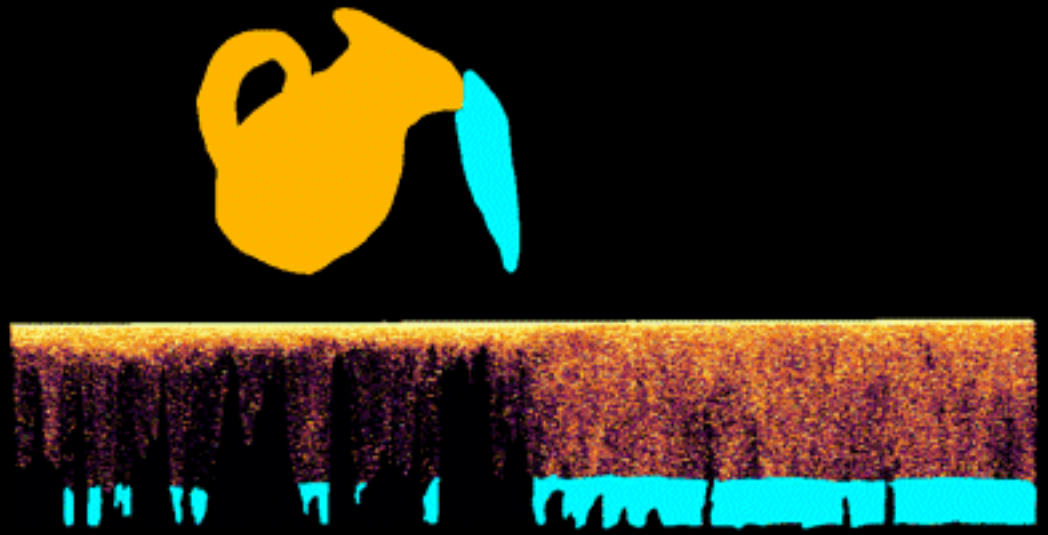
WATERSHED

Pour water into valleys



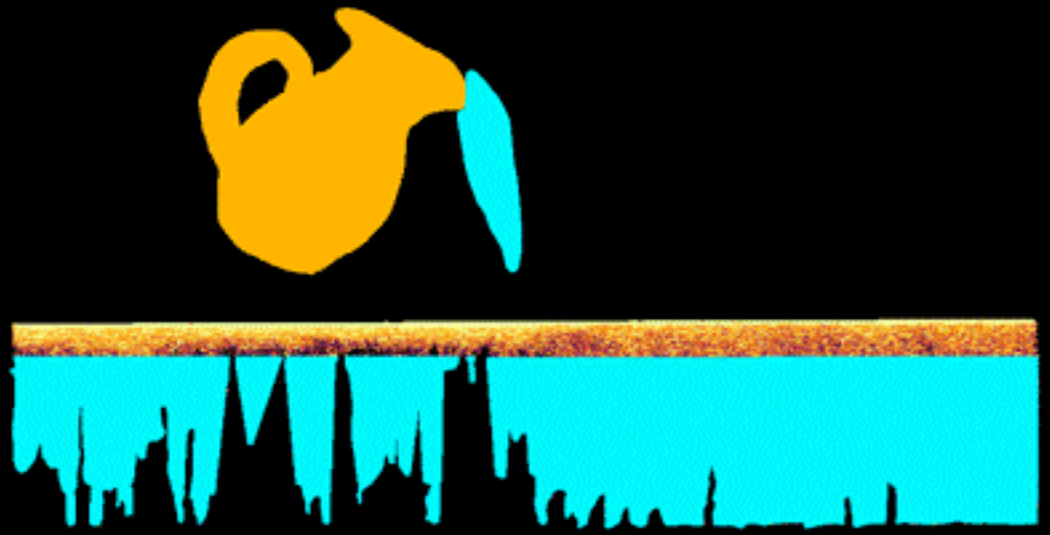
WATERSHED

Pour water into valleys

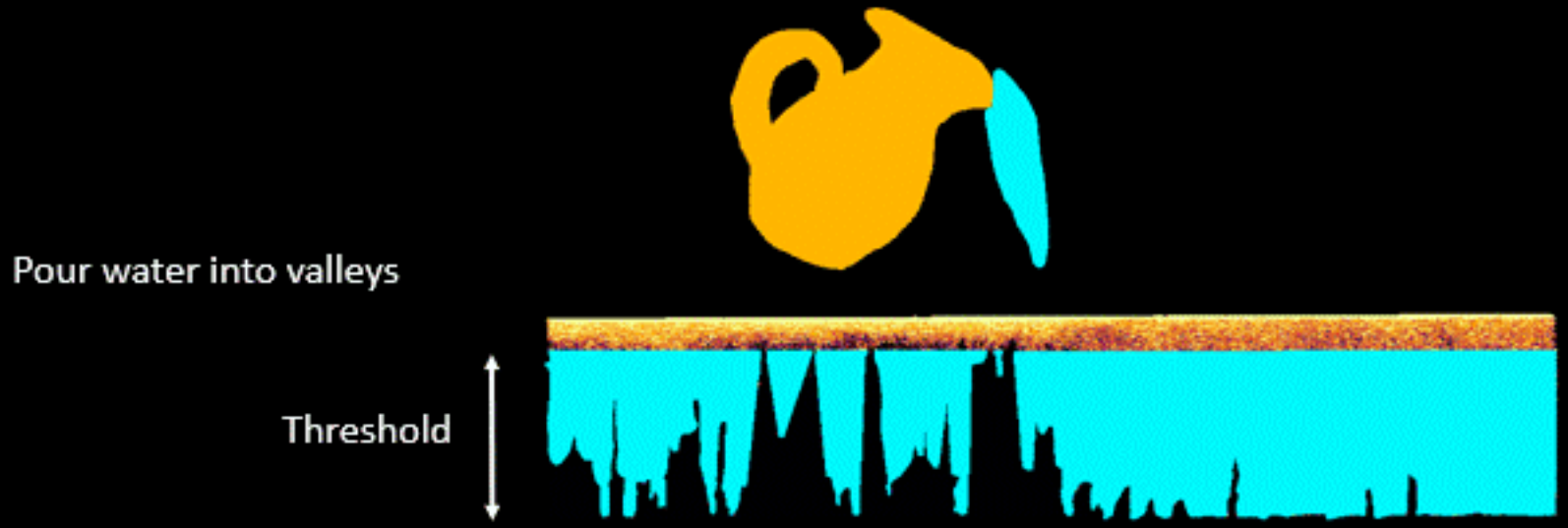


WATERSHED

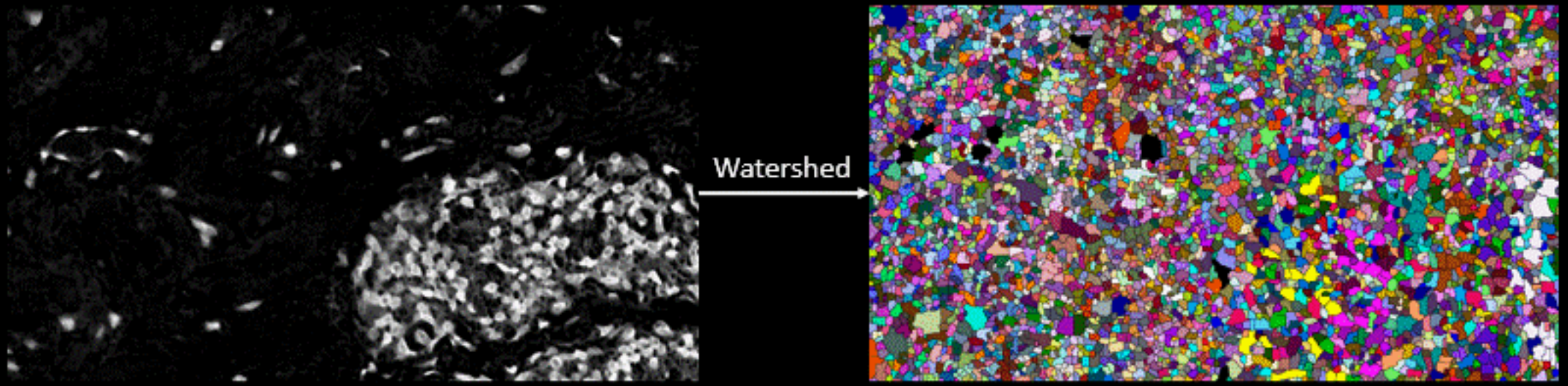
Pour water into valleys



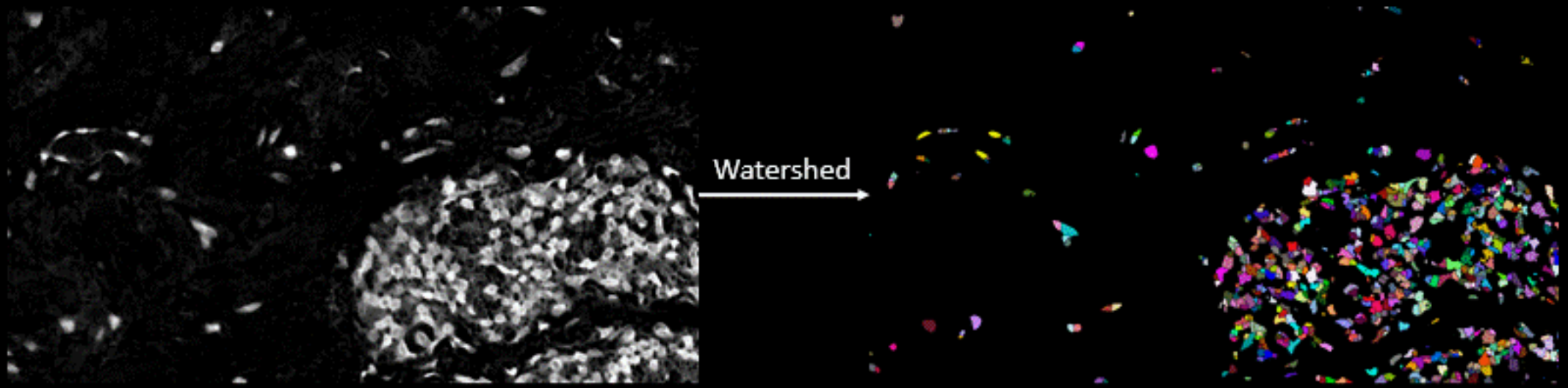
WATERSHED



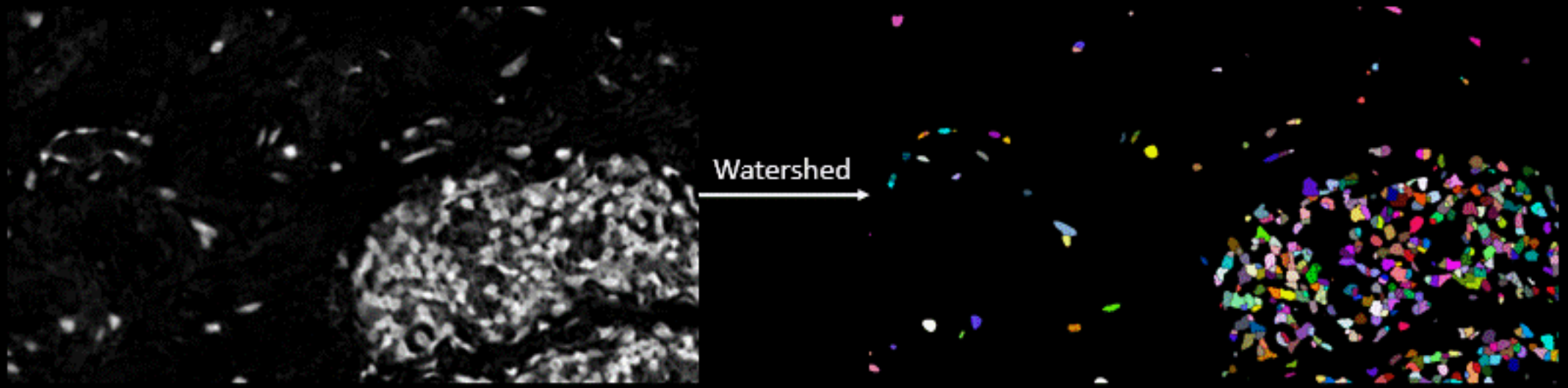
WATERSHED



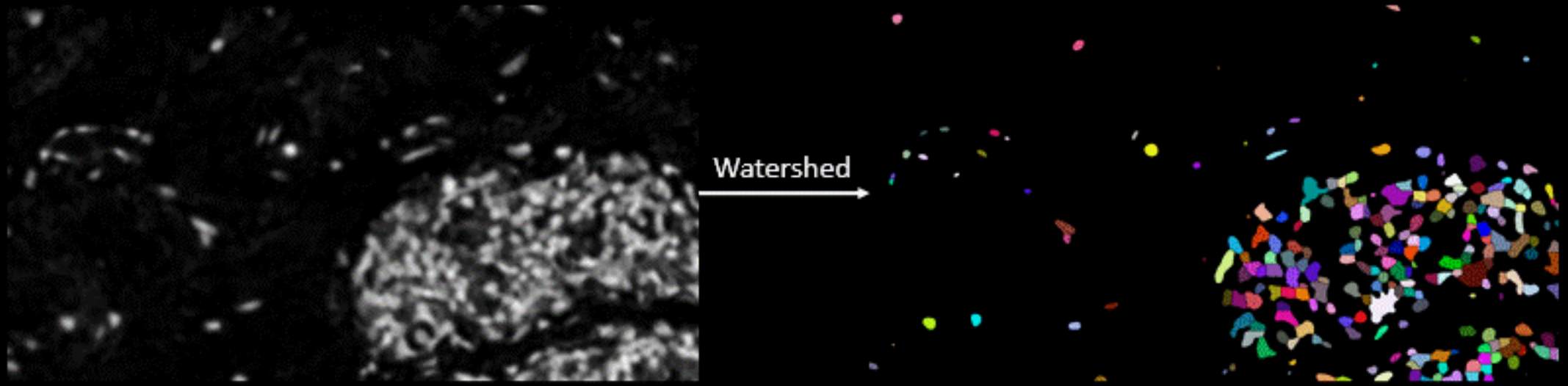
WATERSHED



WATERSHED



WATERSHED



Cell Detection Tool in QuPath

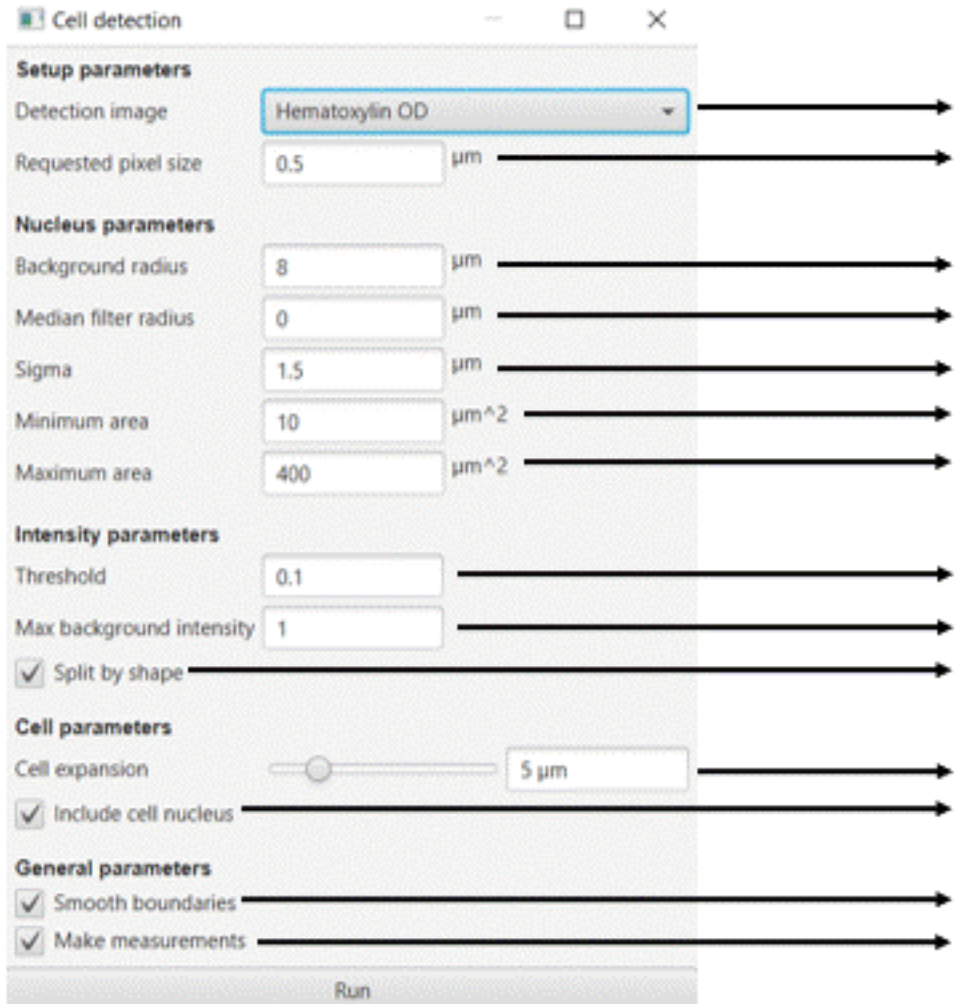


Image **component** used for cell detection

Image **resolution** used for cell detection

Radius used for **minimum filtering**

Radius for **median filtering**

Kernel size used for **Gaussian blur** before watershed

Nuclei with **area inferior** to this value are **filterd out**

Nuclei with **area superior** to this value are **filterd out**

Threshold used for **watershed**

Threshold used for **minimum filtering**

Separate nuclei based on shape (**binary watershed**)

Size used for cell expansion to define **cytoplasm area**

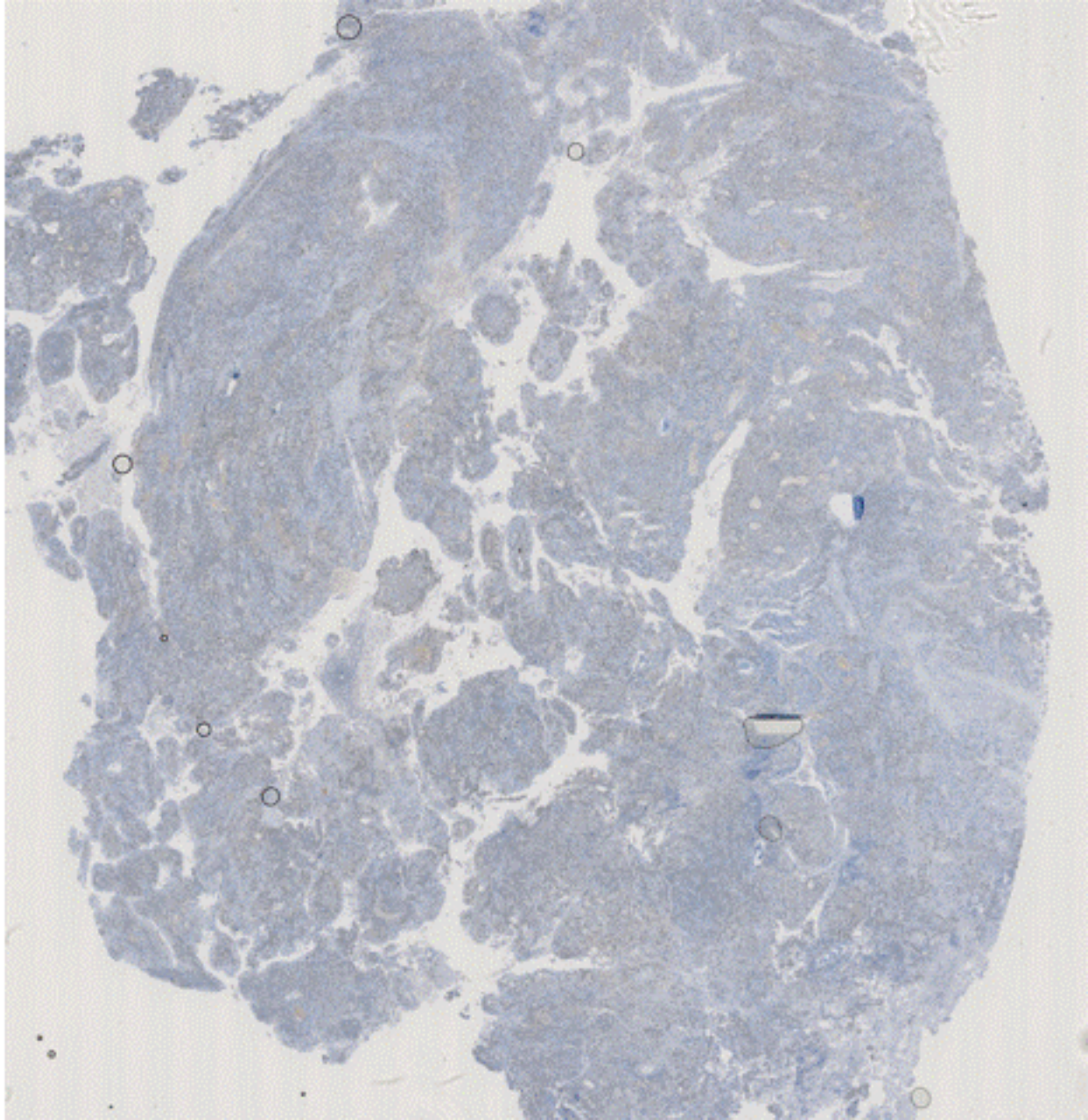
Define **nuclei and cytoplasm** areas or only cell areas

Define **smooth** cell **boundaries**

Get measurements associated with nuclei

Nuclei Segmentation

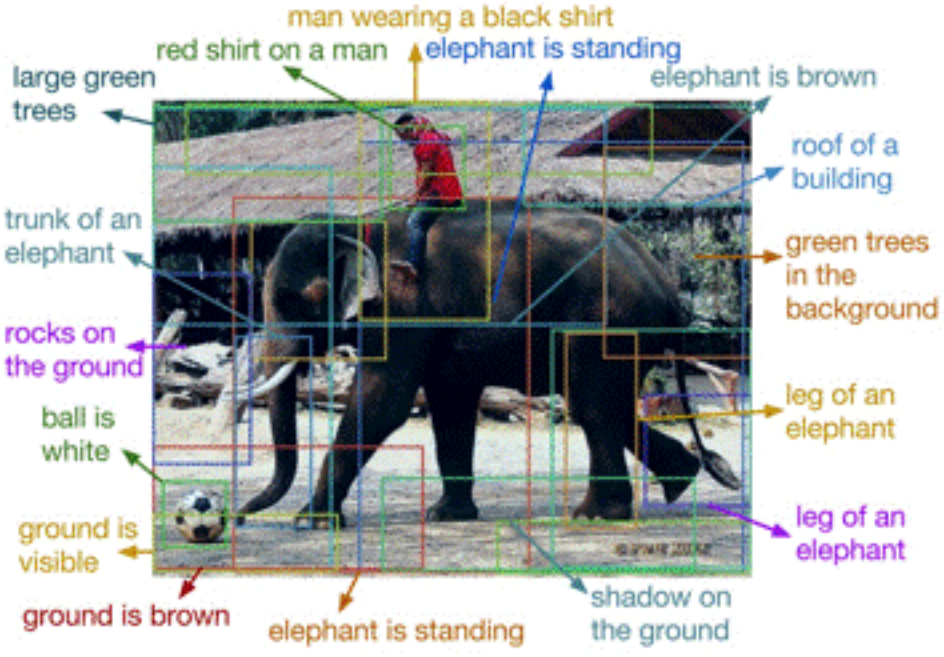
- **Open** KI67_lung.ndpi
- Open **Positive cell detection**
- Create an **annotation**
- In the annotation, **detect positive** and **negative cells** by defining **one threshold**
- Identify tissue with **Create thresholder**
- **Apply** "Positive cell detection" on **the tissue**
- Get **number of nuclei** and **proportion** of **positive cells**



Deep Learning

Speech Recognition (2011)

Computer vision (2012)

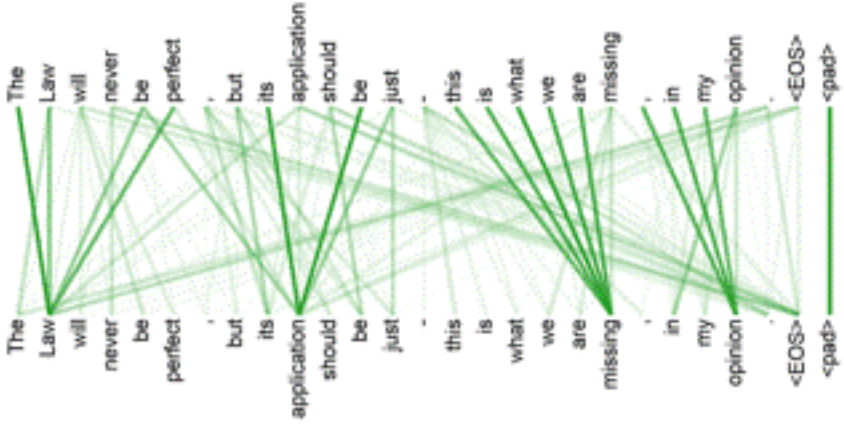


DenseCap, Johnson *et al.*, 2016

Go game (2016)

Mastering the game of Go with deep neural networks and tree search, Silver *et al.*, Nature, 2016

Natural Language Processing for translation and chatbots (transformers 2017)



Attention is all you need, Vaswani *et al.*, 2017

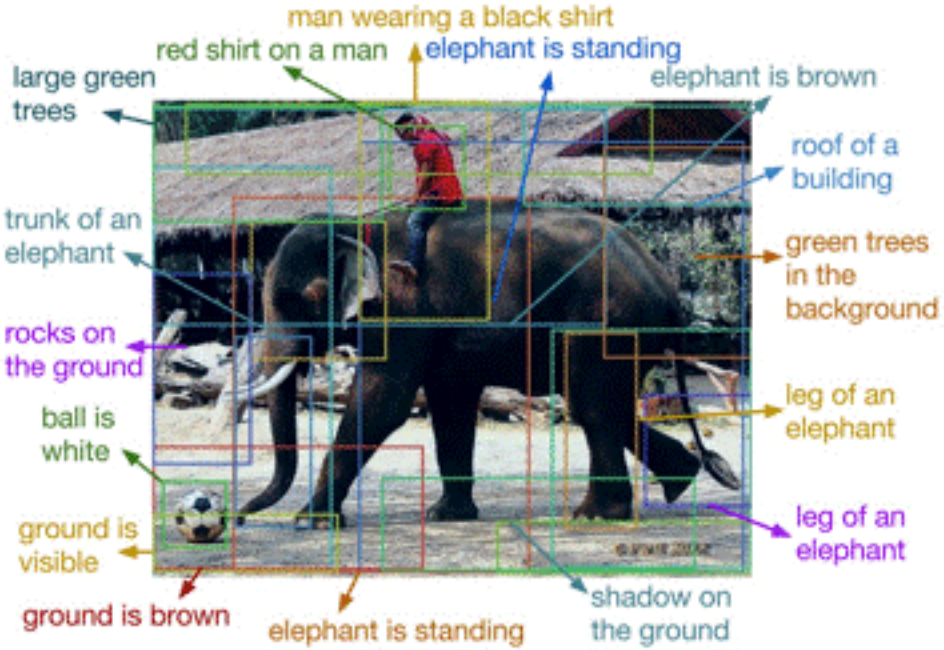
Tech industry

Art, music, journalism, ...

DEEP LEARNING

Speech Recognition (2011)

Computer vision (2012)

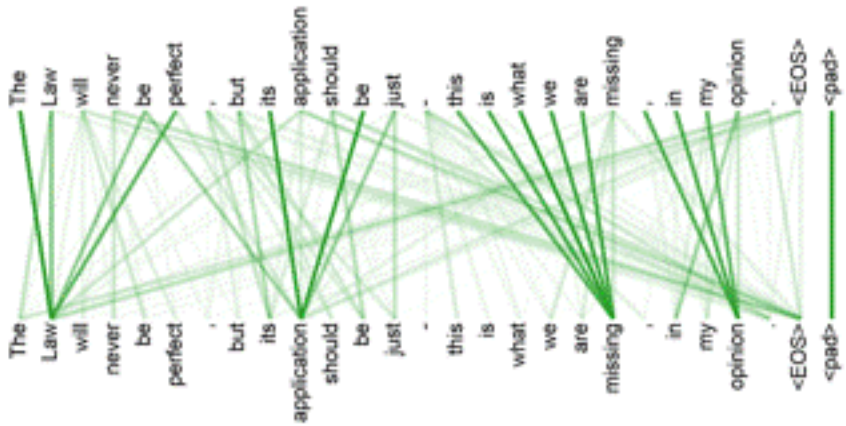


DenseCap, Johnson *et al.*, 2016

Go game (2016)

Mastering the game of Go with deep neural networks and tree search, Silver *et al.*, Nature, 2016

Natural Language Processing for translation and chatbots (transformers 2017)



Attention is all you need, Vaswani *et al.*, 2017

Tech industry

Art, music, journalism, ...

What is AI, more particularly deep learning?

DEEP LEARNING REQUIRES:

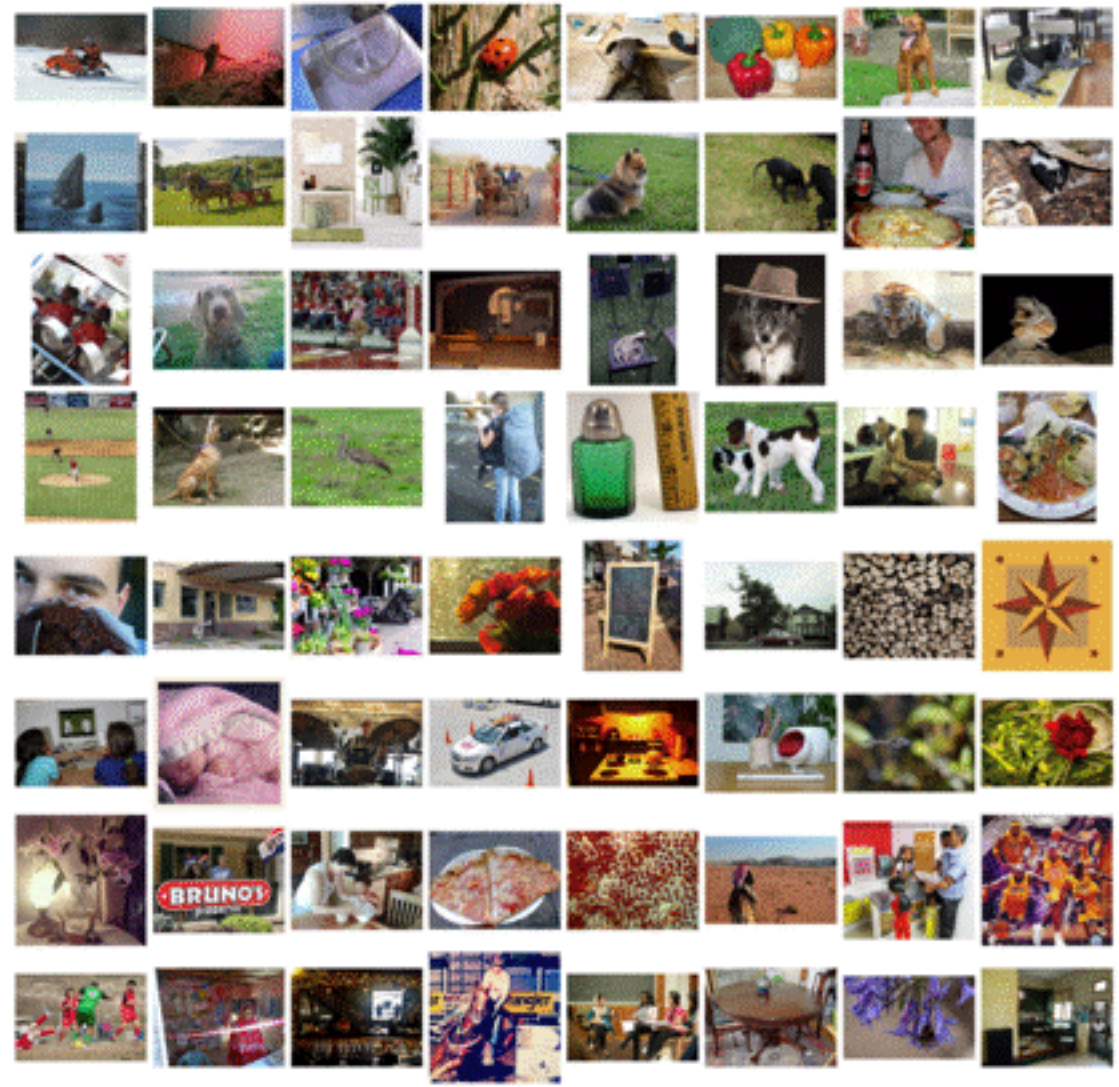
1) High computational power

4.4 Speedup factor of GPU code

The GPU code scales well with network size. For small nets the speedup is small (but still over 10) since they fit better inside the CPU cache, and GPU resources are underutilized. For huge nets (ex: Table 2) the GPU implementation is more than 60 times faster than a compiler-optimized CPU version. Given the flexibility of our GPU version, this is a significant speedup. One epoch takes 35 GPU minutes but more than 35 CPU hours.

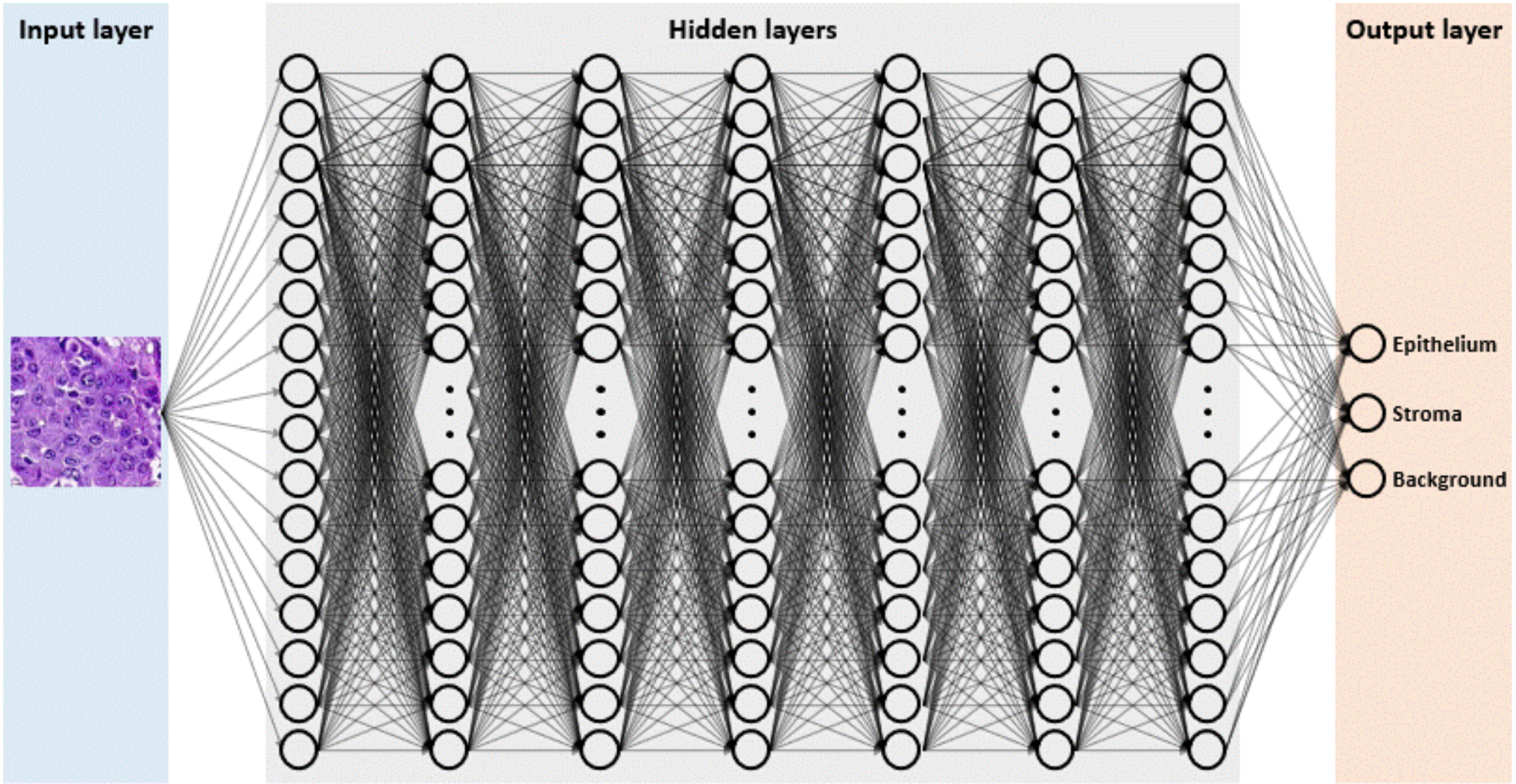
Cireşan *et al.*, High-Performance Neural Networks for Visual Object Classification. Tech Report, 2011.

2) Large amounts of training data

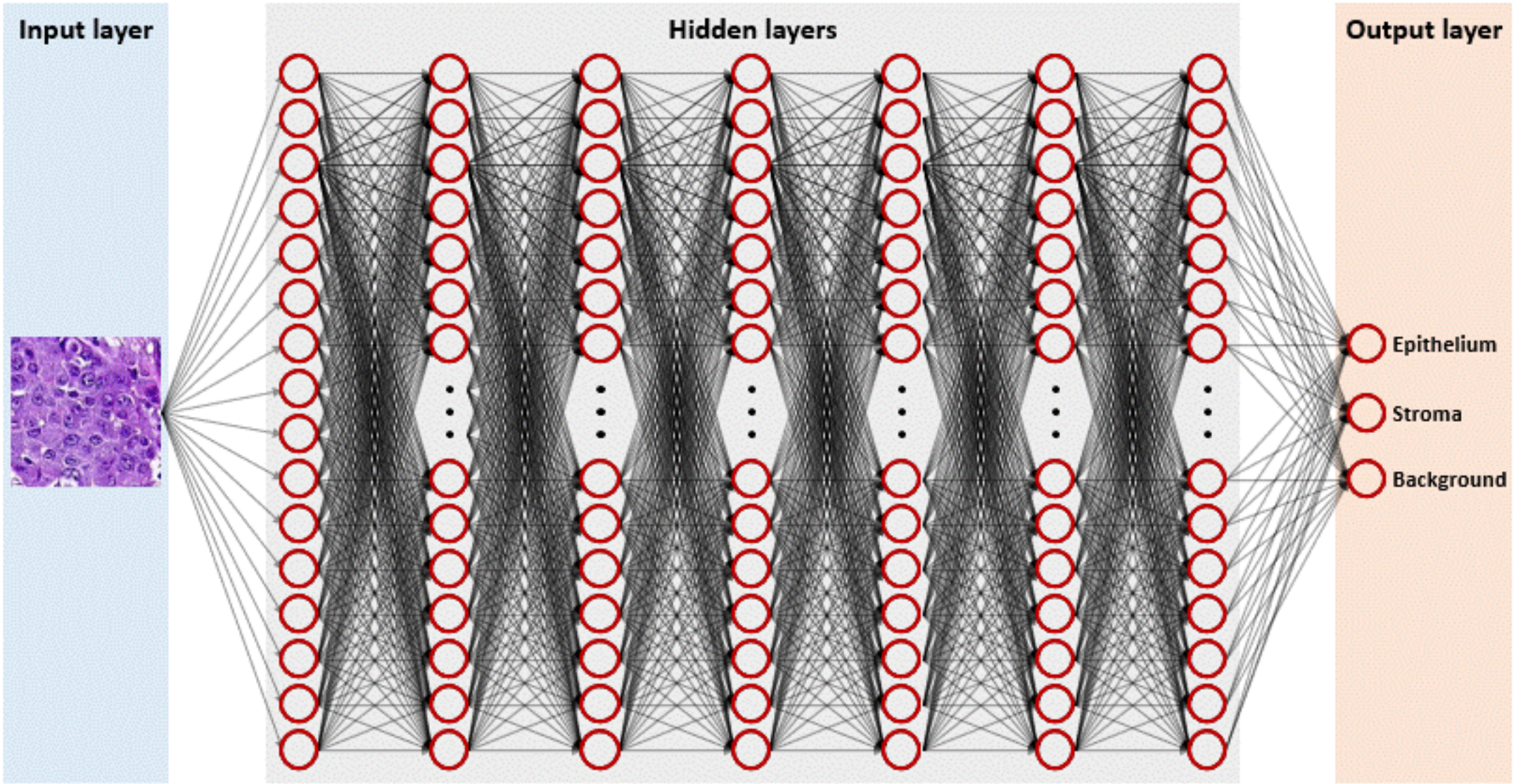


ImageNet Large Scale Visual Recognition Challenge (2010 – 2017)
1000 classes – 10,000,000 images

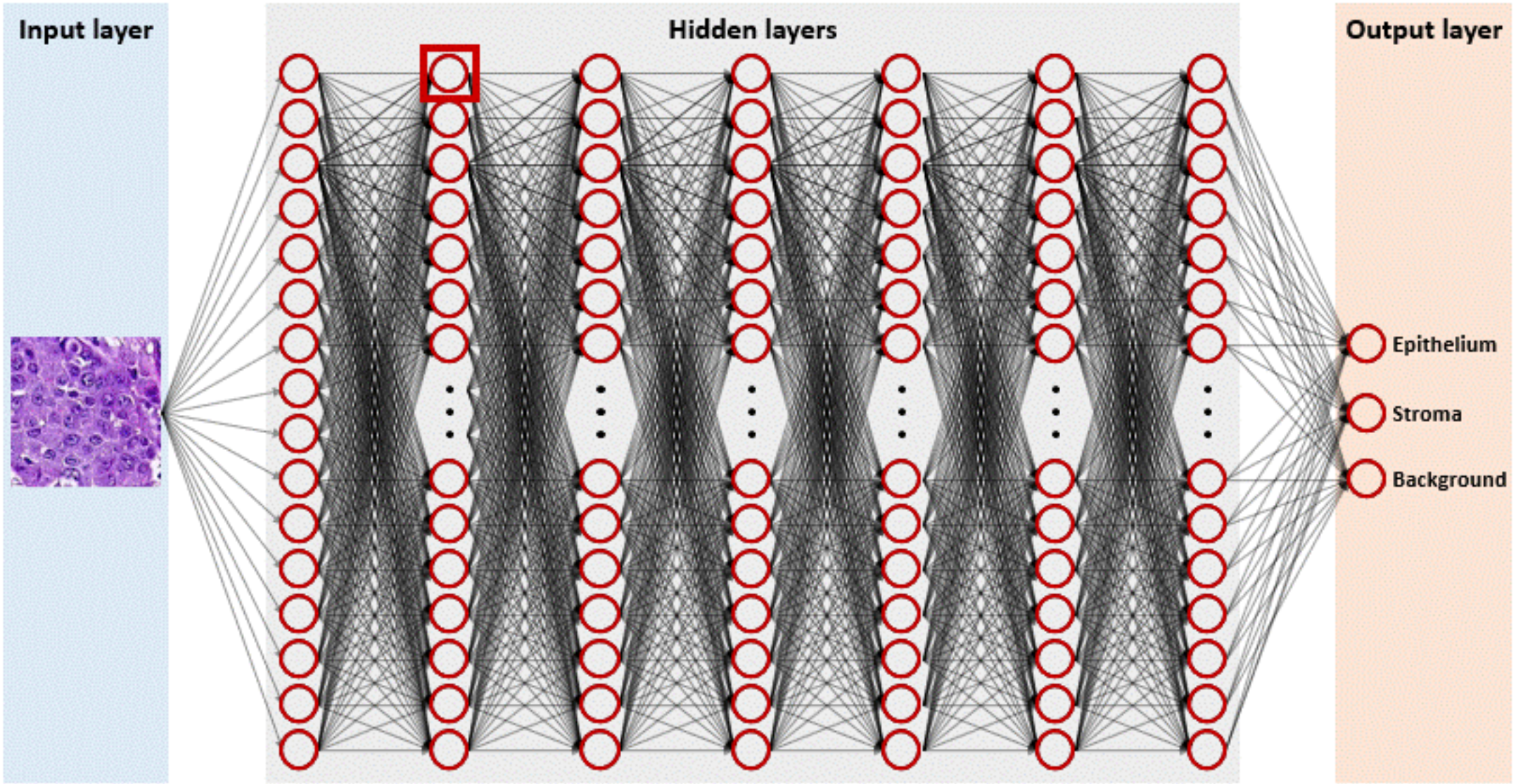
Deep Convolutional Neural Networks



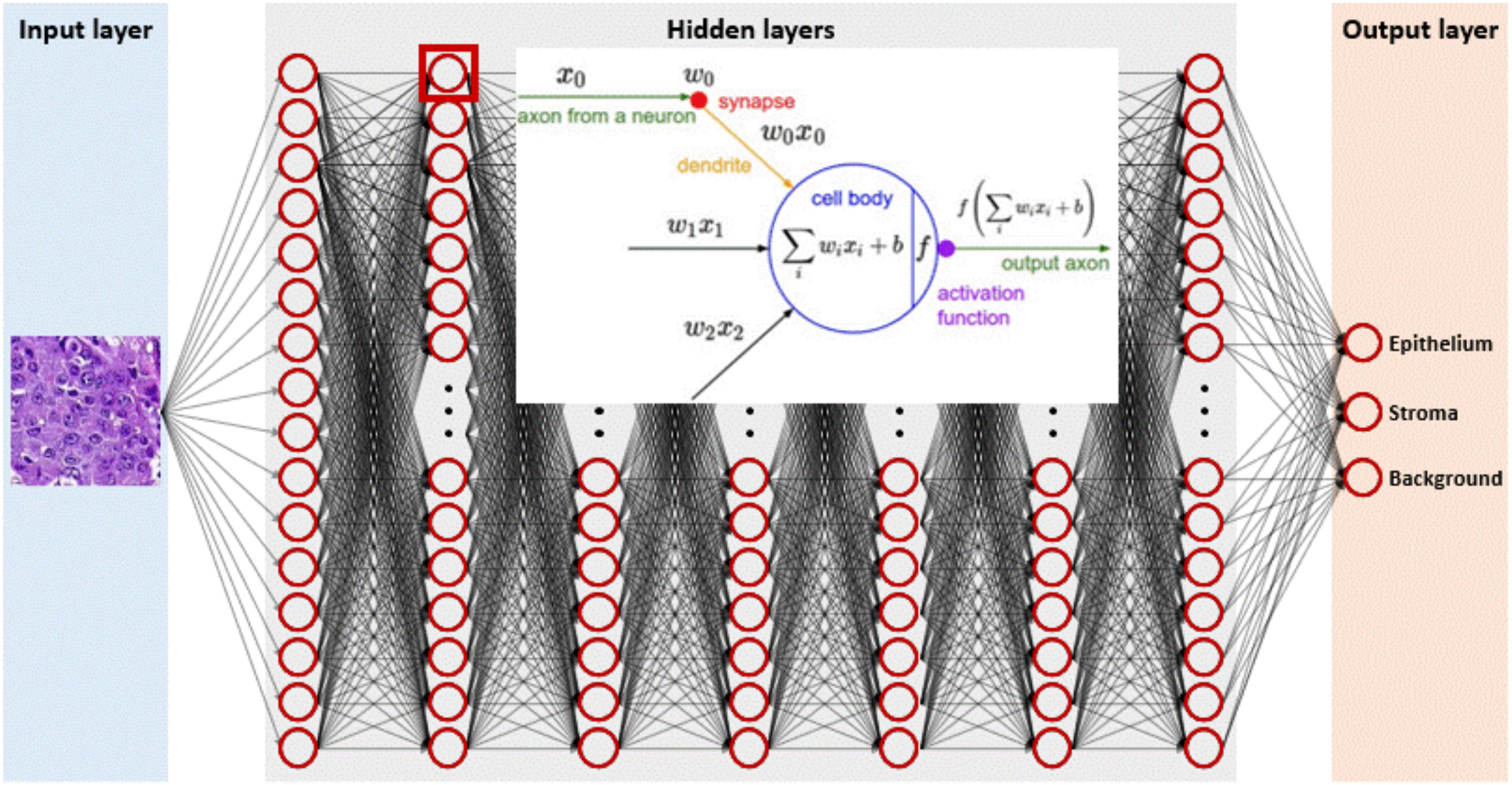
Deep Convolutional Neural Networks



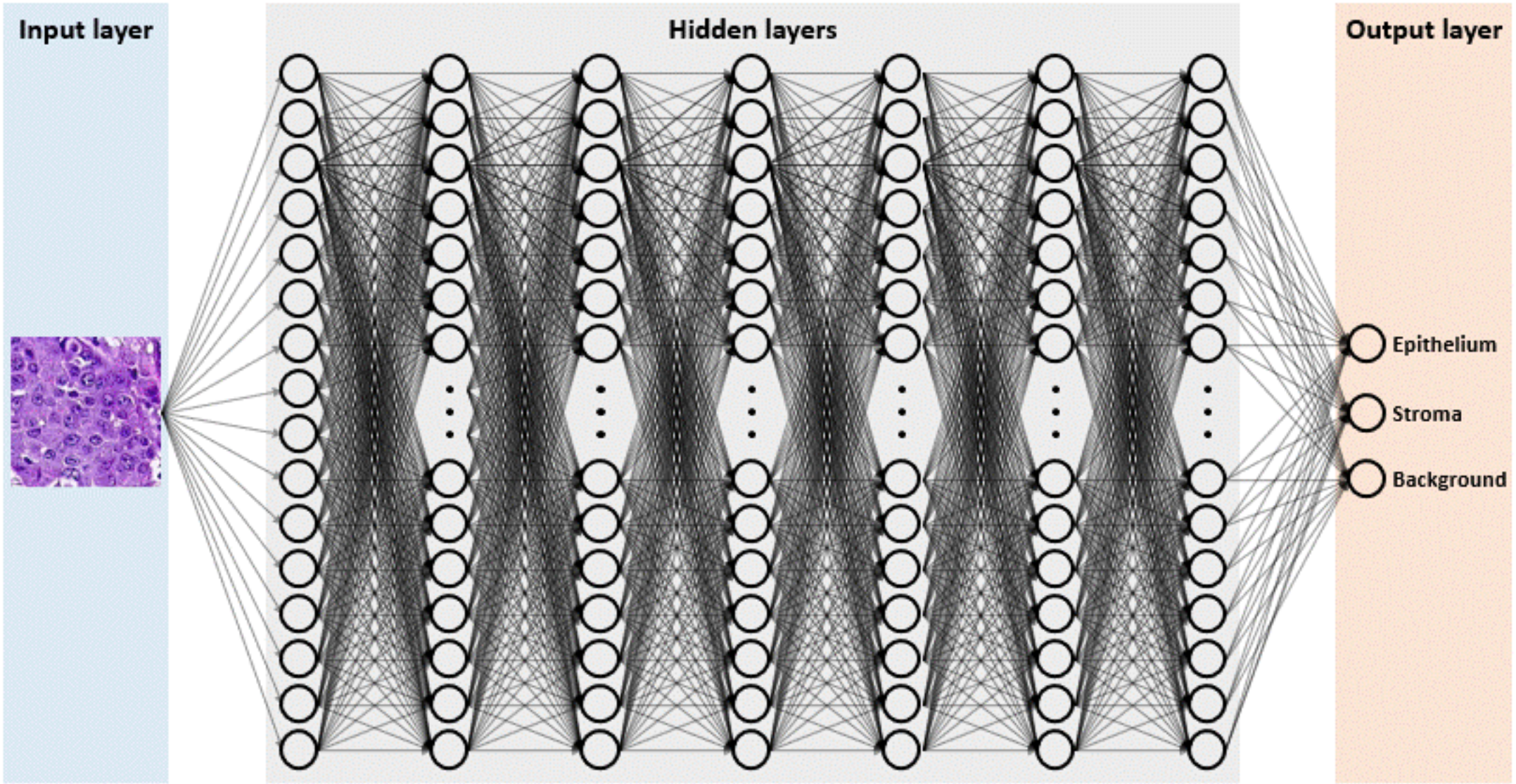
Deep Convolutional Neural Networks



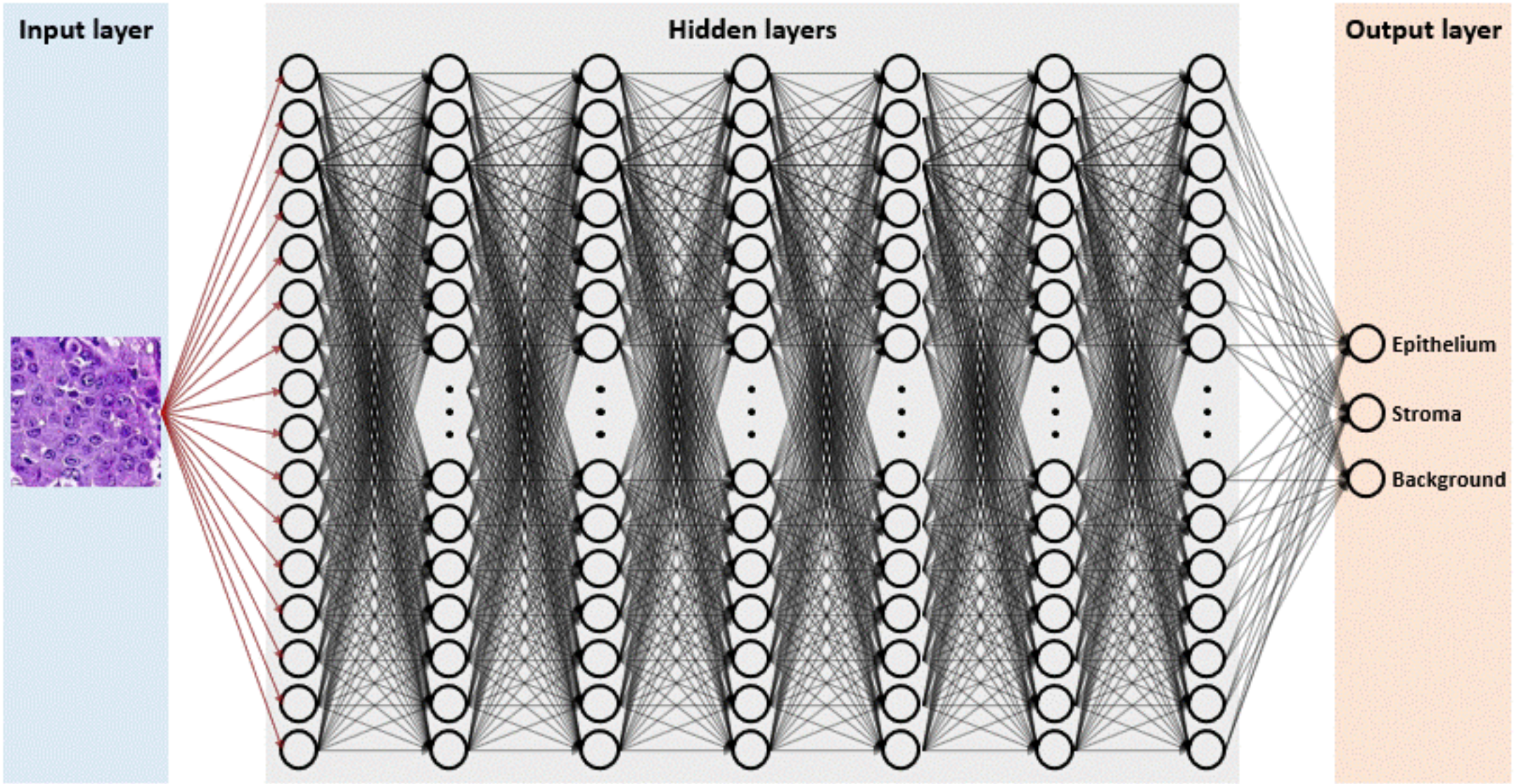
Deep Convolutional Neural Networks



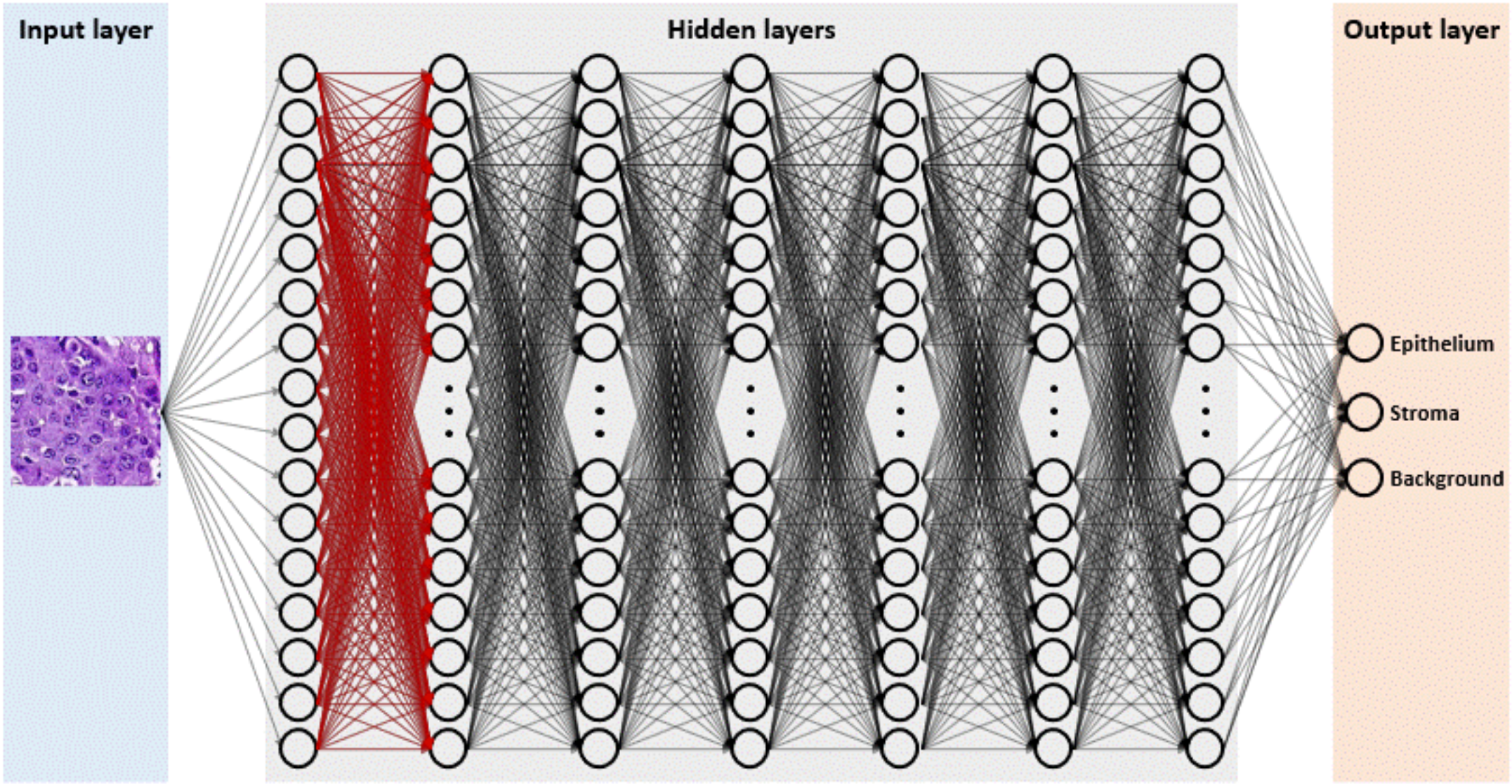
Deep Convolutional Neural Networks



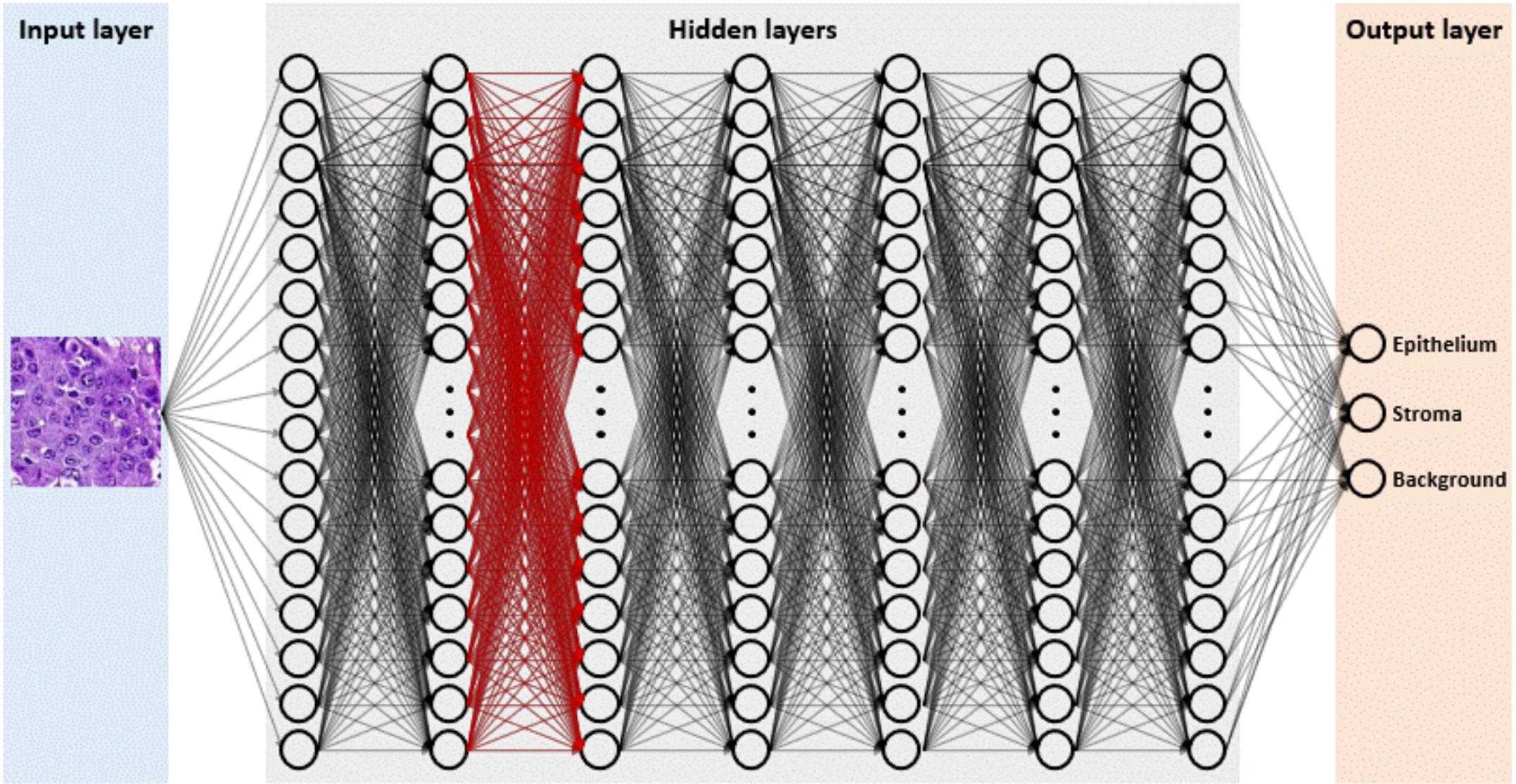
Deep Convolutional Neural Networks



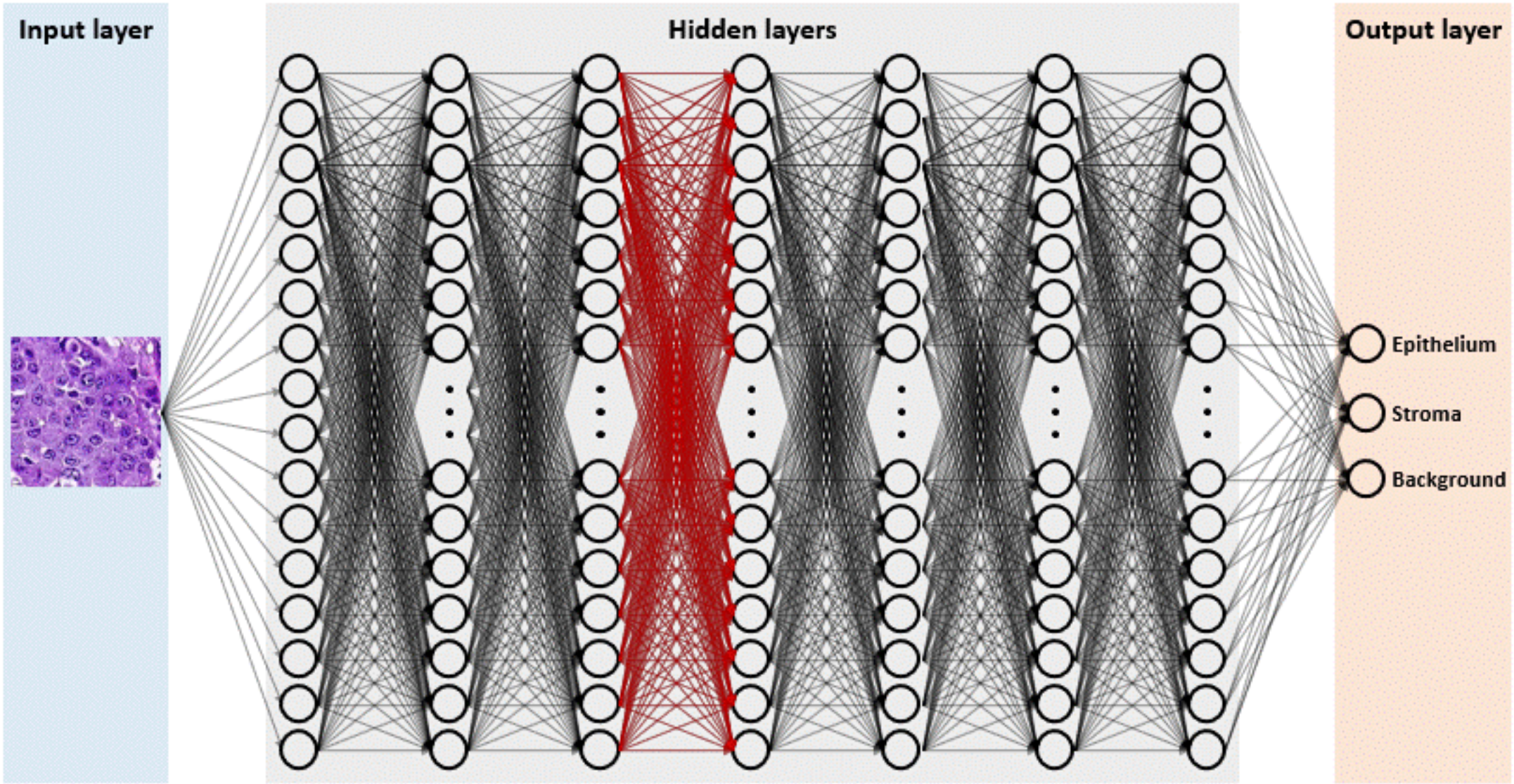
Deep Convolutional Neural Networks



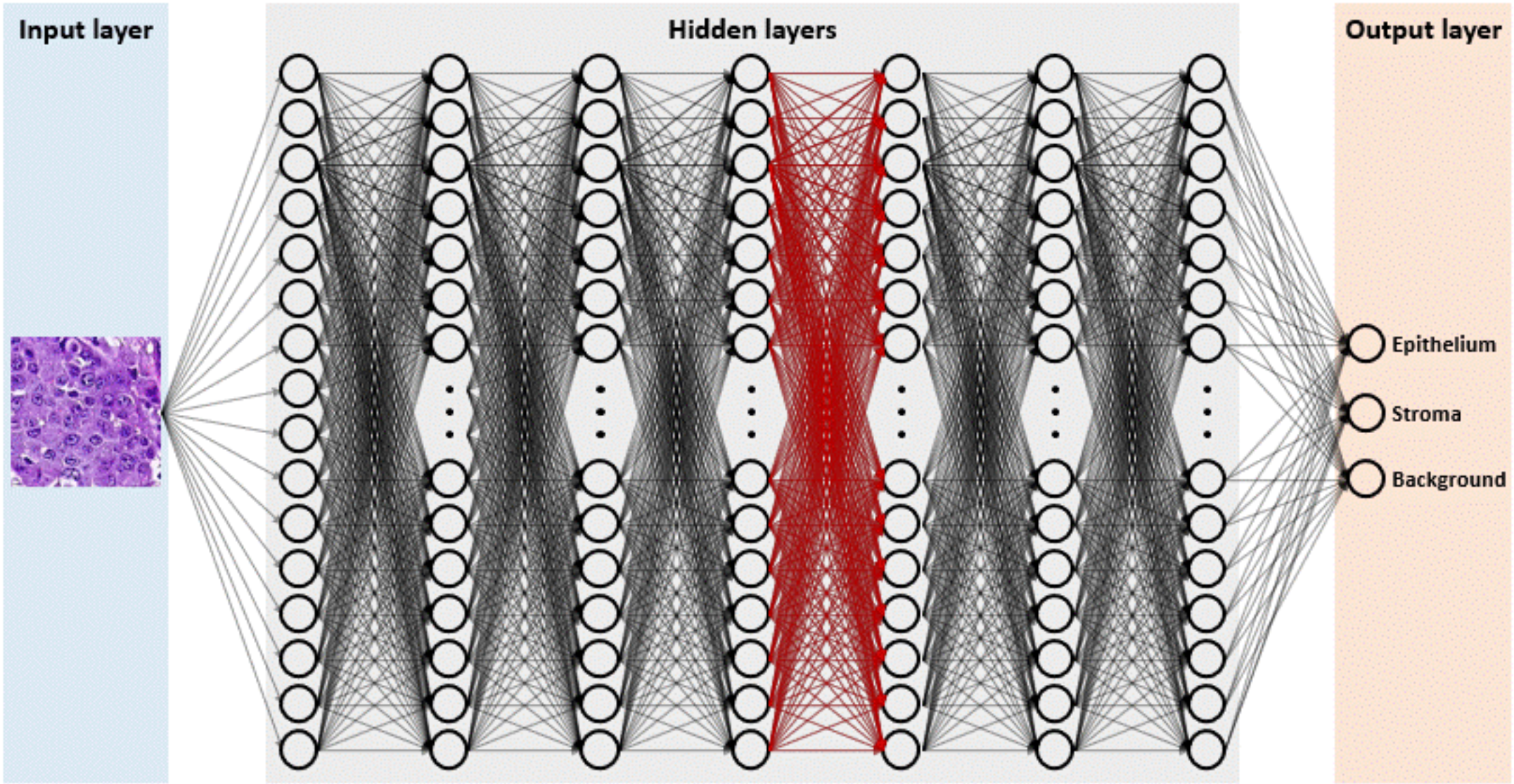
Deep Convolutional Neural Networks



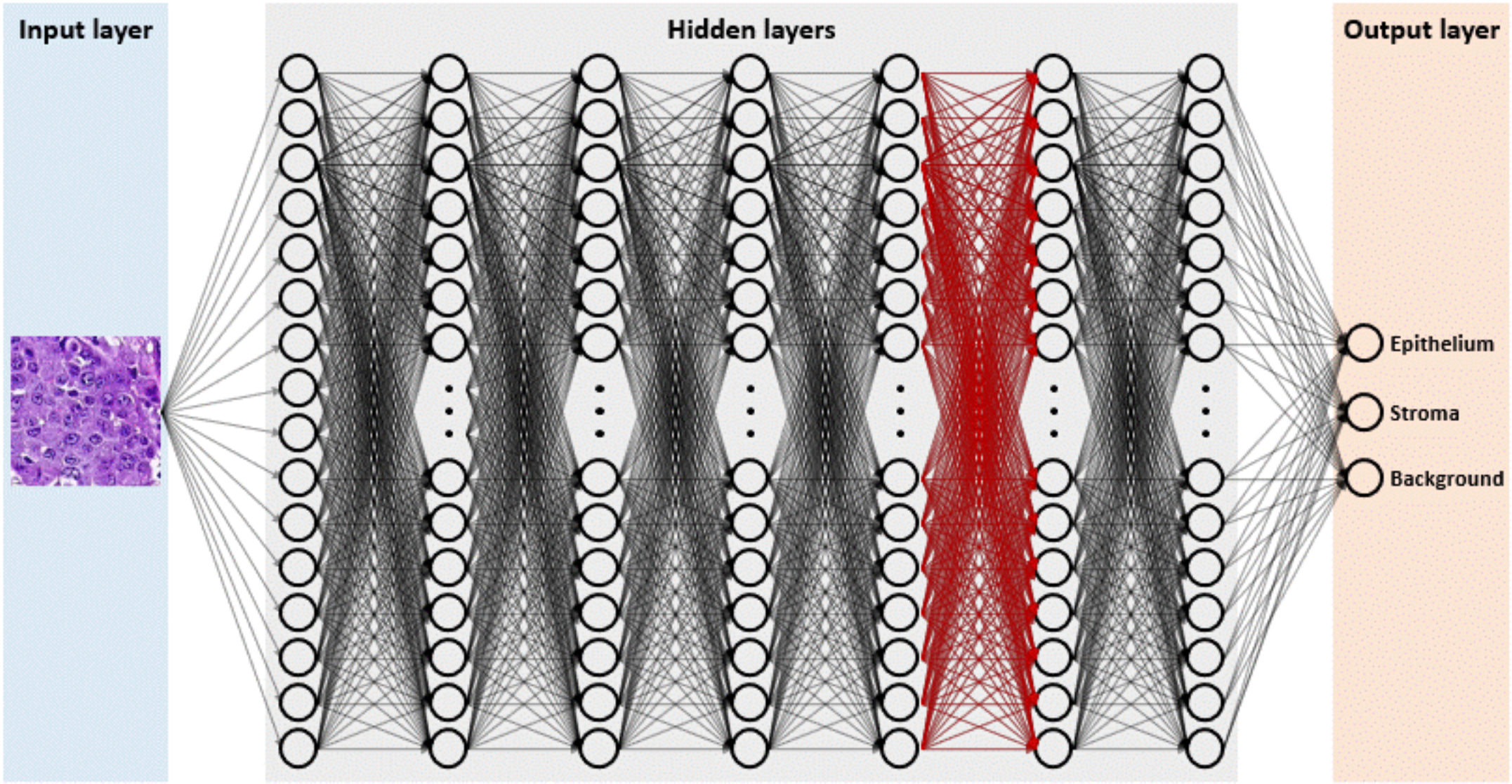
Deep Convolutional Neural Networks



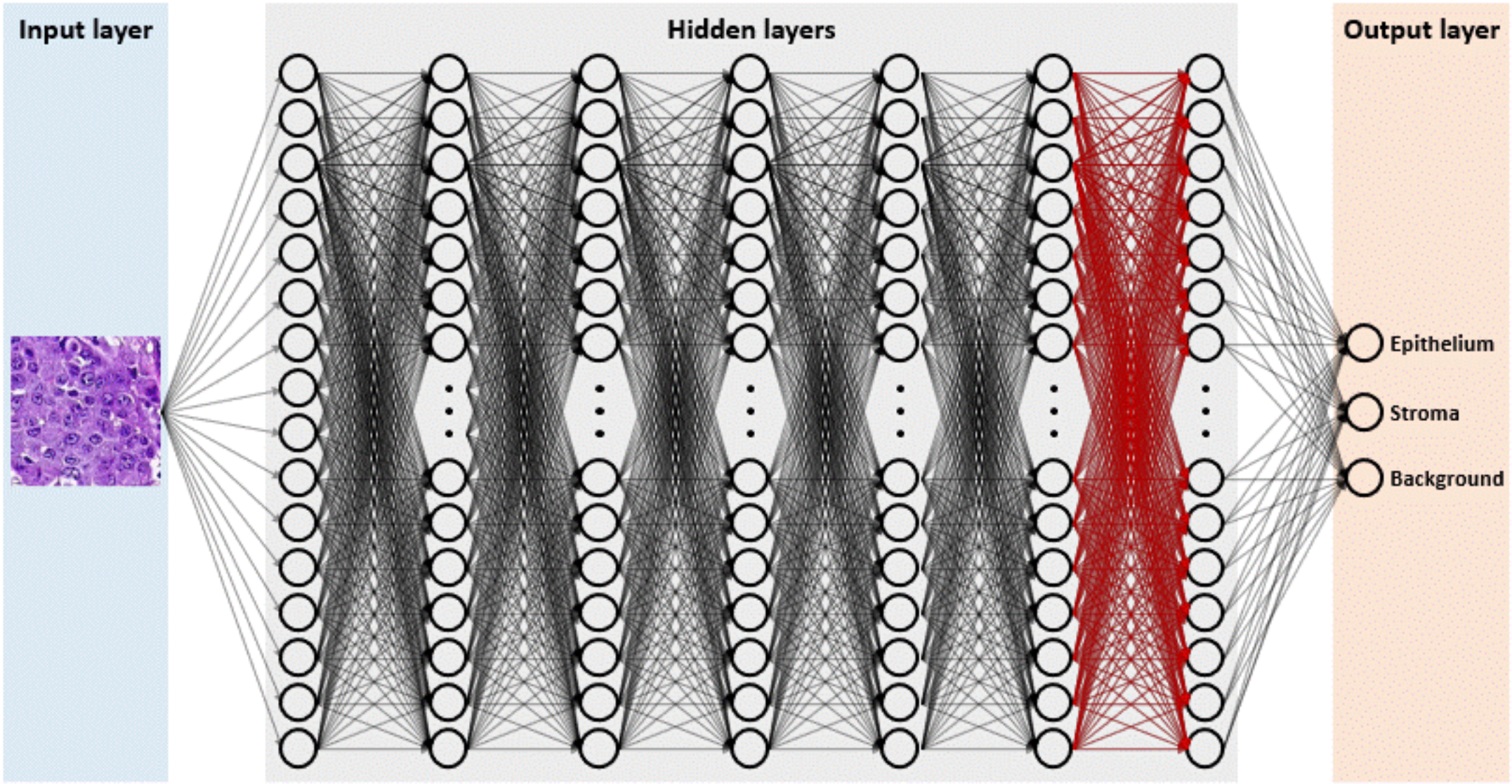
Deep Convolutional Neural Networks



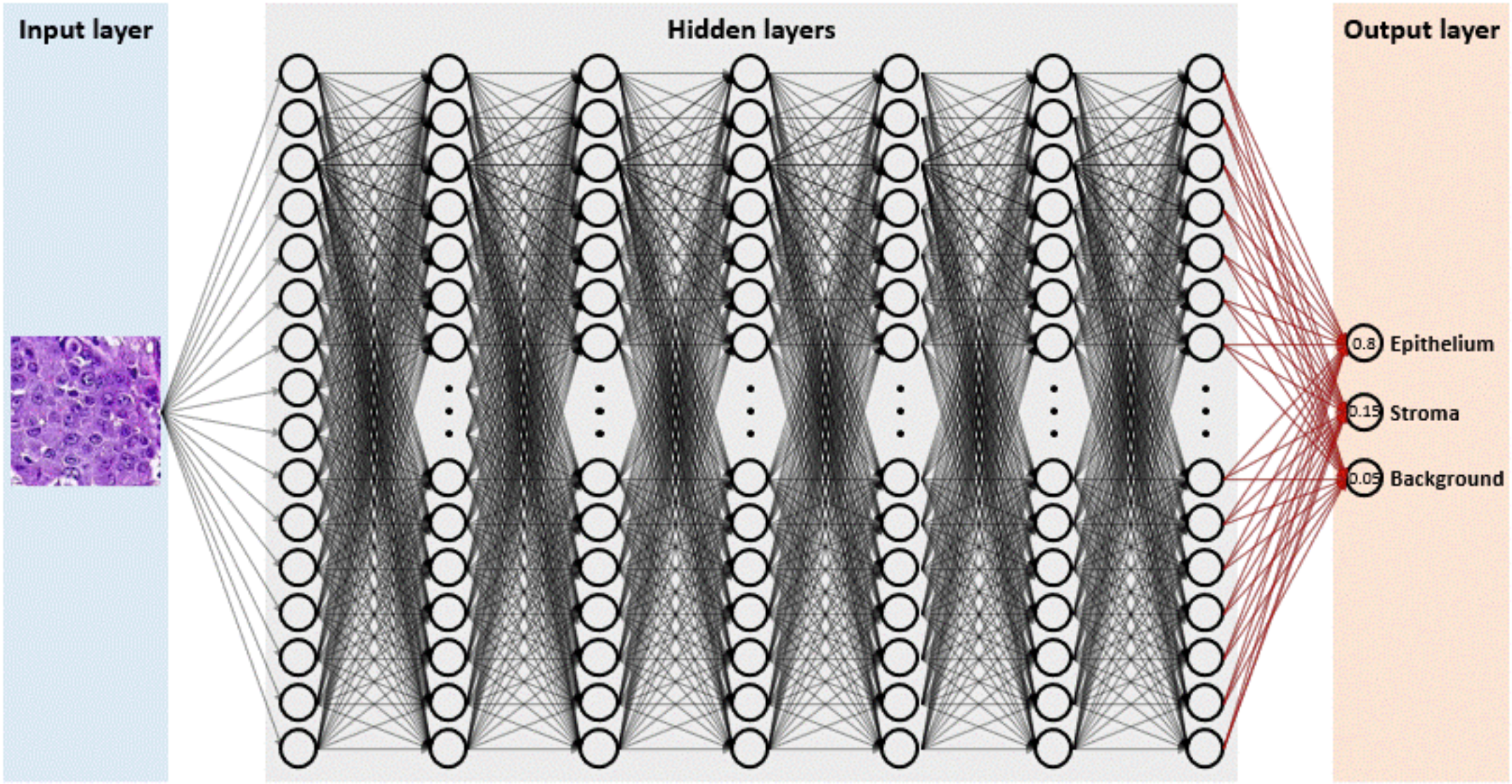
Deep Convolutional Neural Networks



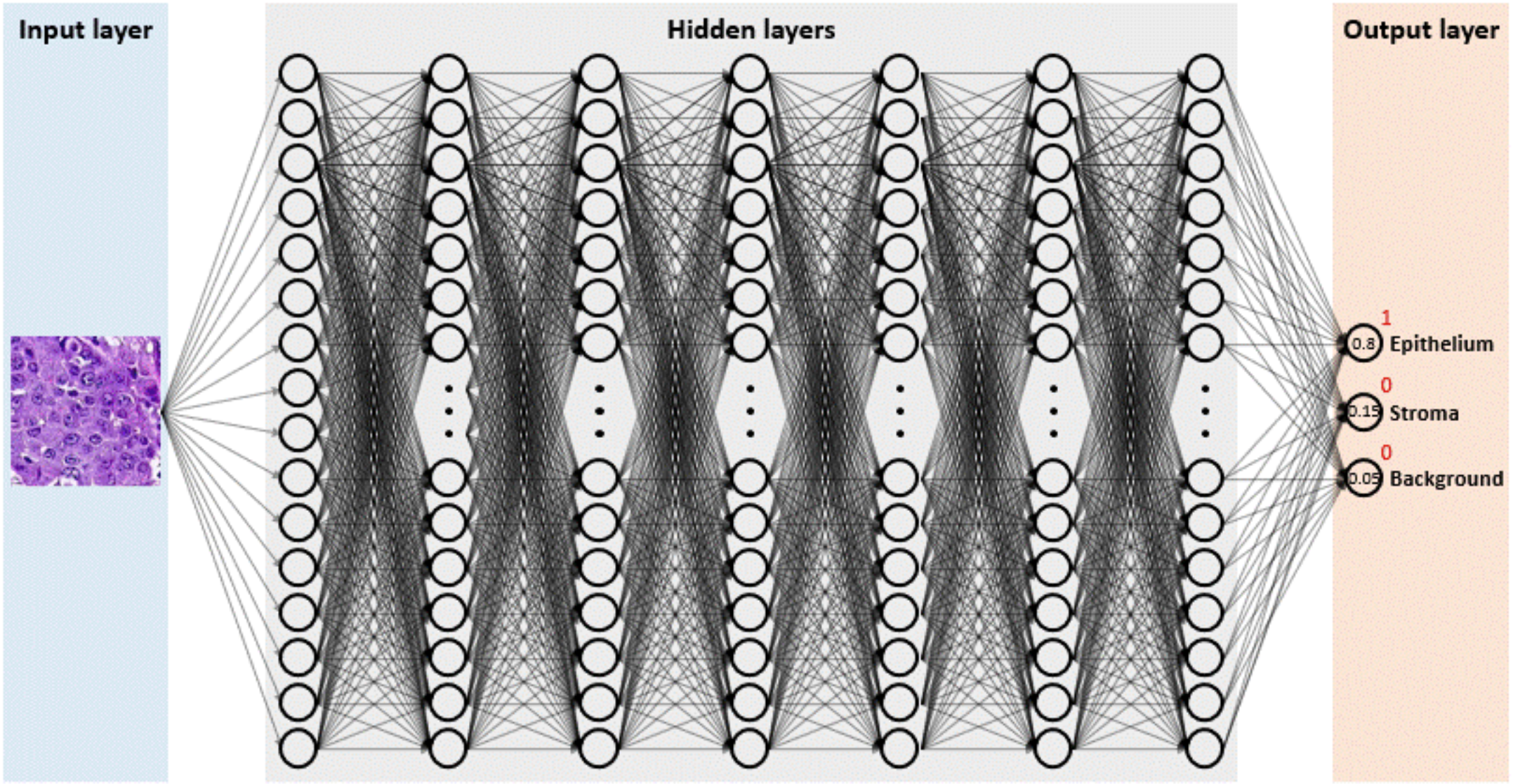
Deep Convolutional Neural Networks



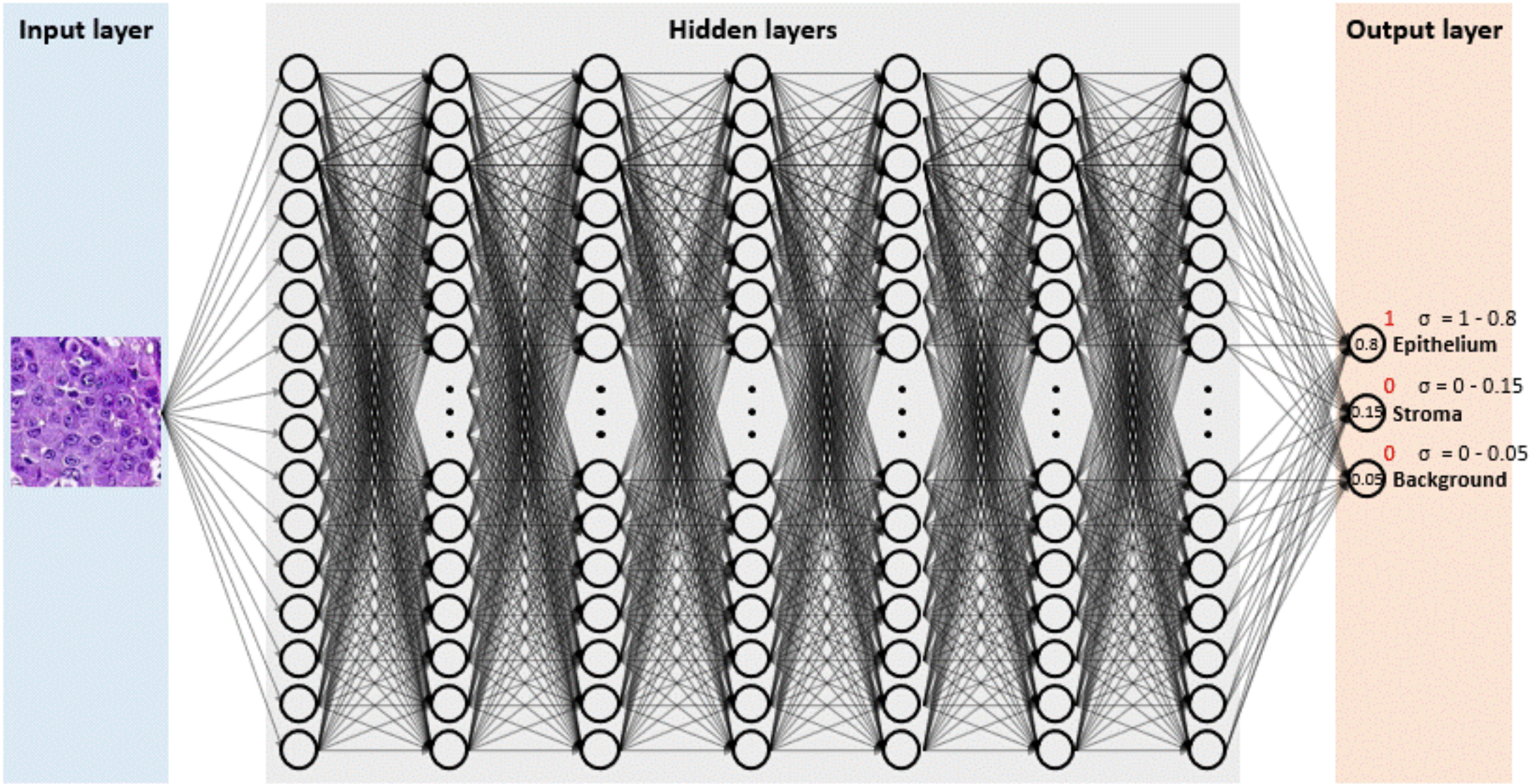
Deep Convolutional Neural Networks



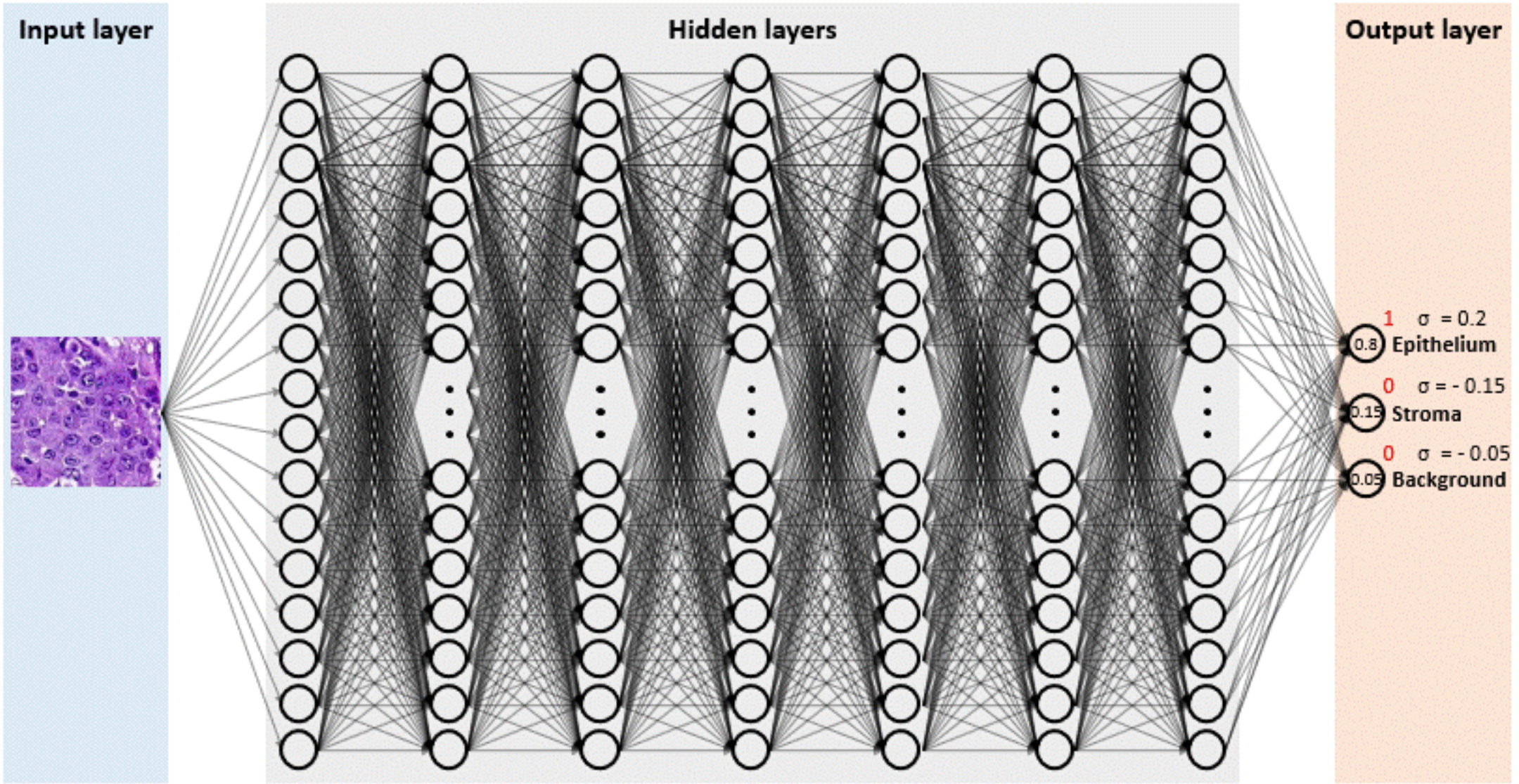
Deep Convolutional Neural Networks



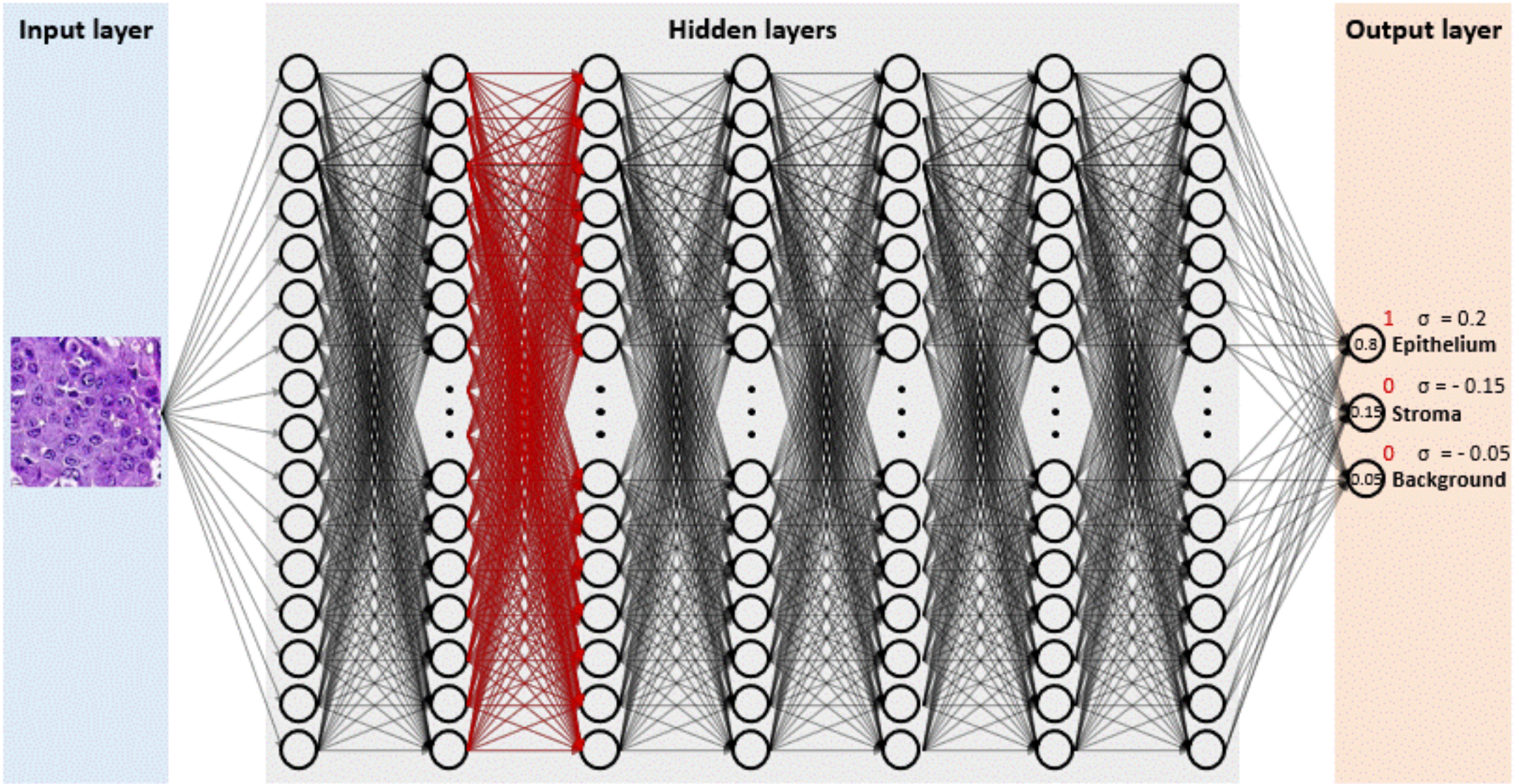
Deep Convolutional Neural Networks



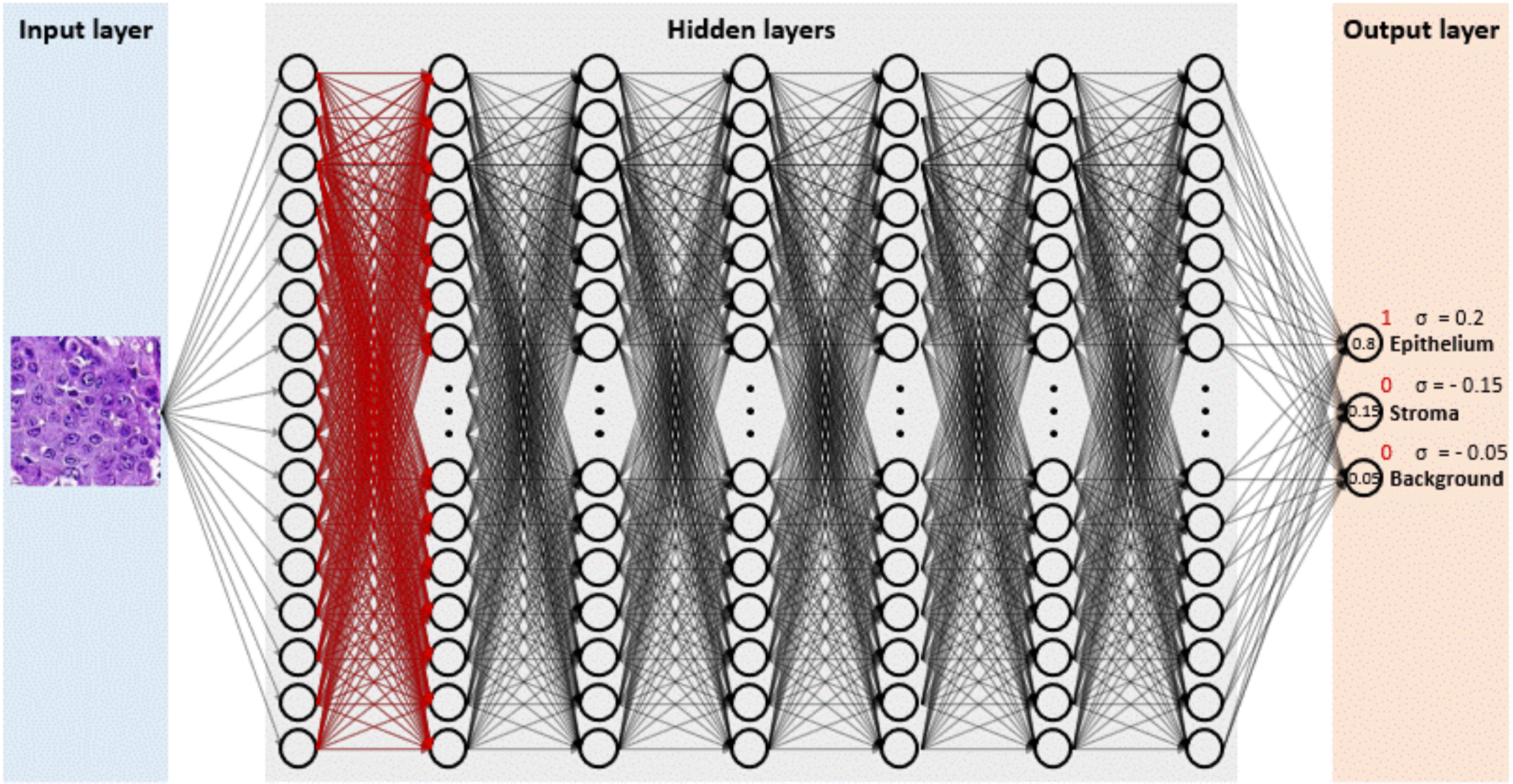
Deep Convolutional Neural Networks



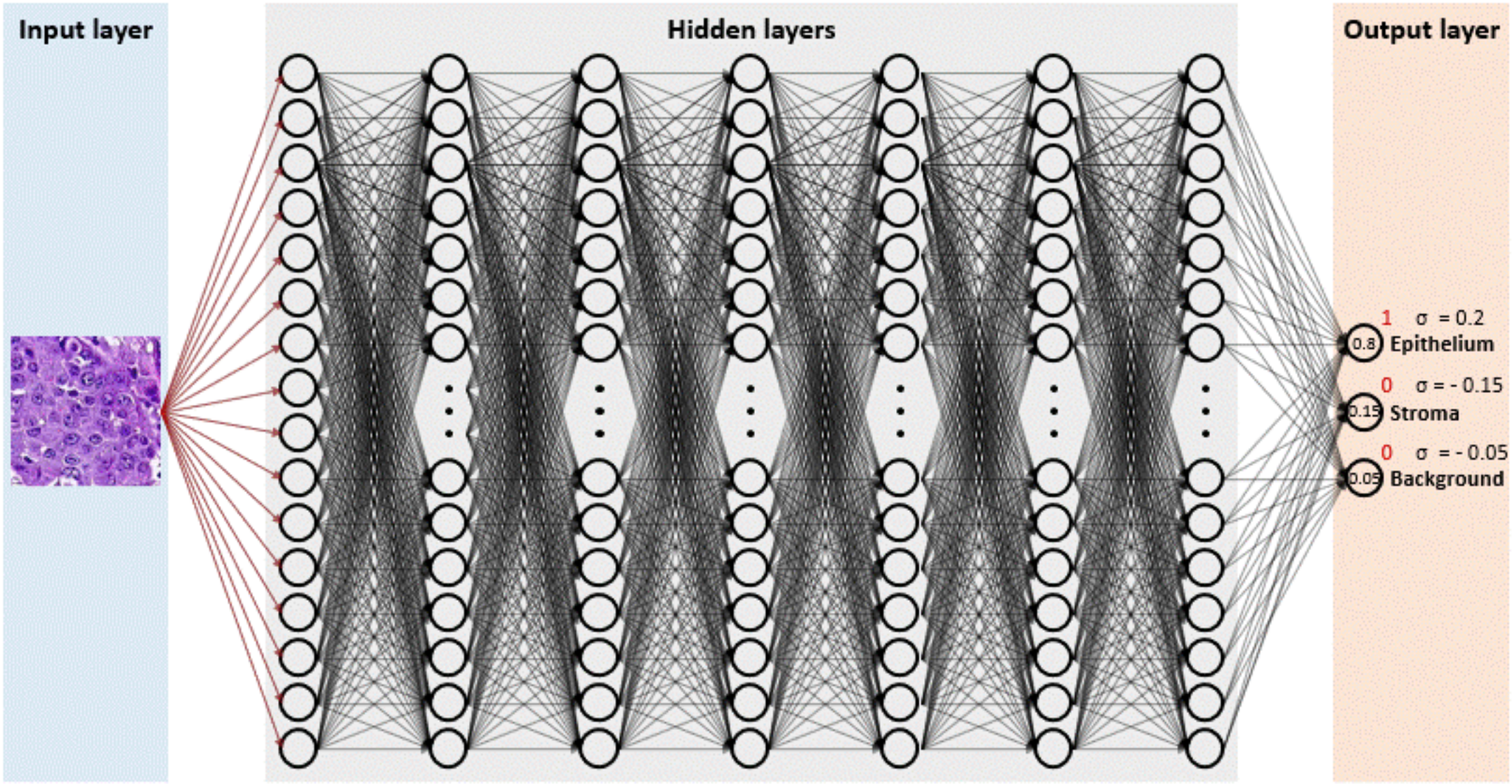
Deep Convolutional Neural Networks



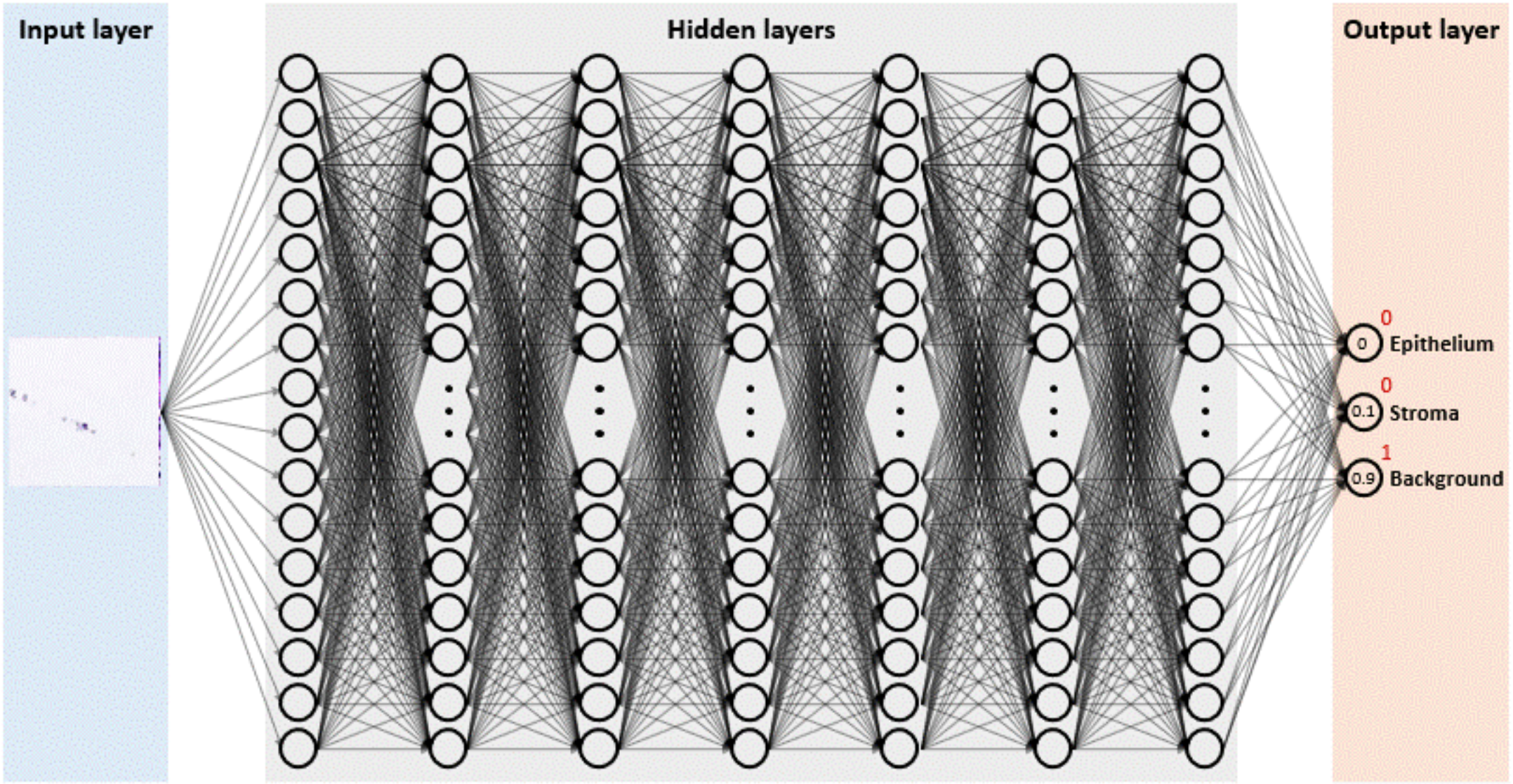
Deep Convolutional Neural Networks



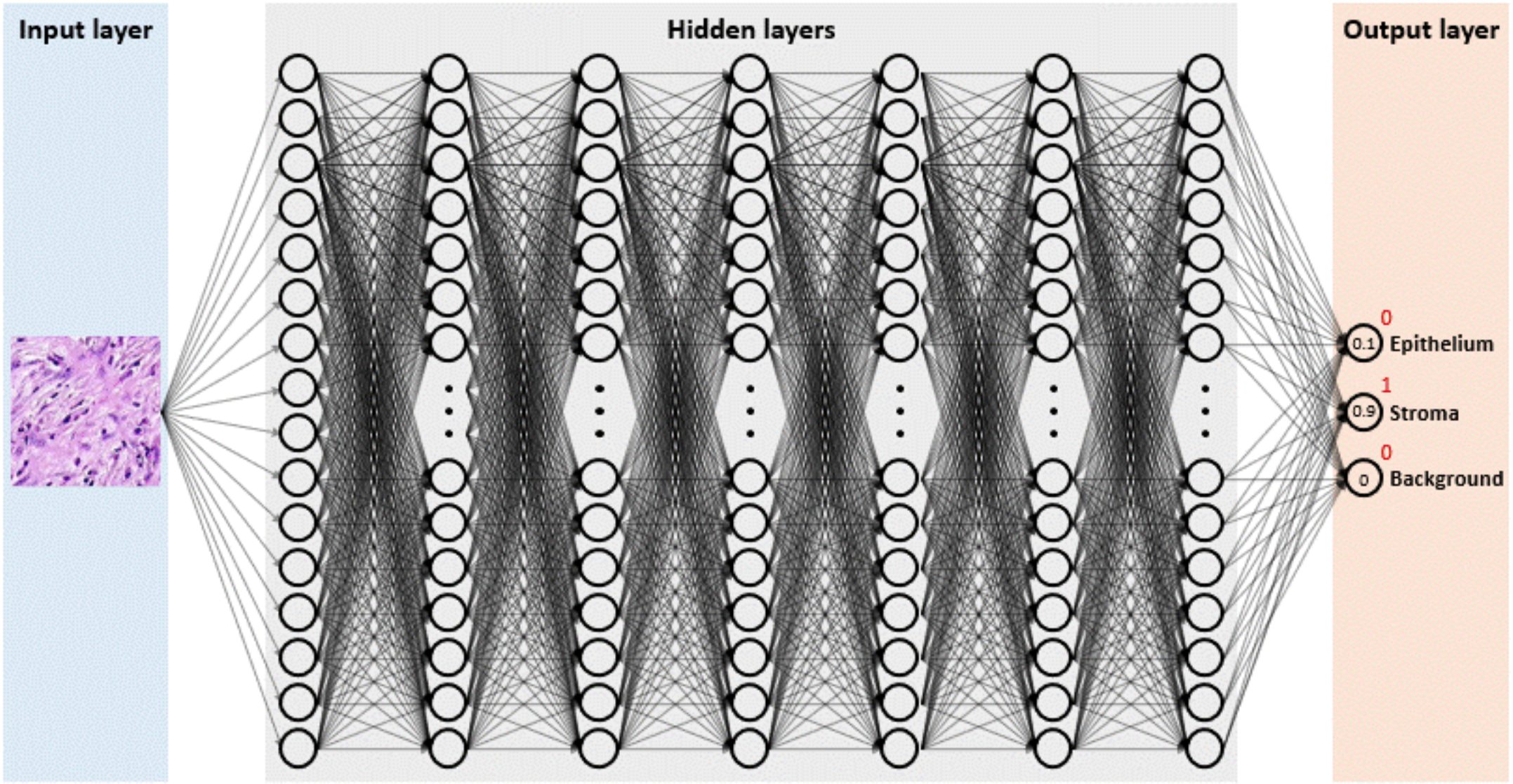
Deep Convolutional Neural Networks



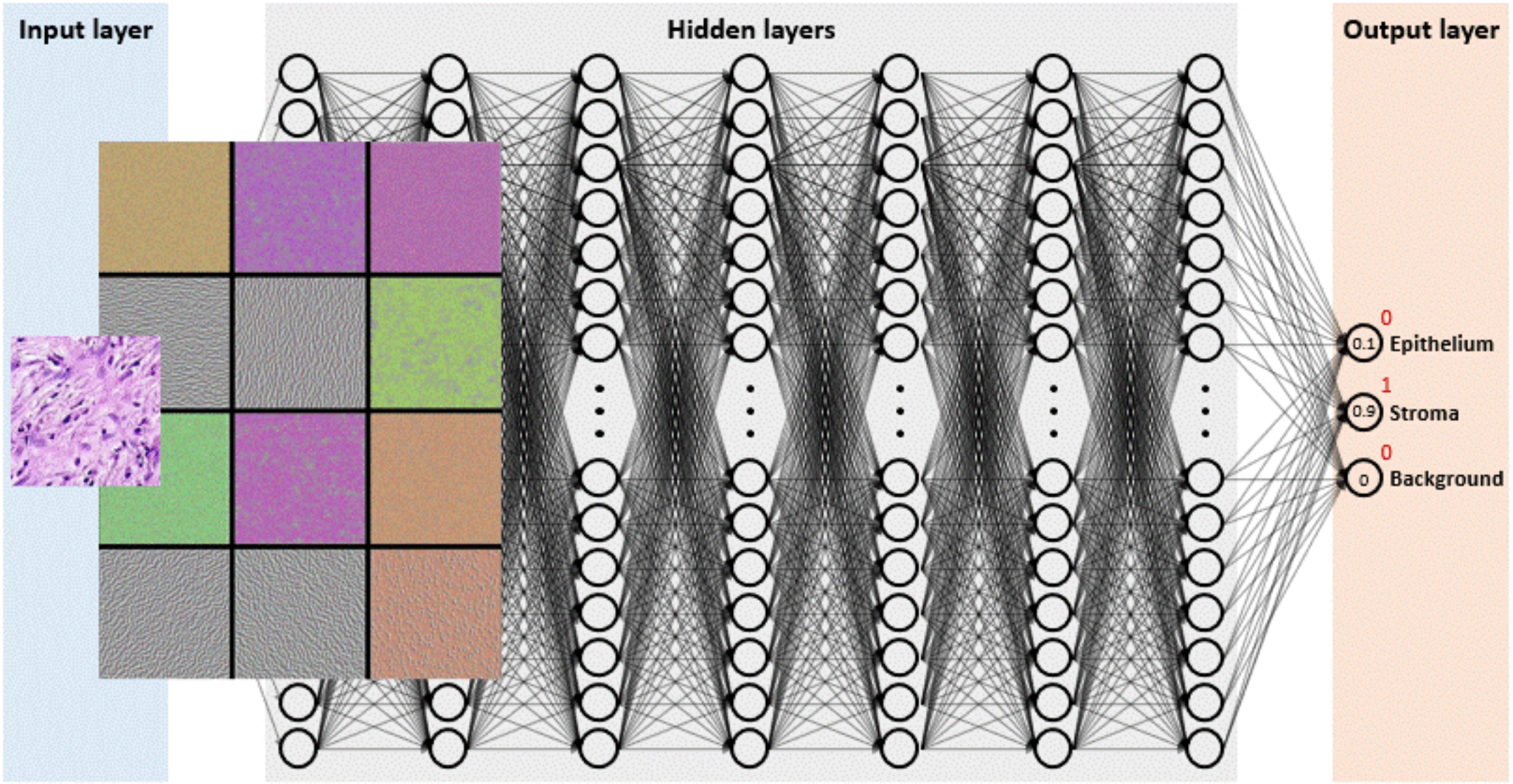
Deep Convolutional Neural Networks



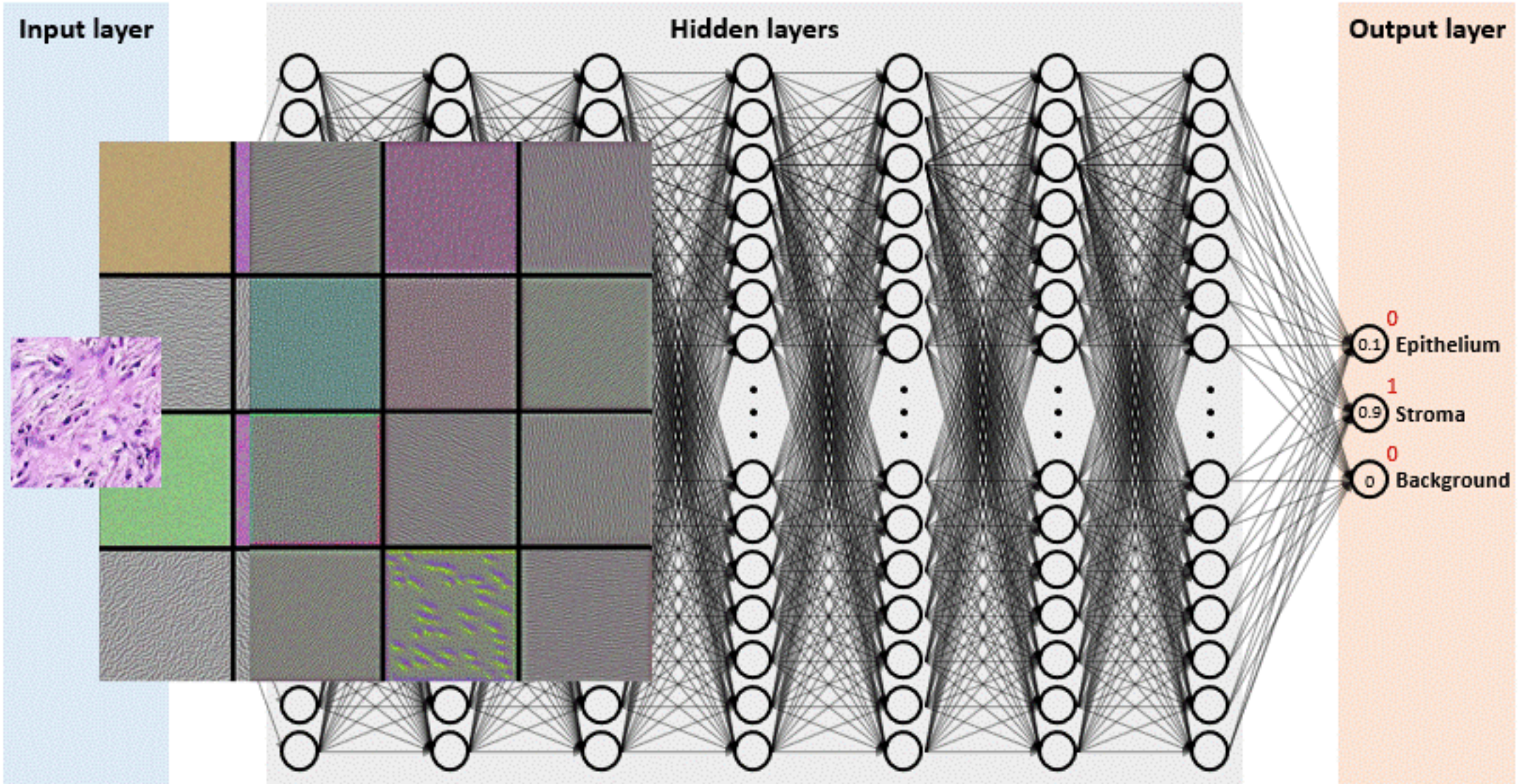
Deep Convolutional Neural Networks



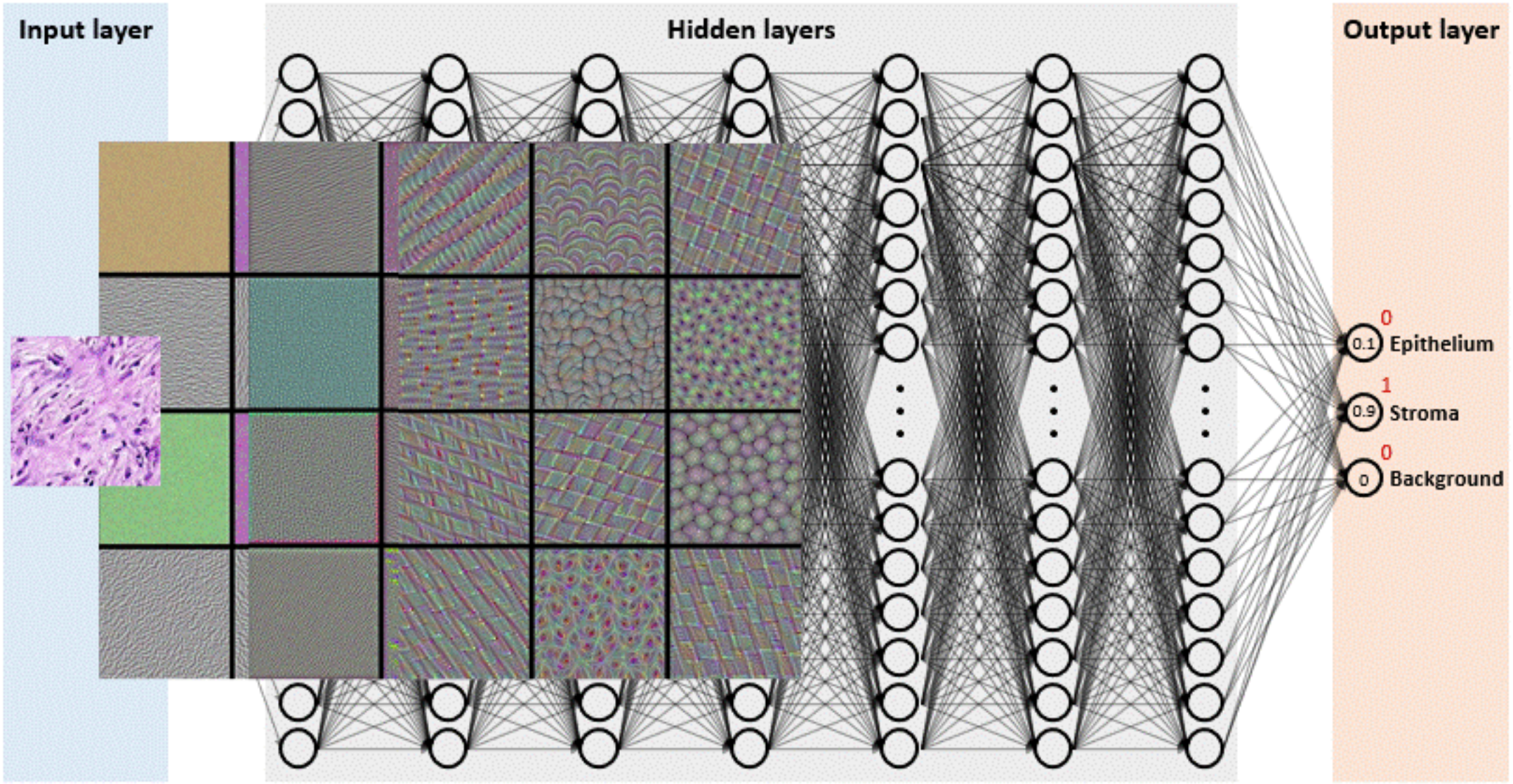
Deep Convolutional Neural Networks



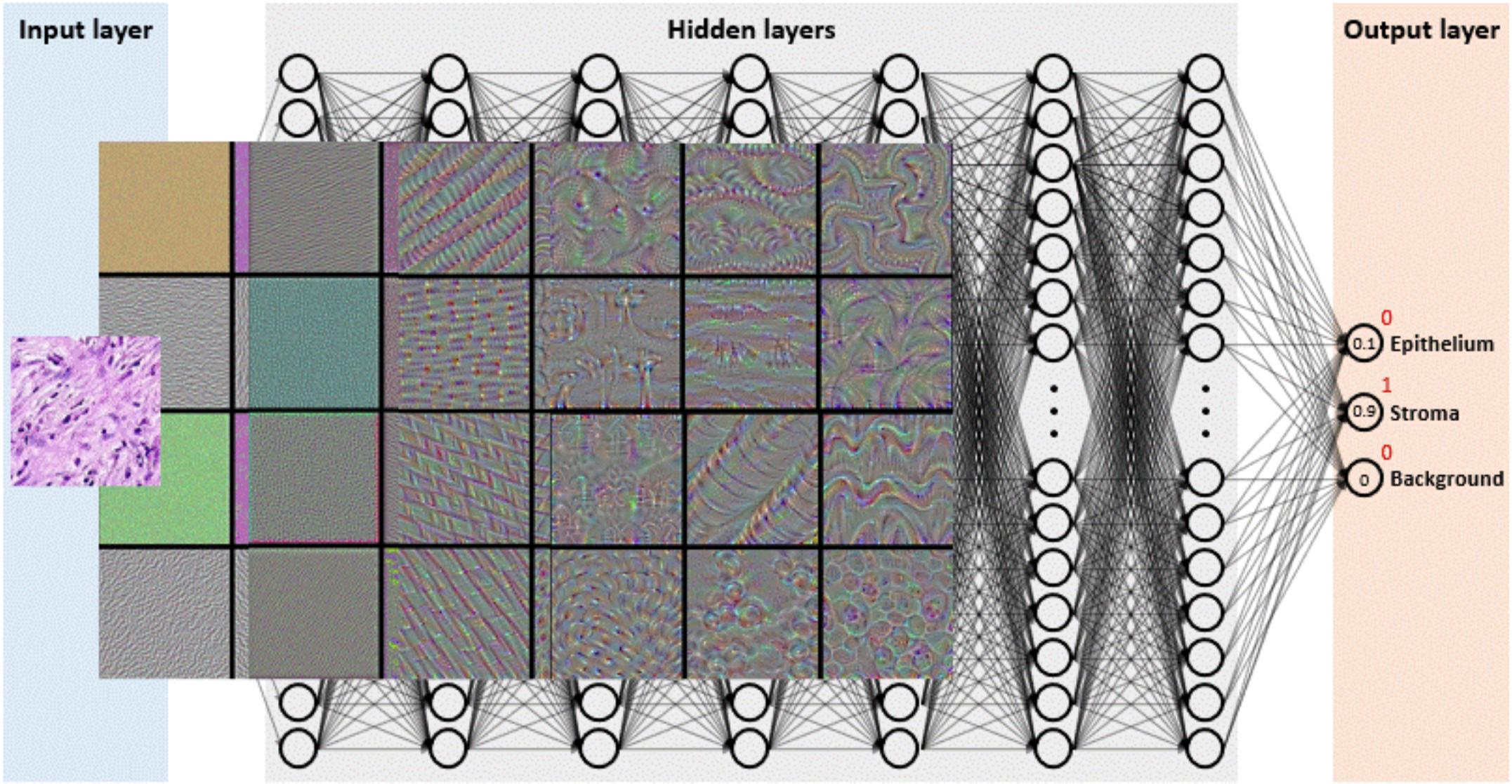
Deep Convolutional Neural Networks



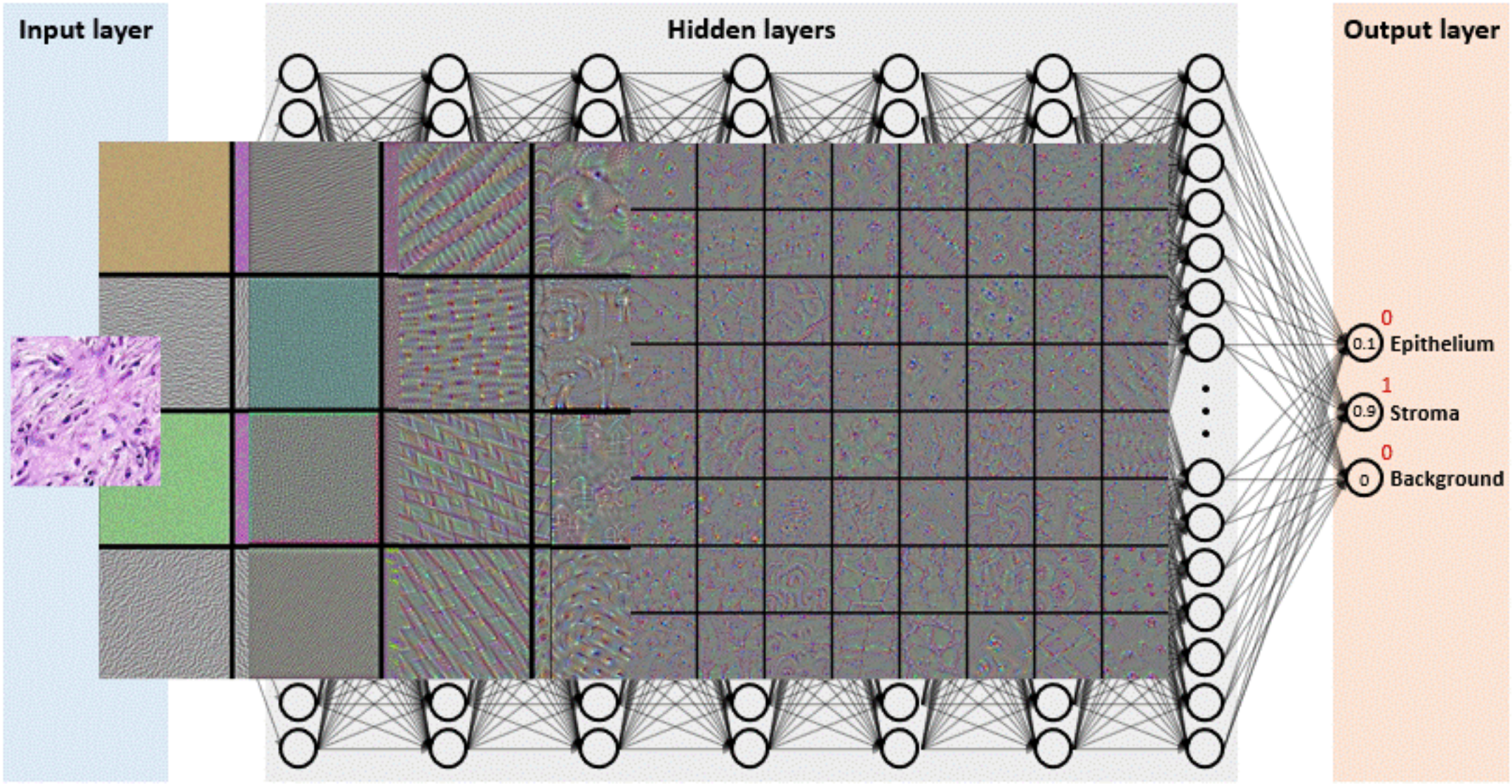
Deep Convolutional Neural Networks



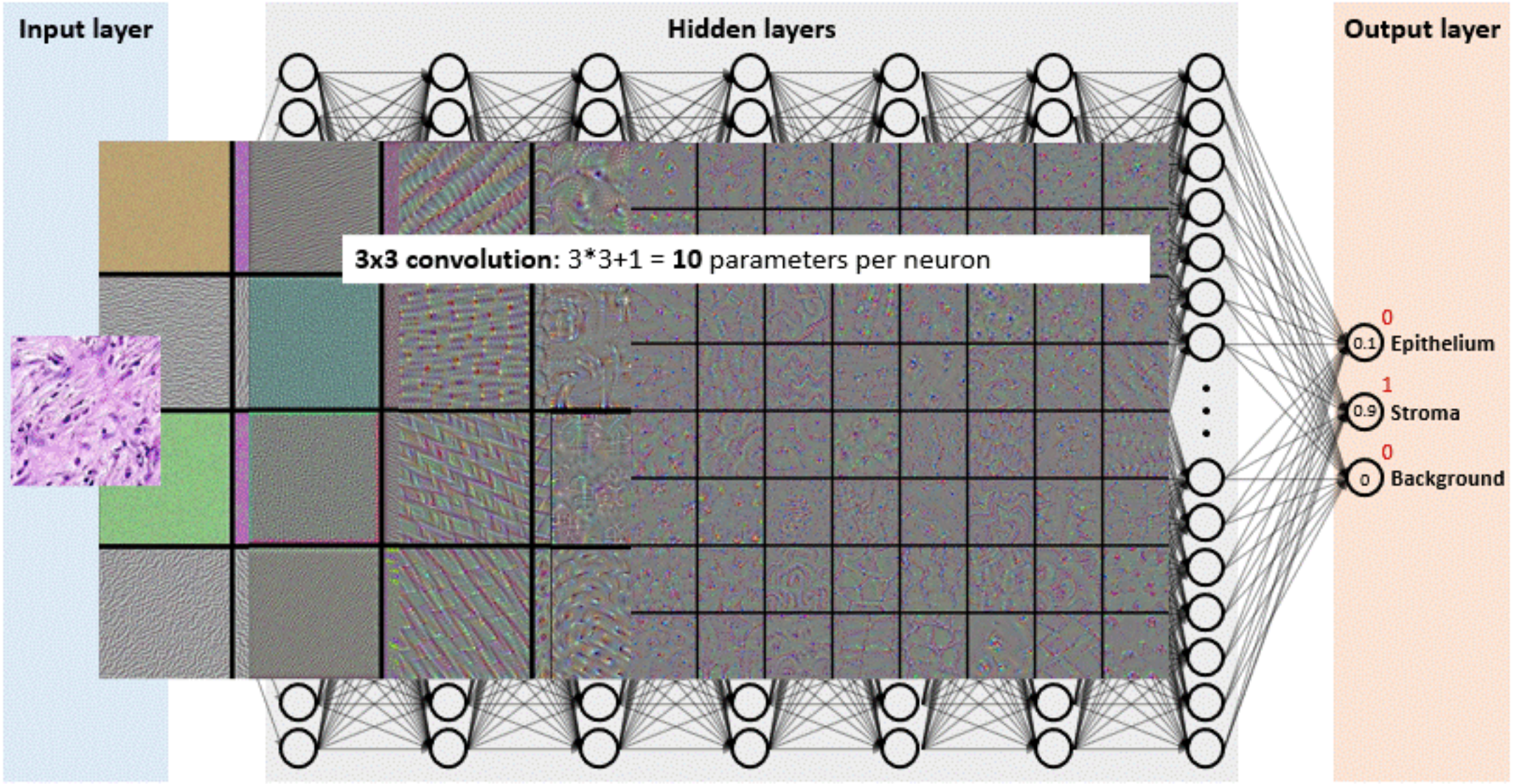
Deep Convolutional Neural Networks



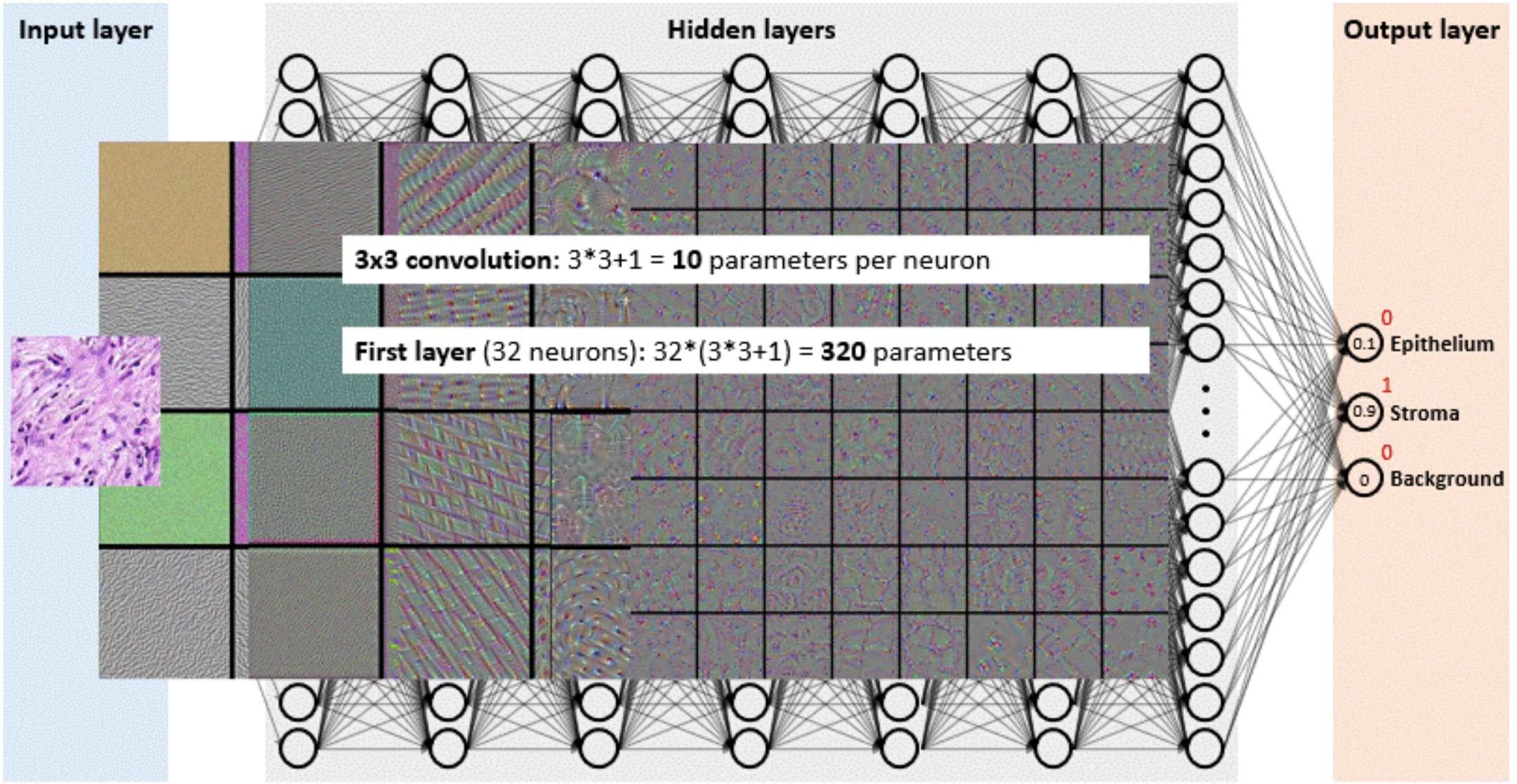
Deep Convolutional Neural Networks



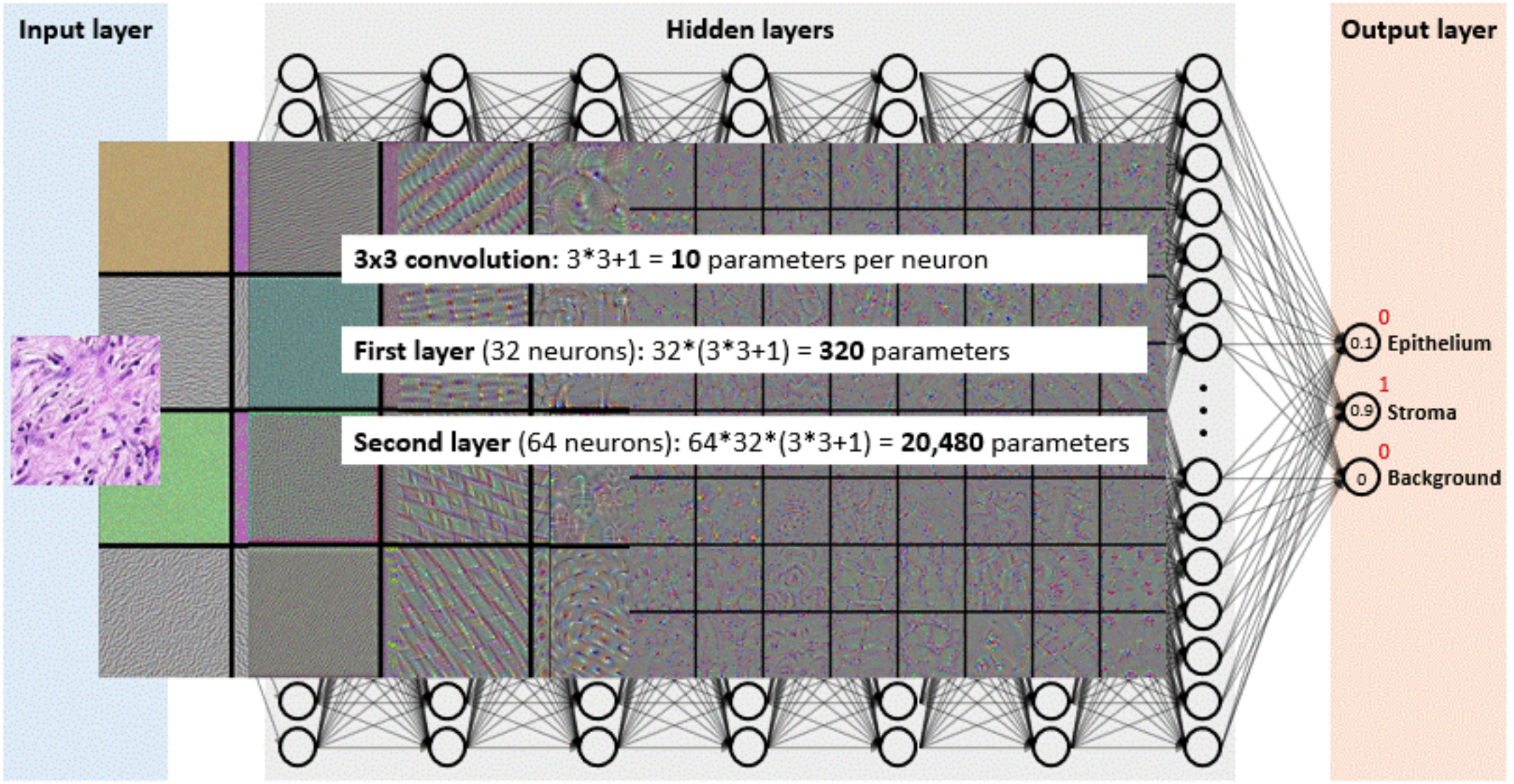
Deep Convolutional Neural Networks



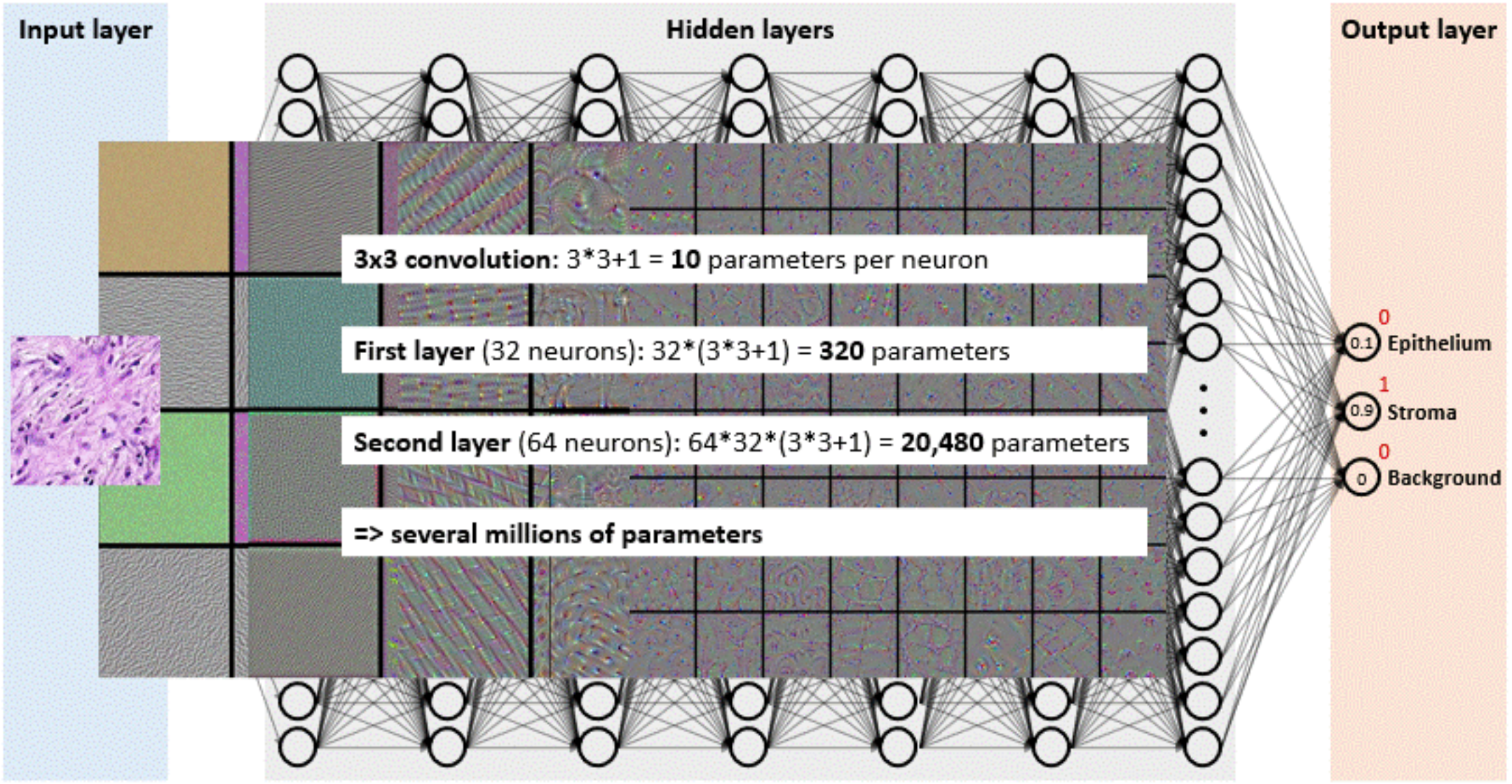
Deep Convolutional Neural Networks



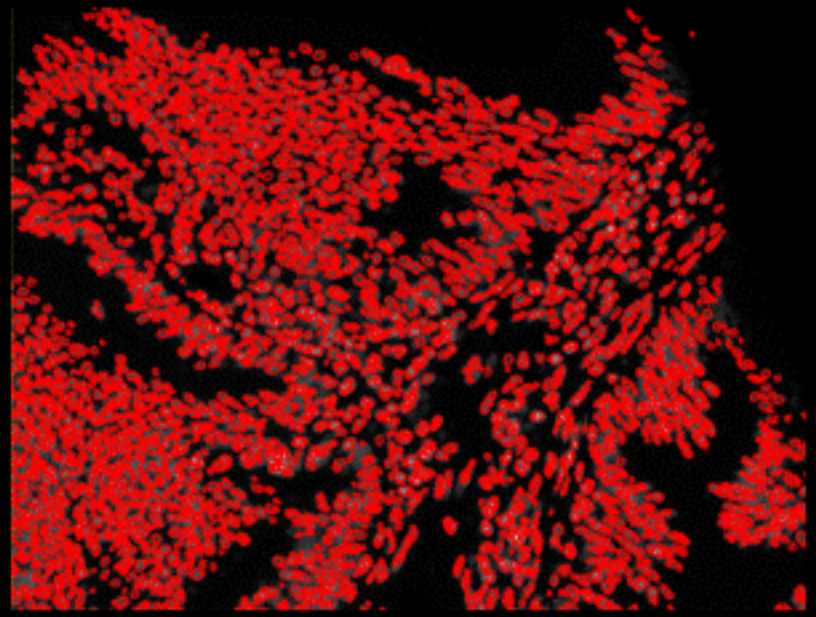
Deep Convolutional Neural Networks

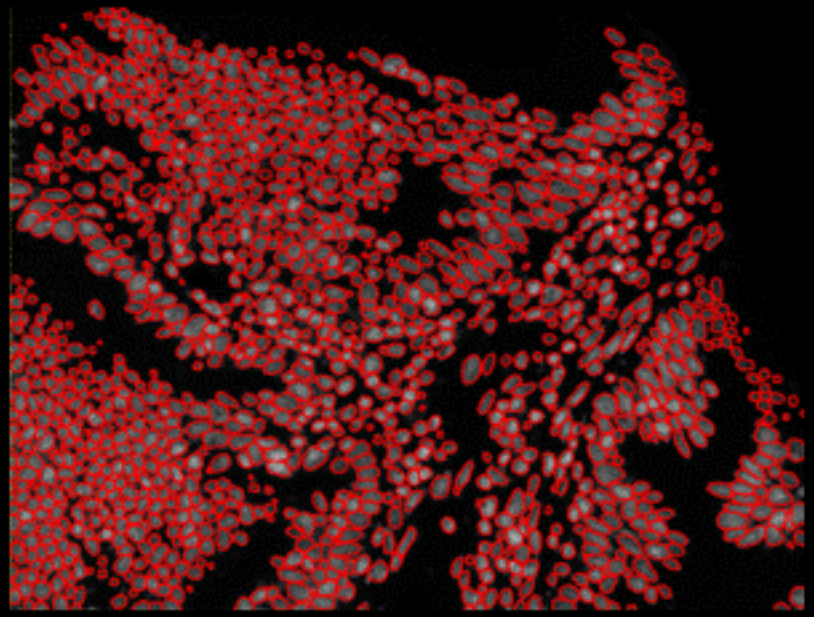


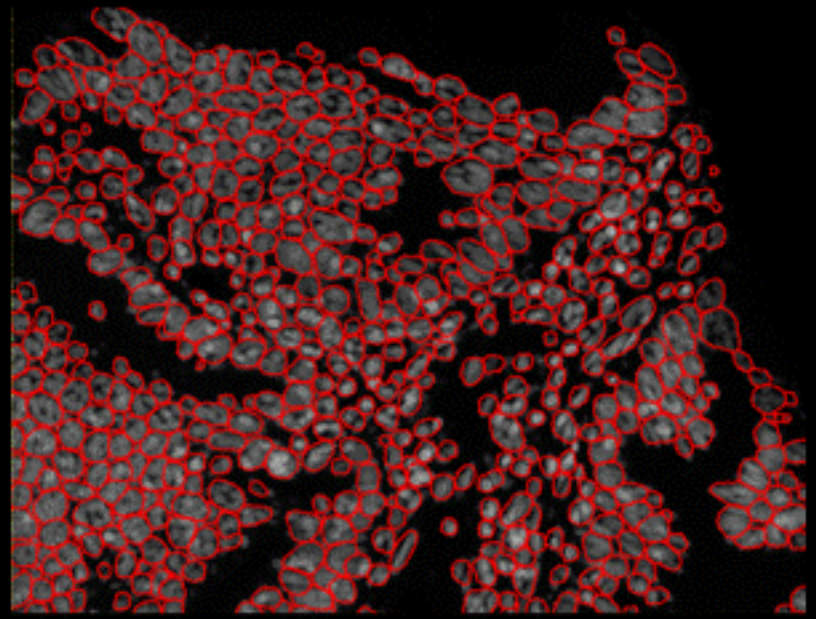
Deep Convolutional Neural Networks

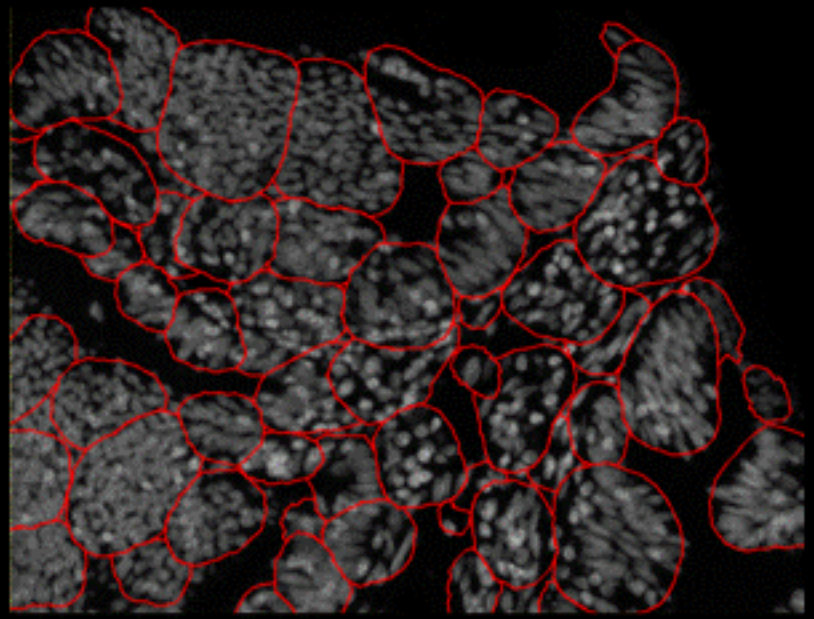


WARNINGS

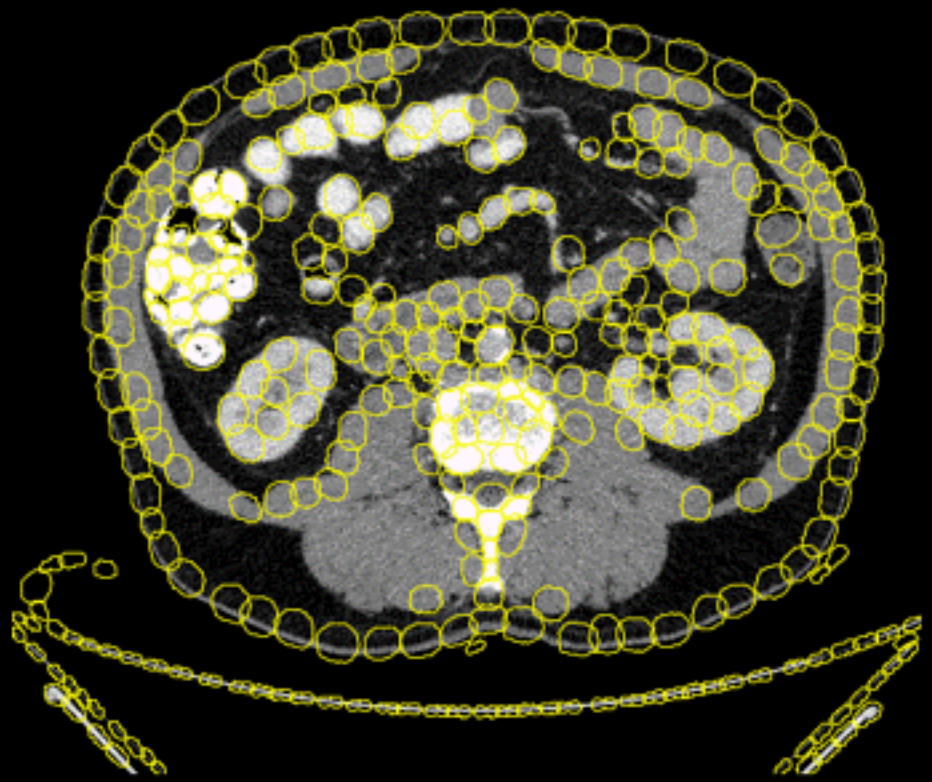


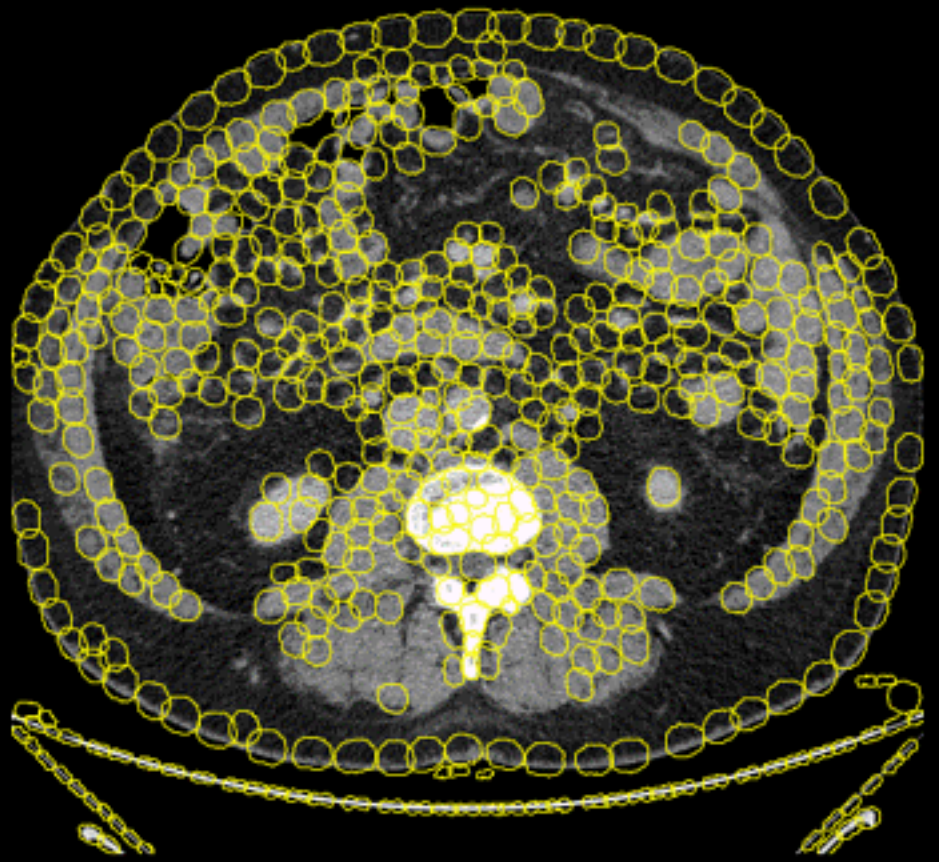






Warnings





NUCLEI SEGMENTATION WITH DEEP LEARNING-BASED METHOD STARDIST

Cell Detection with Star-convex Polygons

Uwe Schmidt[1,*], Martin Weigert[1,*], Coleman Broaddus[1], and Gene Myers[1,2]

[1] Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany
Center for Systems Biology Dresden, Germany
[2] Faculty of Computer Science, Technical University Dresden, Germany

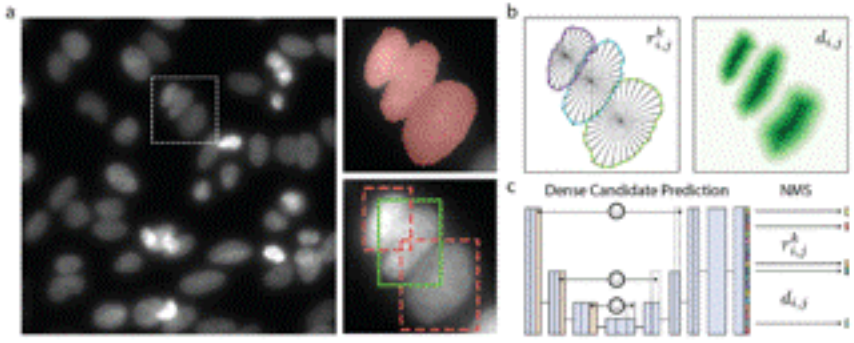


Fig. 1: *(a)* Potential segmentation errors for images with crowded nuclei: Merging of touching cells (upper right) or suppression of valid cell instances due to large overlap of bounding box localization (lower right). *(b)* The proposed StarDist method predicts object probabilities $d_{i,j}$ and star-convex polygons parameterized by the radial distances $r^k_{i,j}$. *(c)* We densely predict $r^k_{i,j}$ and $d_{i,j}$ using a simple U-Net architecture [17] and then select the final instances via non-maximum suppression (NMS).

Nuclei Segmentation with Deep Learning-based Method StarDist

Cell Detection with Star-convex Polygons

Uwe Schmidt[1,*], Martin Weigert[1,*], Coleman Broaddus[1], and Gene Myers[1,2]

[1] Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany
Center for Systems Biology Dresden, Germany
[2] Faculty of Computer Science, Technical University Dresden, Germany

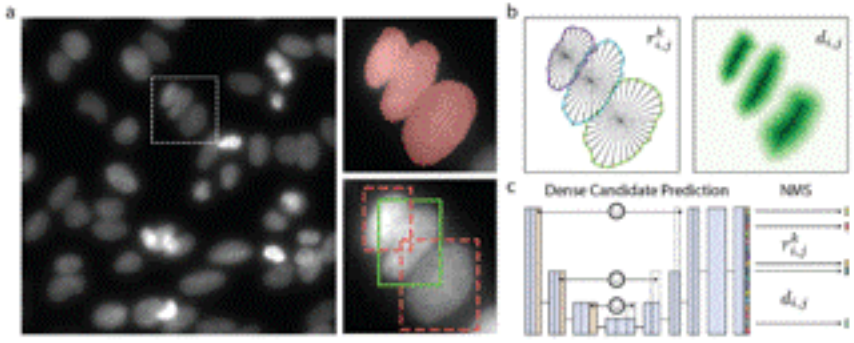
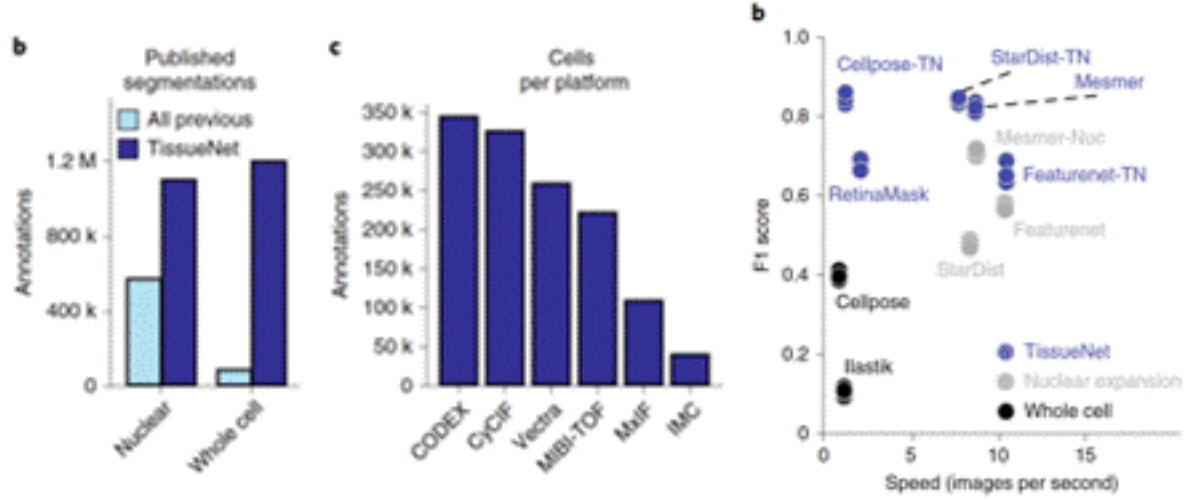


Fig. 1: *(a)* Potential segmentation errors for images with crowded nuclei: Merging of touching cells (upper right) or suppression of valid cell instances due to large overlap of bounding box localization (lower right). *(b)* The proposed StarDist method predicts object probabilities $d_{i,j}$ and star-convex polygons parameterized by the radial distances $r^k_{i,j}$. *(c)* We densely predict $r^k_{i,j}$ and $d_{i,j}$ using a simple U-Net architecture [16] and then select the final instances via non-maximum suppression (NMS).



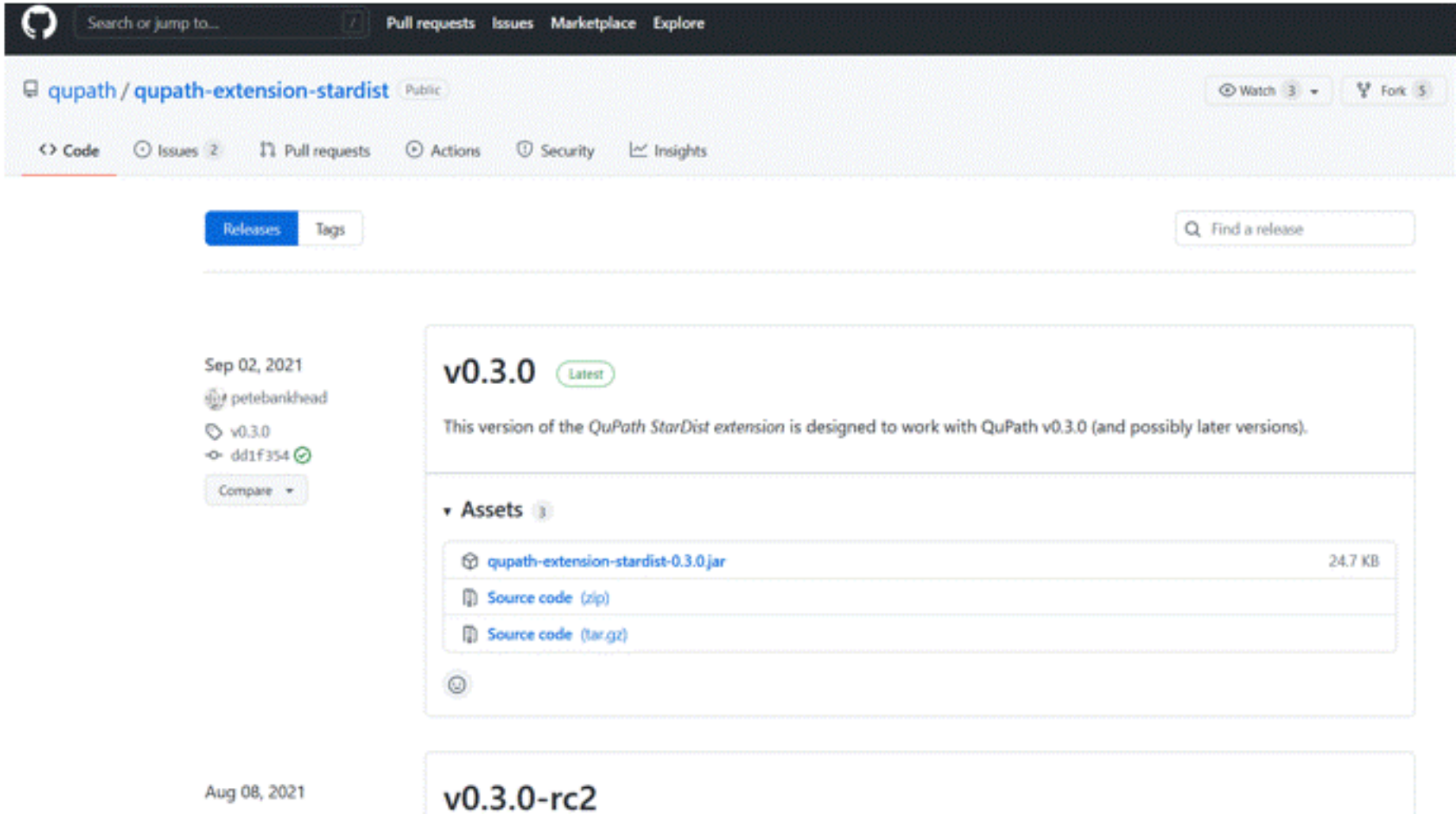
Greenwald et al. **Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning.** *Nature Biotechnology (2022)*

Pécot *et al.* **A deep learning segmentation strategy that minimizes the amount of manually annotated images.** *F1000 Research (2022)*

Pécot *et al.* **Deep learning tools and modeling to estimate the temporal expression of cell cycle proteins from 2D still images.** *PLOS Computational Biology (2022)*

SEGMENTATION WITH STARDIST

Download the latest **Stardist extension** for QuPath and **drag it into QuPath**



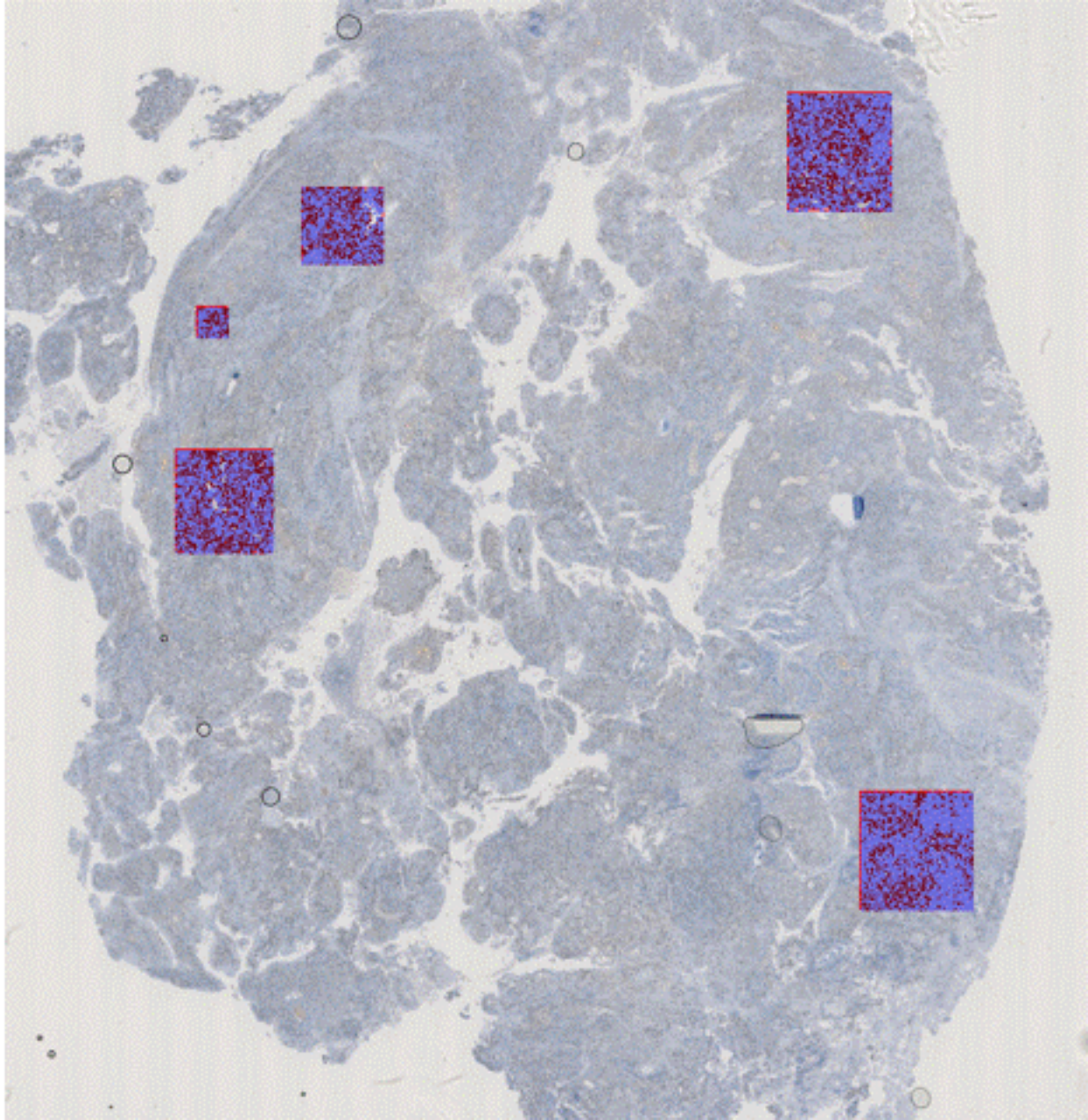
SEGMENTATION WITH STARDIST

Open **Scrip editor**, then open nucleus_detection_hematoxylin_dab.groovy

```groovy
import qupath.ext.stardist.StarDist2D
import qupath.lib.images.servers.ColorTransforms
import qupath.imagej.gui.ImageJMacroRunner

min_nuclei_area = 15

// Specify the model directory (you will need to change this!)
def pathModel = "C:/Work/QuPath/scripts/StardistModels/TissueNet_all.pb"

def stardist_segmentation = StarDist2D.builder(pathModel)
        .threshold(0.5)              // Prediction threshold
        .normalizePercentiles(1, 99.8) // Percentile normalization
        .pixelSize(0.5)              // Resolution for detection
        .channels(
            ColorTransforms.createColorDeconvolvedChannel(getCurrentImageData().getColorDeconvolutionStains(), 1),
            ColorTransforms.createColorDeconvolvedChannel(getCurrentImageData().getColorDeconvolutionStains(), 2)
            )
        .cellExpansion(5.0)          // Approximate cells based upon nucleus expansion
        .cellConstrainScale(1.5)     // Constrain cell expansion using nucleus size
        .measureShape()              // Add shape measurements
        .measureIntensity()          // Add cell measurements (in all compartments)
        .build()


def imageData = getCurrentImageData()
def hierarchy = imageData.getHierarchy()
def annotations = hierarchy.getAnnotationObjects()

// Run detection for the selected objects
stardist_segmentation.detectObjects(imageData, annotations)

//def toDelete = getDetectionObjects().findAll {measurement(it, 'Circularity') < 0.9}
def toDelete = getDetectionObjects().findAll {measurement(it, 'Area µm^2') < min_nuclei_area}
removeObjects(toDelete, true)


println 'Done!'

```

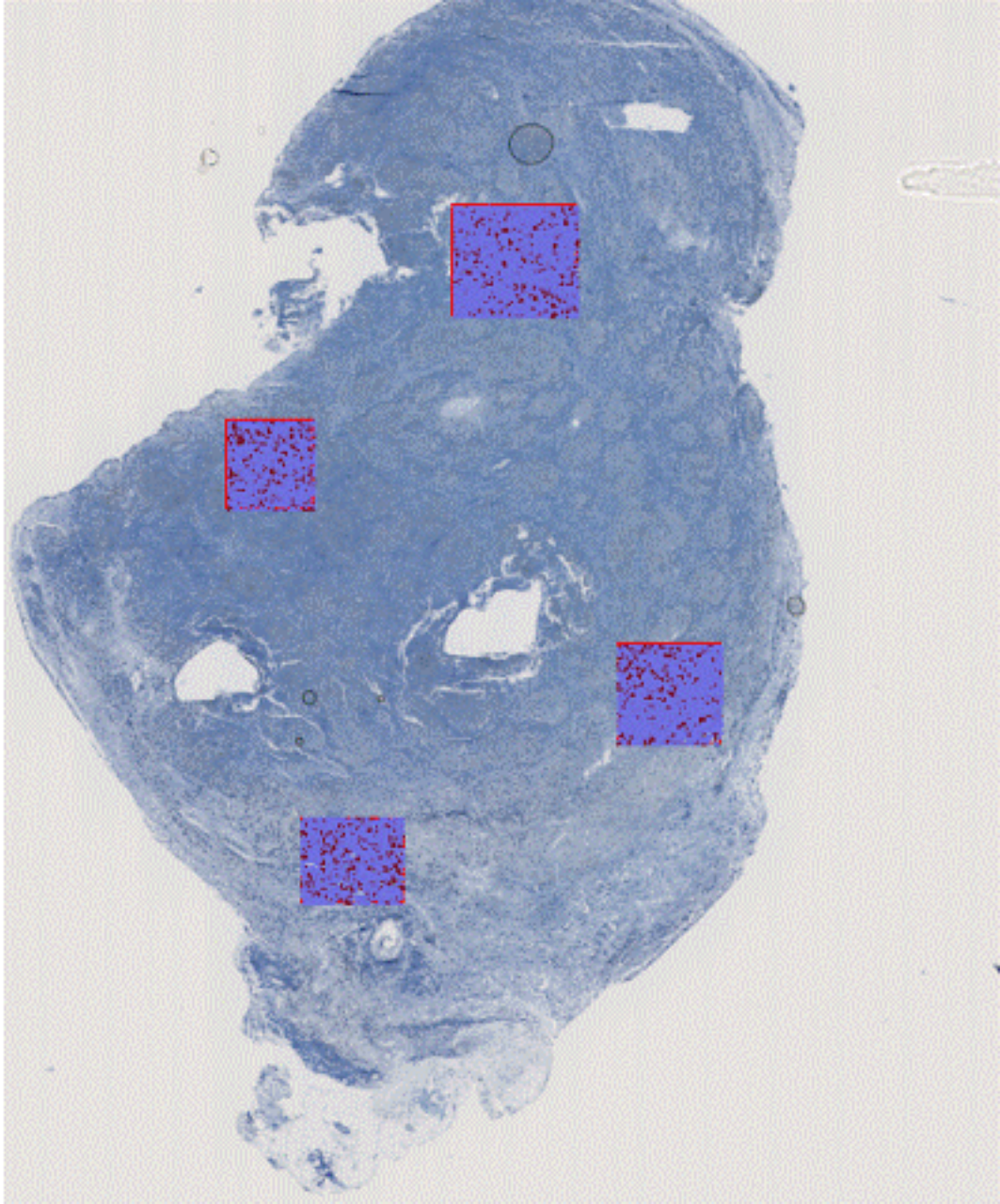
Segmentation with Stardist

- **Open** KI67_lung.ndpi
- Define a **small rectangle annotation** to test the **Stardist parameters**
- Open **Create single measurement classifier** and define threshold to identify DAB+ cells
- Run **stardist** on a small number of annotations and identify DAB+ cells
- Get the **proportion of DAB+ cells**
- **Compare** with the results obtained with the **watershed**-based approach

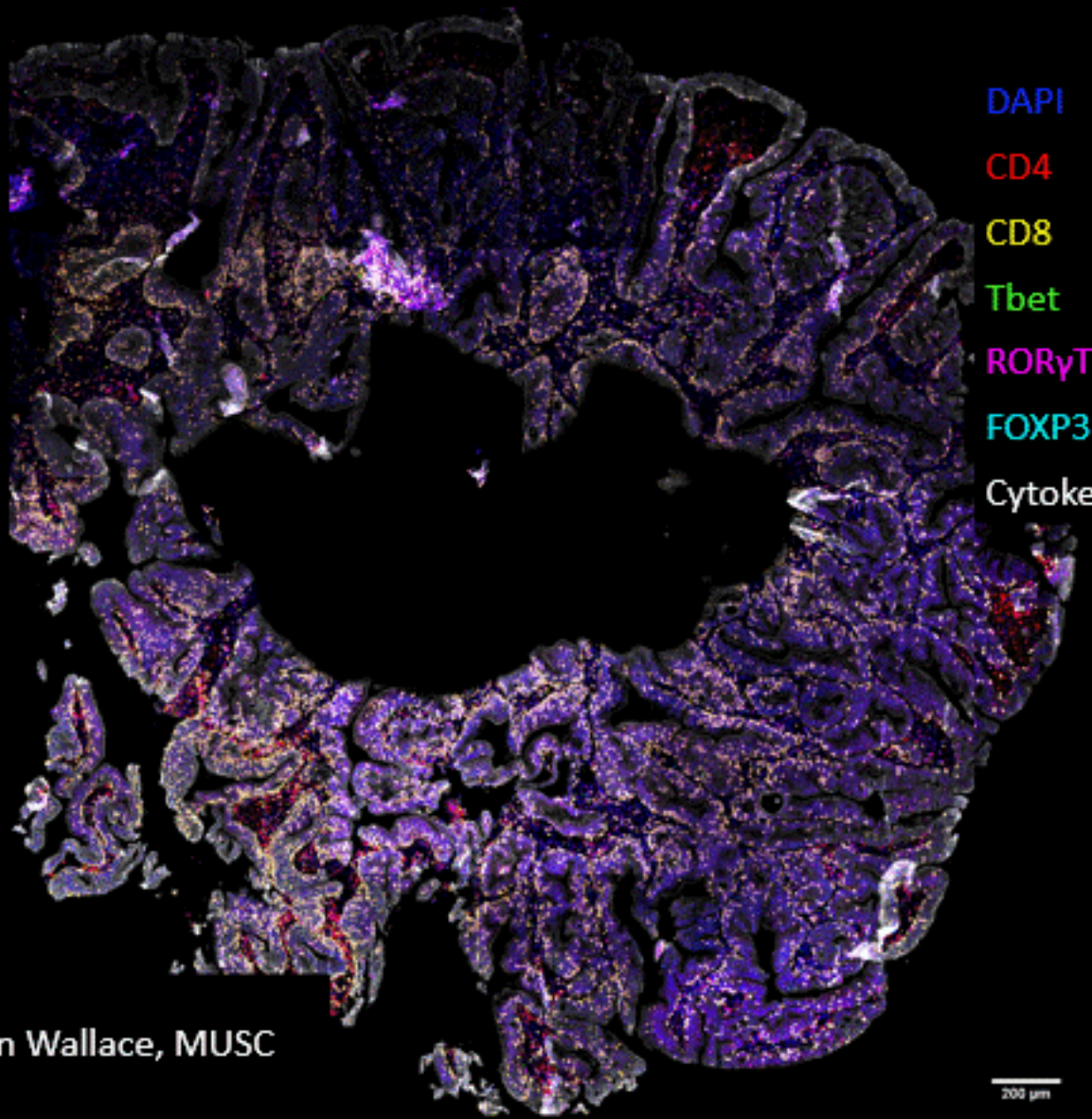


Segmentation with Stardist

- **Open** KI67_lymphoma.ndpi
- Define **3 or 4 ROIs**
- Modify script to do **both segmentation** and **thresholding** (workflow tab)
- Is it a **good way** to **quantify** this image ?

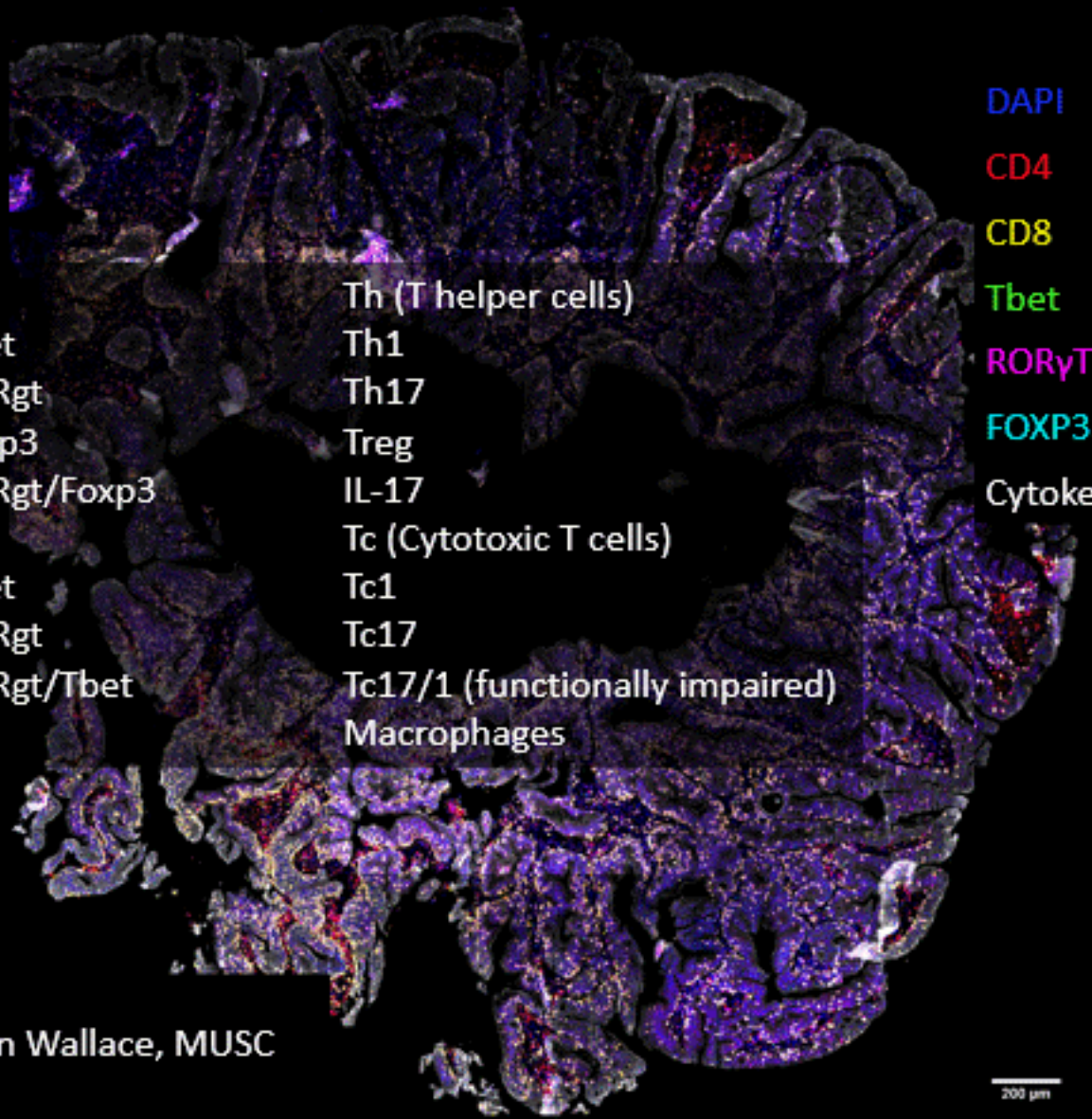


Multi/Hyper-plexed Images



Polyp study with Kristin Wallace, MUSC

Multi/Hyper-plexed Images



CD4	Th (T helper cells)
CD4/Tbet	Th1
CD4/RORgt	Th17
CD4/Foxp3	Treg
CD4/RORgt/Foxp3	IL-17
CD8	Tc (Cytotoxic T cells)
CD8/Tbet	Tc1
CD8/RORgt	Tc17
CD8/RORgt/Tbet	Tc17/1 (functionally impaired)
FOXP3	Macrophages

Polyp study with Kristin Wallace, MUSC

Shallow Machine Learning for Object Classification

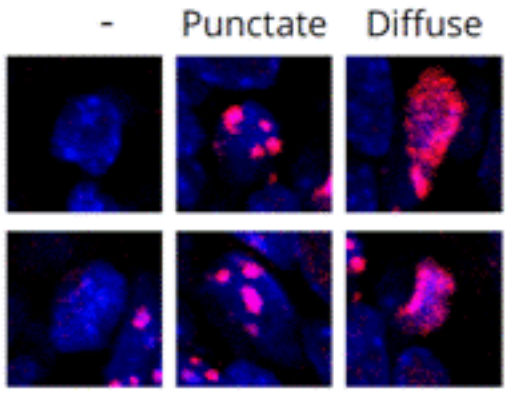
As for pixel classification:

- **Examples** of classes are **manually** defined by the user
- A **classifier** is **trained** with these examples
- Data is then **automatically classified** by using the trained classifier

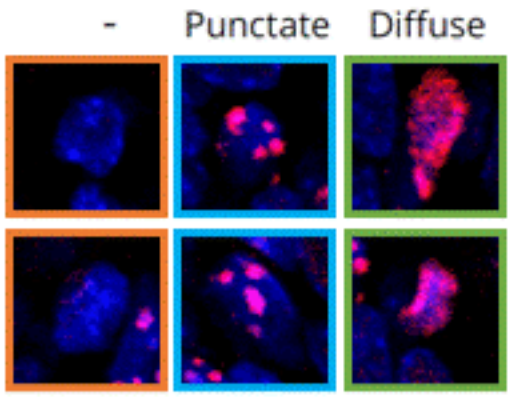
But this time, features are **measurements associated to detections** (most often cells or nuclei) such as:

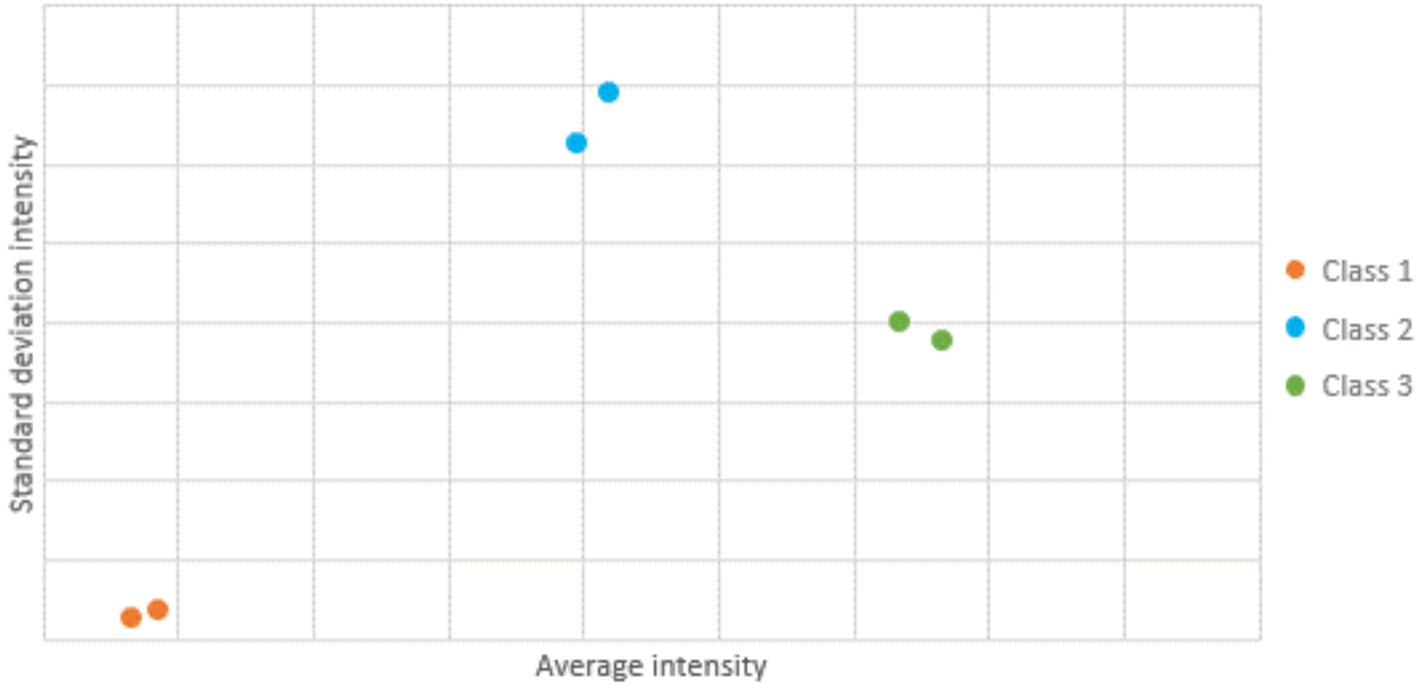
- **Average** intensity
- **Median** intensity
- **Standard deviation** of intensity
- **Minimum/Maximum** intensity
- Object **area**
- Object **Circularity**
- ...

Shallow Machine Learning for Object Classification

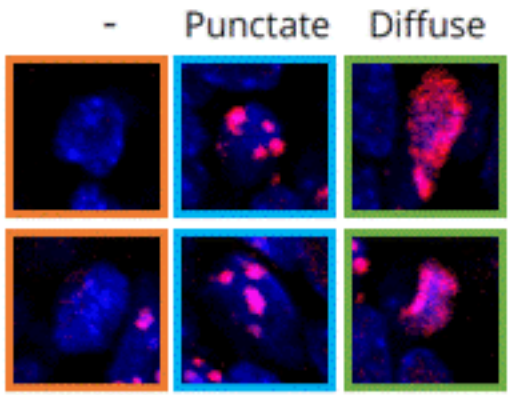


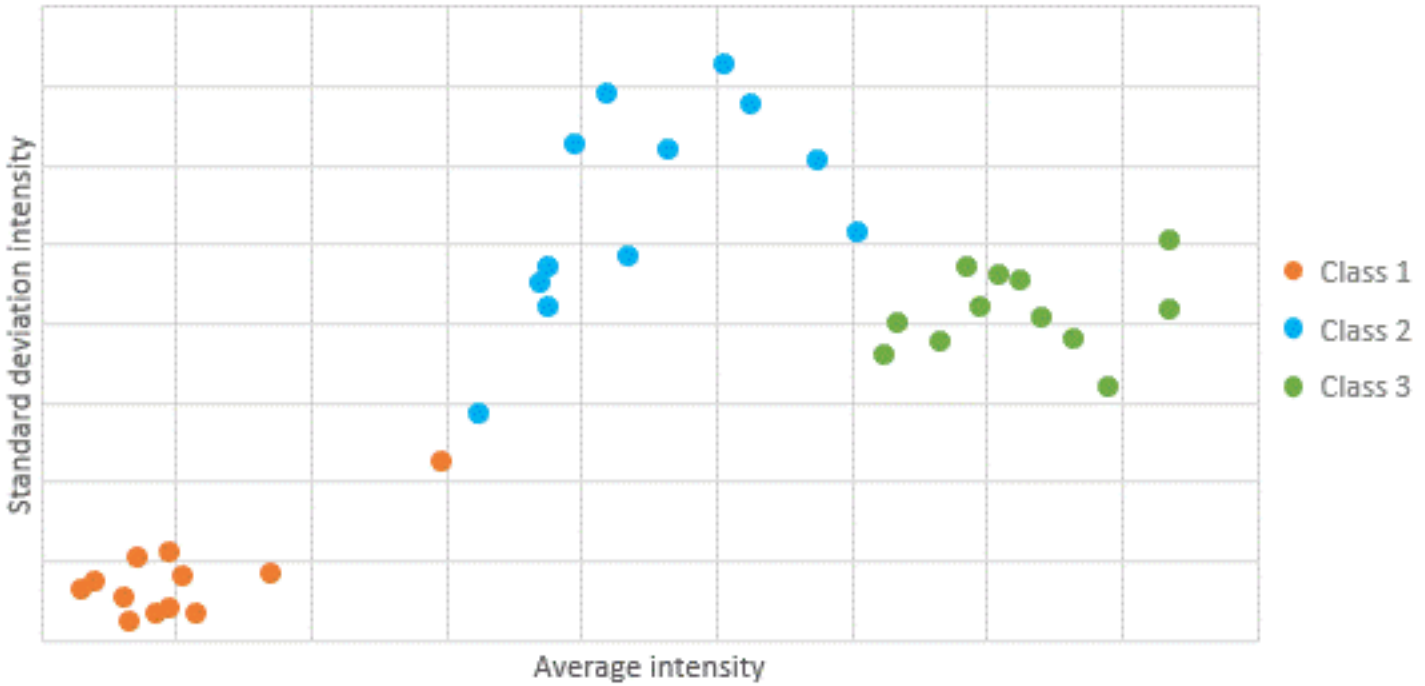
Shallow Machine Learning for Object Classification



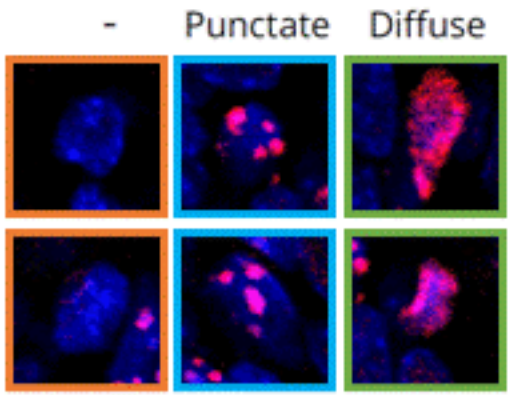


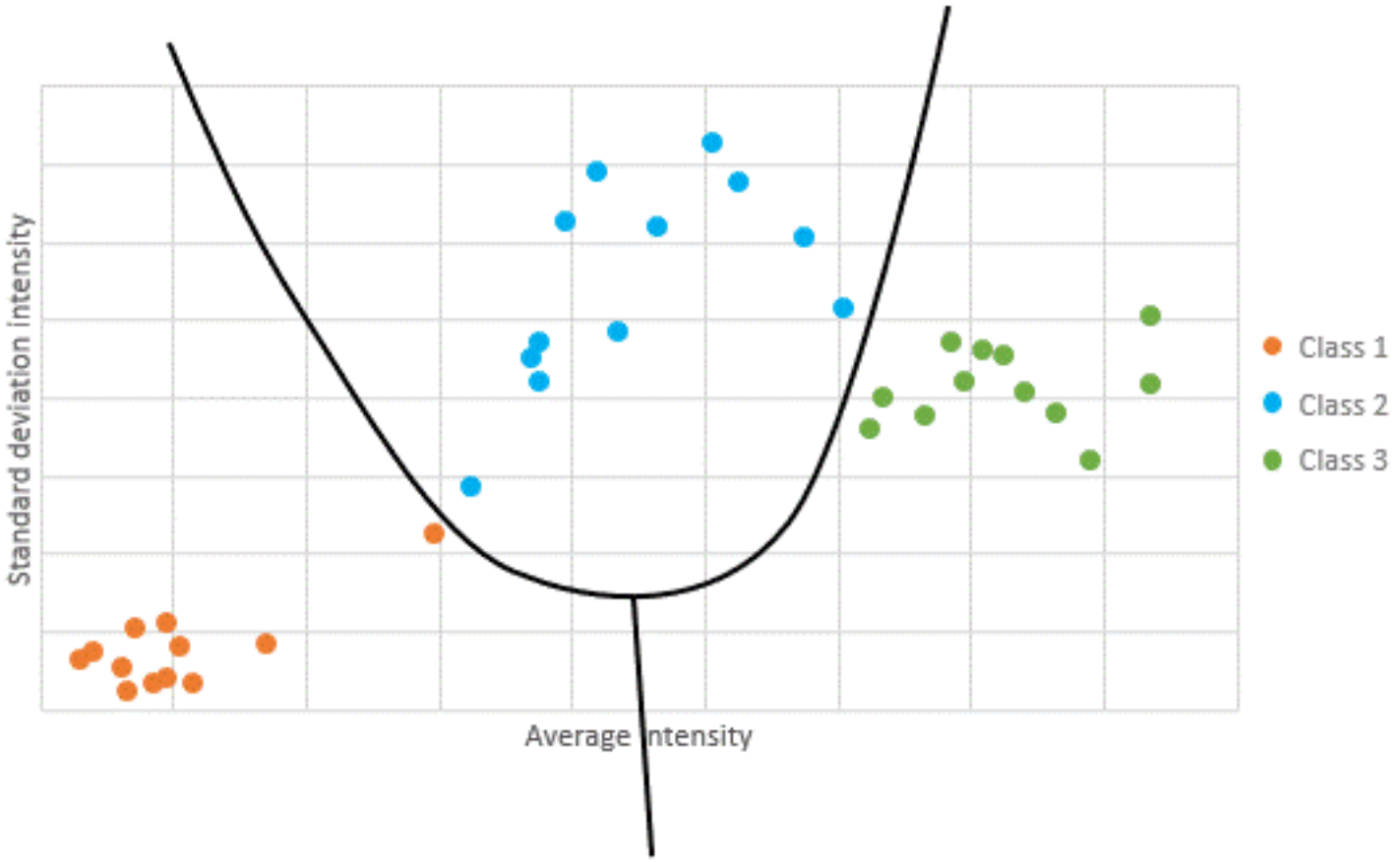
Shallow Machine Learning for Object Classification





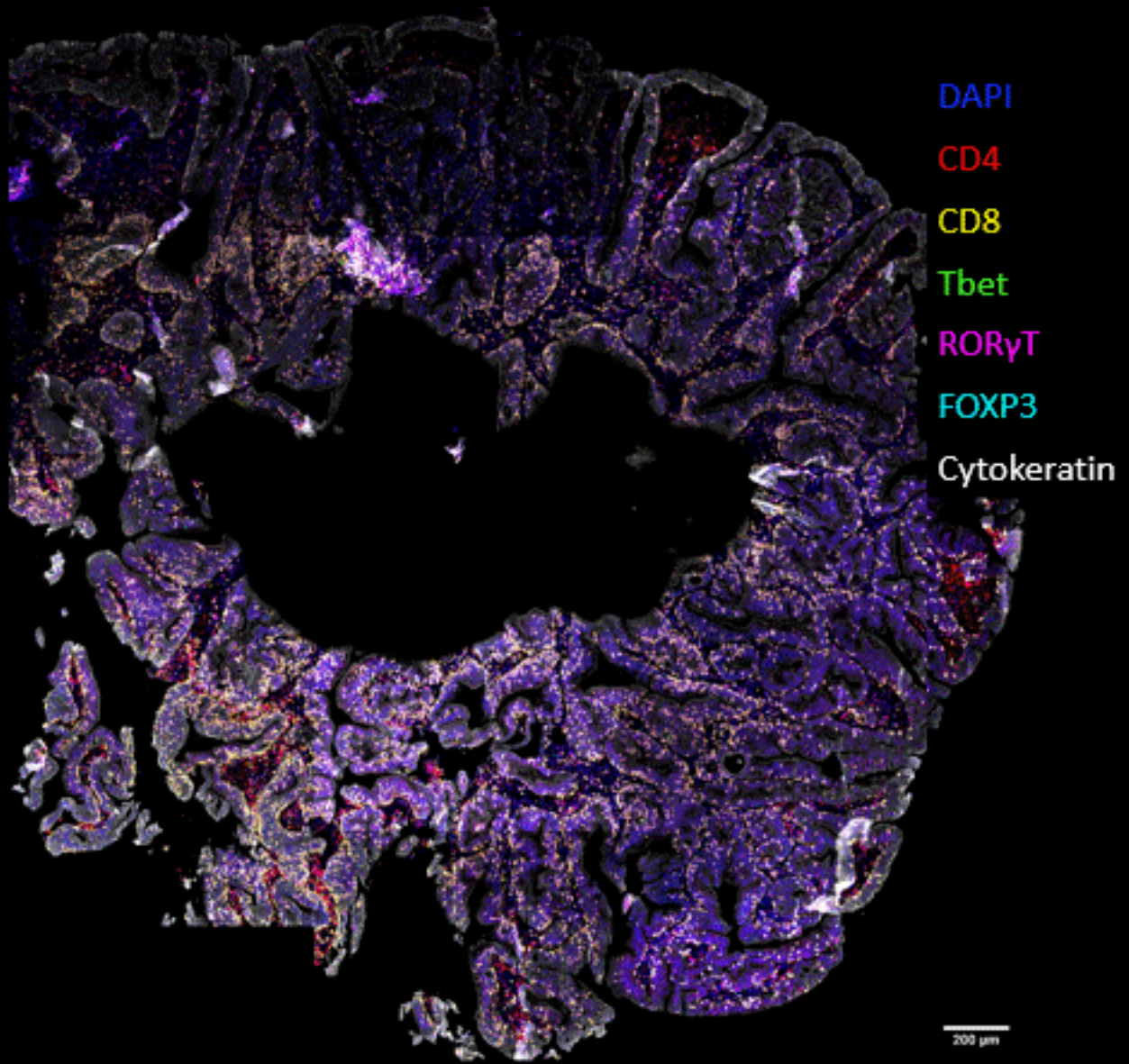
Shallow Machine Learning for Object Classification



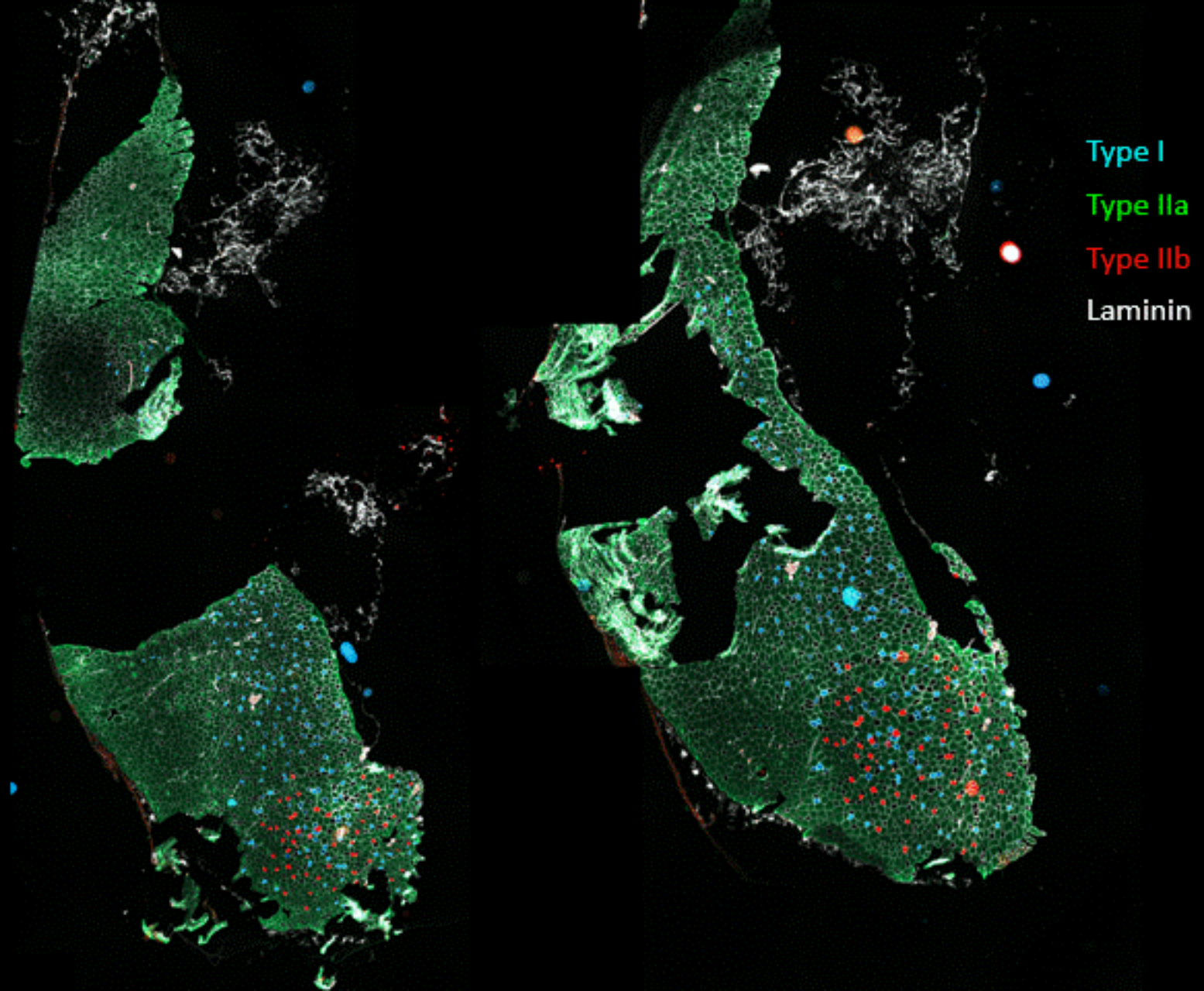


Multi/Hyper-plexed Images

- Use a pixel classifier to segment epithelium and stroma, save it and run it
- Define a small annotation and run Stardist to optimize parameters for nuclei segmentation
- Apply Stardist to the entire image
- Train an object classifier to identify positive cells for each marker
- Compute distances to tissues and between cell types
- Export measurements

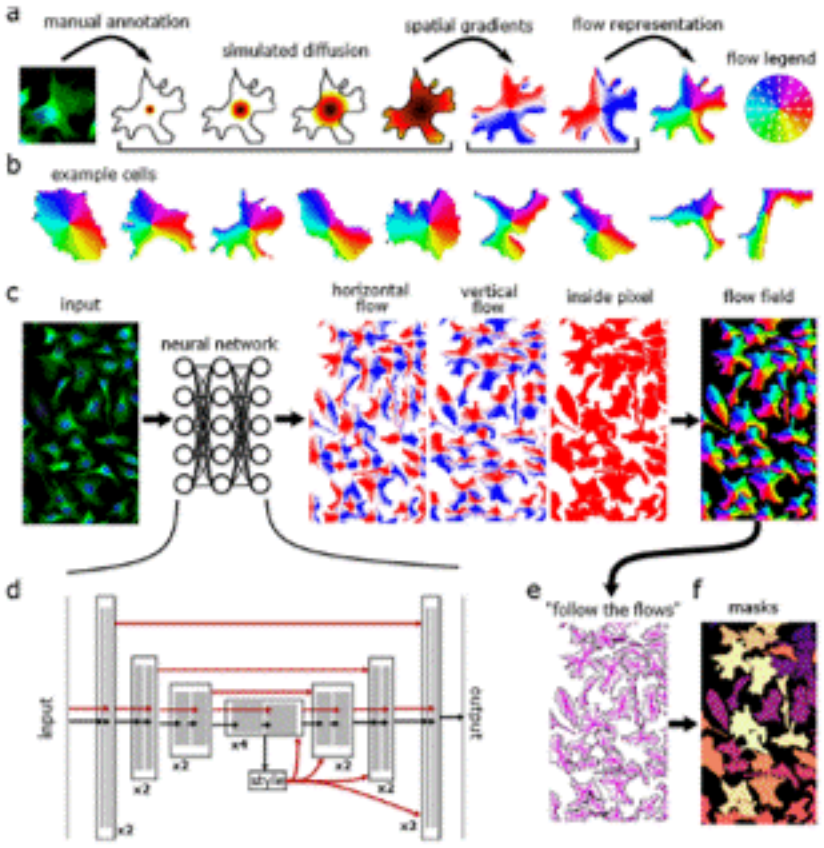


Fluorescence Images of Muscle Fibers

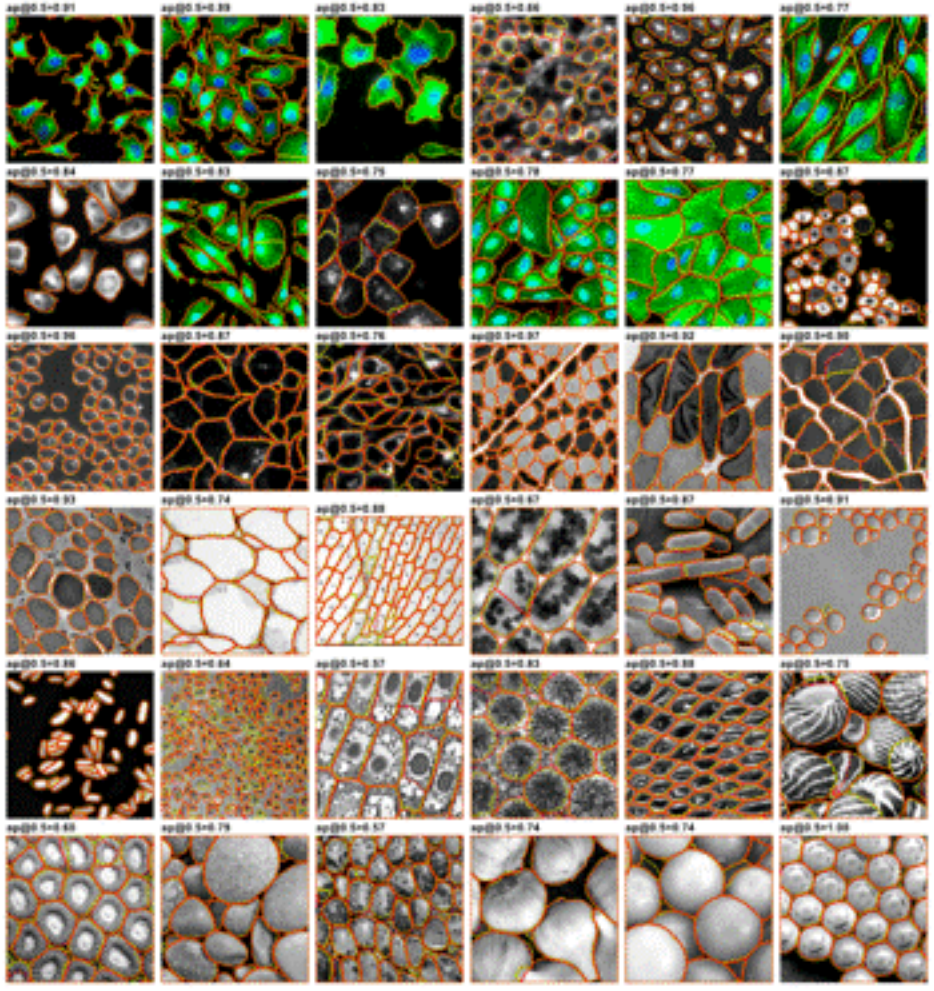


Acquired at APEX

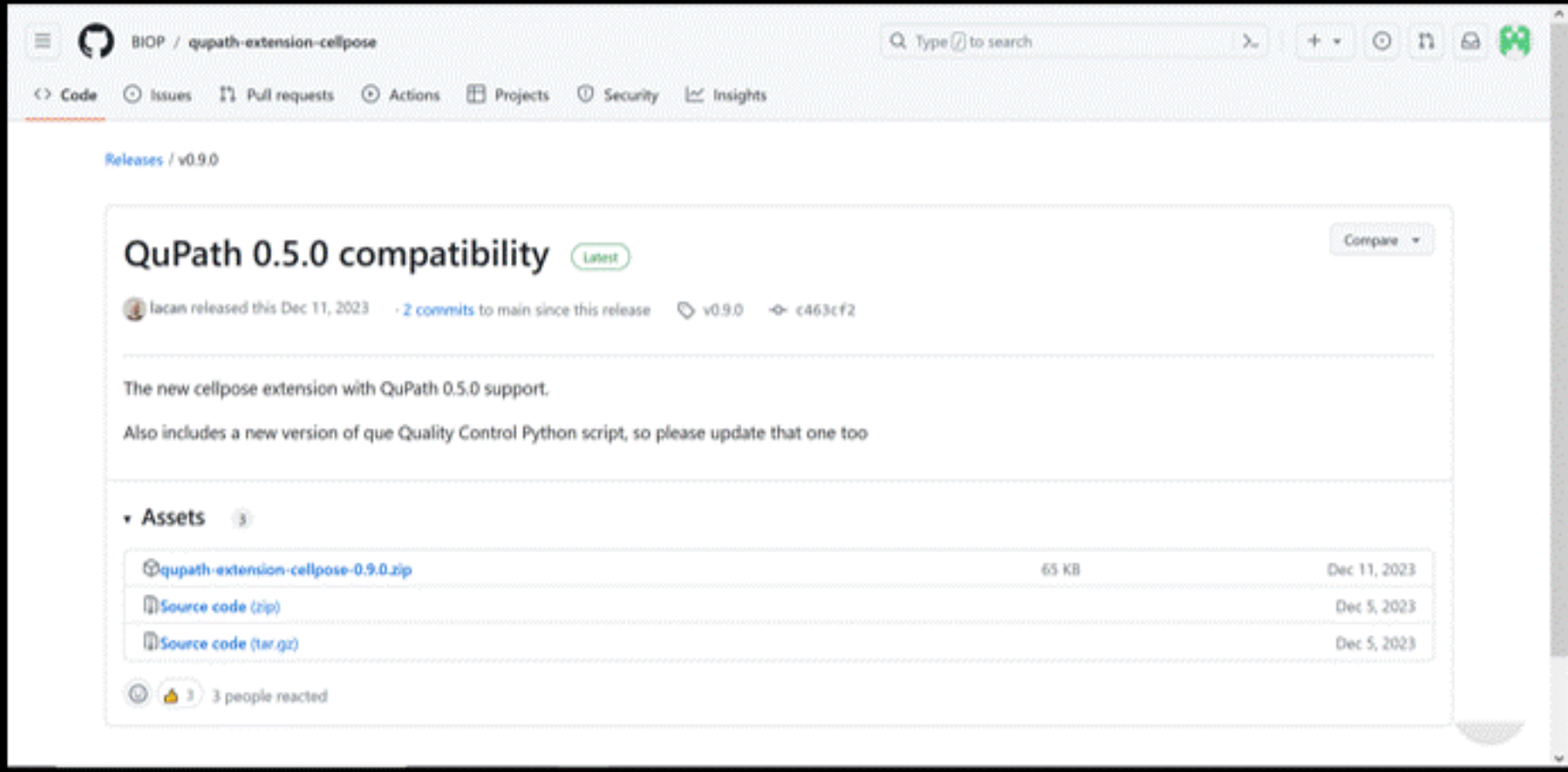
CELLPOSE



Stringer *et al.*, Cellpose: a generalist algorithm for cellular segmentation. *Nature Methods*, 2021



Download the latest **Cellpose extension** for QuPath and **drag it into QuPath**



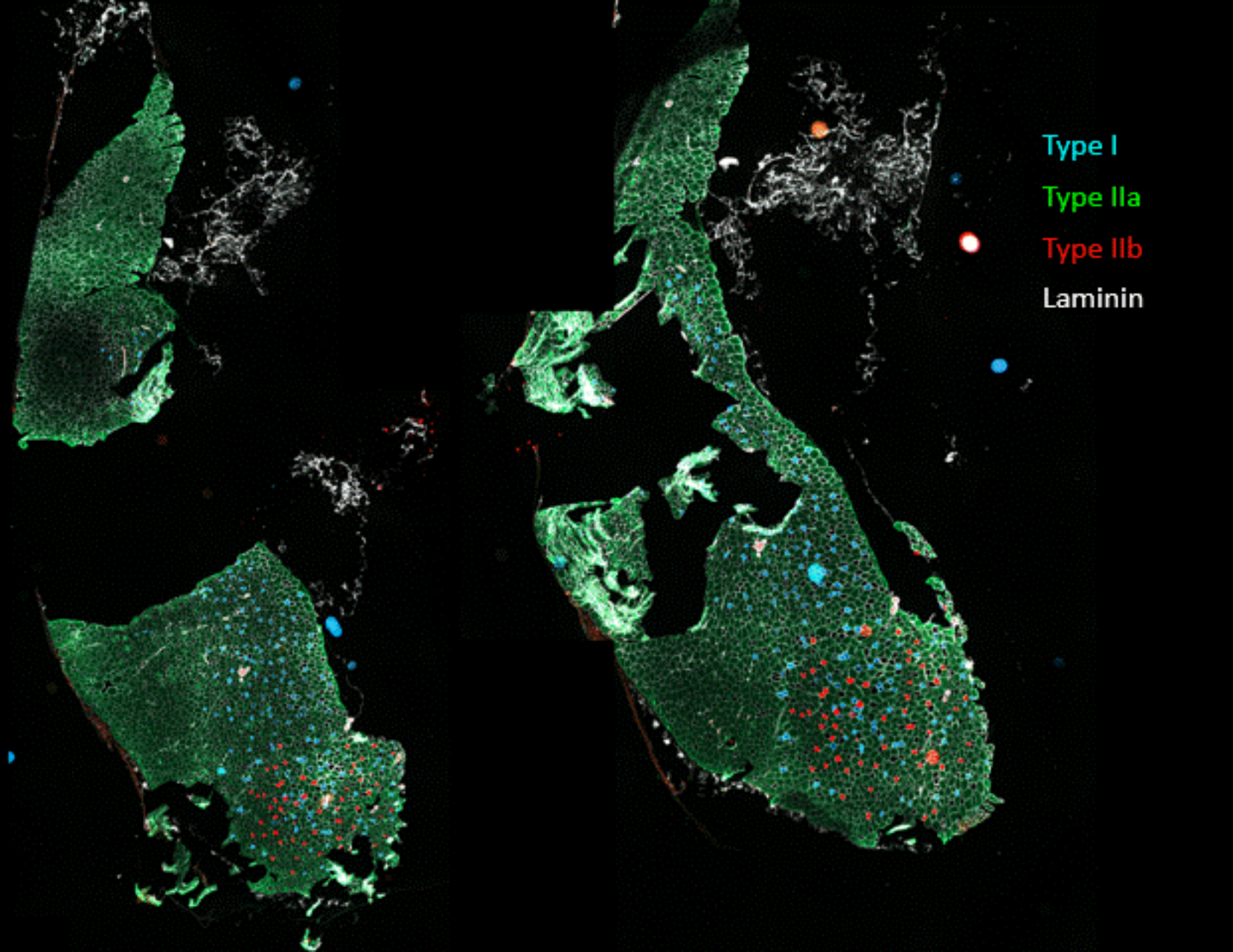
Install Anaconda

In an **Anaconda prompt:**

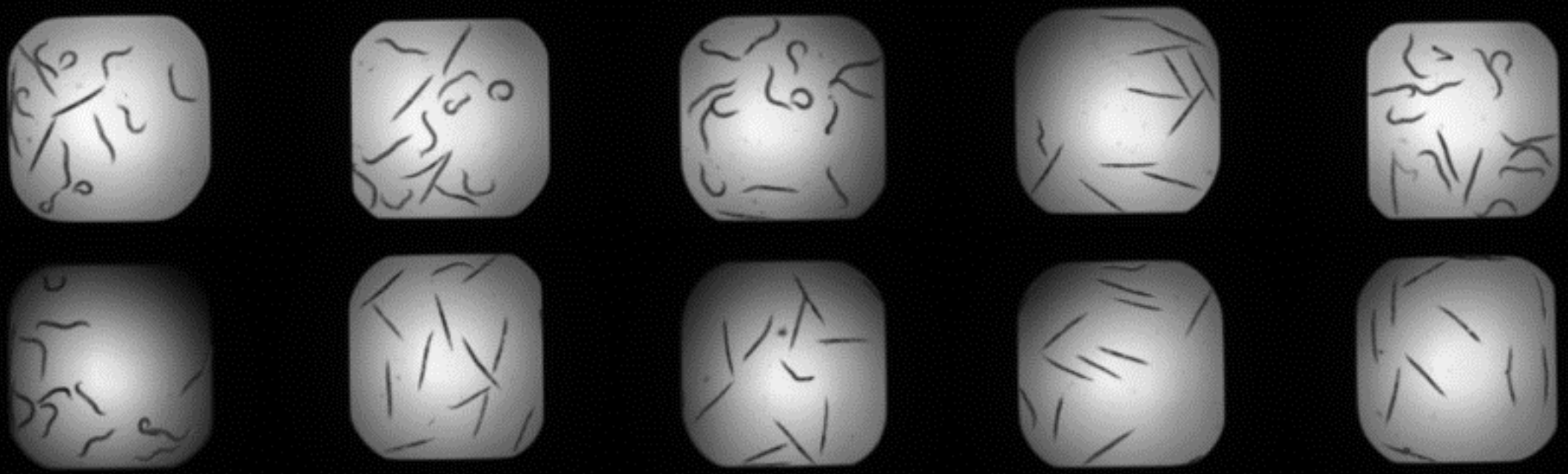
- Python –m venv CellposeEnv // create a virtual environment called CellposeEnv
- .\CellposeEnv\Scripts\activate // activate the virtual environment CellposeEnv
- pip install cellpose // install Cellpose

In Edit -> properties, indicate the path to the python executable in the virtual environment

- Define a small annotation and tune parameters to accurately segment muscle fibers with cellpose
- Create training image with both images
- Segment fibers in the training image
- Train an object classifier to identify the different muscle types
- Apply the classifier to each image and export measurements



BRIGHT FIELD IMAGES OF C.ELEGANS



Available at https://bbbc.broadinstitute.org/BBBC010

OMNIPOSE

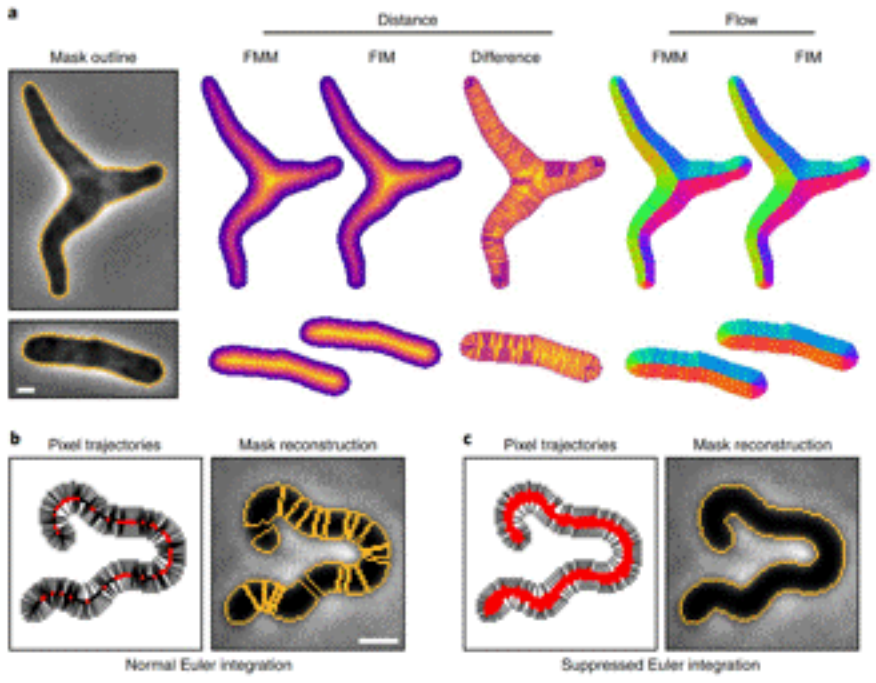


Fig. 2 | Core innovations of Omnipose. a, Comparison of distance field algorithms and corresponding flow fields on ground-truth masks. FMM produces ridges in the distance field resulting from pixelation on the cell mask boundary. Our smooth FIM algorithm minimizes these features. The difference image (FIM − FMM) highlights artifacts in the FMM method. Flow fields are calculated as the normalized gradient of the distance field. Boundary pixelation affects the FMM flow field deep into the cell, regardless of cell size. **b**,**c**, Comparison of mask reconstruction algorithms on a smooth flow field. Boundary pixel trajectories and resulting mask outlines from standard Euler integration (**b**). Trajectories and mask outlines under suppressed Euler integration (**c**). Red dots indicate the final positions of all cell pixels, not only the boundary pixels for which trajectories are displayed. Bacteria displayed are *E. coli* CS703-1 (**a**) and *H. pylori* (**b**,**c**) both treated with aztreonam. Scale bars, 1 μm. Images are representative of 1,299 *E. coli* and 701 *H. pylori* cells in the total ground-truth dataset, respectively.

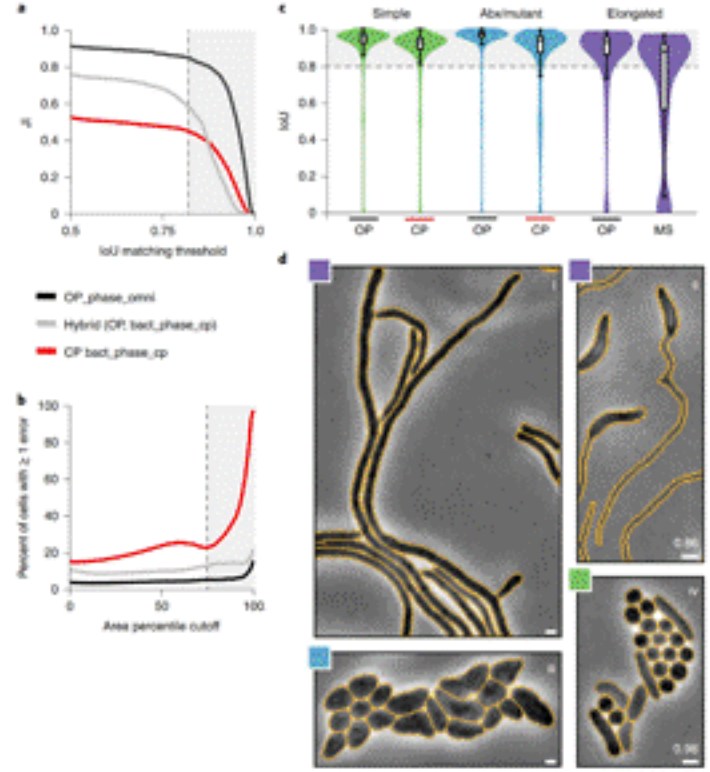


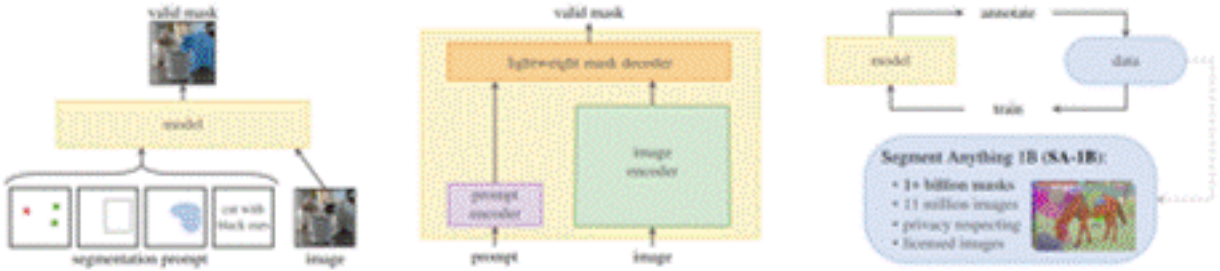
Fig. 3 | Omnipose substantially outperforms Cellpose on elongated cells. a, Overall performance of Omnipose (OP) (bact_phase_omni) and Cellpose (CP) (bact_phase_cp) measured by IJI. The hybrid method (gray) uses the original center-seeking flow output of bact_phase_cp and the mask reconstruction of Omnipose. Gray box represents IoU ≥ 0.8. $n = 19{,}570$ cells in the test set. **b**, Quantification of segmentation performance by cell size. The percent of cells with at least one segmentation error is computed for cells in each area percentile group from 1 to 100. Gray box represents the top quartile. **c**, Omnipose IoU distribution on the bact_phase dataset compared to the next highest performing algorithm in each of three cell categories (simple, $n = 12{,}869$; Abx/mutant, $n = 6{,}138$; and elongated, $n = 531$). Boxes centered on medians from Q1 to Q3, whiskers from Q1 − 1.5 IQR to Q3 + 1.5 IQR. **d**, Example micrographs and Omnipose segmentation. Mean matched IoU values shown. Bacteria displayed are *Streptomyces pristinaespiralis* (i), *Caulobacter crescentus* grown in HBGG medium (ii), *Shigella flexneri* treated with A22 (iii) and a mix of *Pseudomonas aeruginosa*, *Staphylococcus aureus*, *V. cholerae*, and *Bacillus subtilis* (iv). HBGG, Hutner base–imidazole-buffered glucose–glutamate. Scale bars, 1 μm.

Cutler *et al.,* Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation. *Nature Methods*, 2022

FOUNDATION MODEL FOR INSTANCE SEGMENTATION

Segment Anything

Alexander Kirillov[1,2,4] Eric Mintun[2] Nikhila Ravi[1,2] Hanzi Mao[2] Chloe Rolland[3] Laura Gustafson[3]
Tete Xiao[3] Spencer Whitehead Alexander C. Berg Wan-Yen Lo Piotr Dollár[4] Ross Girshick[4]
[1]project lead [2]joint first author [3]equal contribution [4]directional lead
Meta AI Research, FAIR



(a) **Task**: promptable segmentation (b) **Model**: Segment Anything Model (**SAM**) (c) **Data**: data engine (top) & dataset (bottom)

Figure 1: We aim to build a foundation model for segmentation by introducing three interconnected components: a prompt-able segmentation *task*, a segmentation *model* (SAM) that powers data annotation and enables zero-shot transfer to a range of tasks via prompt engineering, and a *data* engine for collecting SA-1B, our dataset of over 1 billion masks.

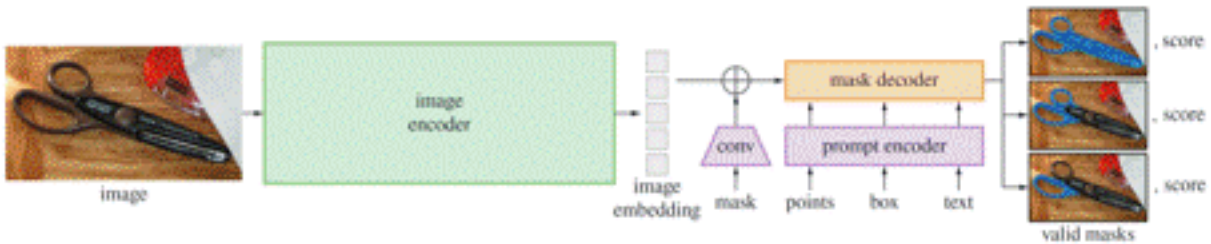


Figure 4: Segment Anything Model (SAM) overview. A heavyweight image encoder outputs an image embedding that can then be efficiently queried by a variety of input prompts to produce object masks at amortized real-time speed. For ambiguous prompts corresponding to more than one object, SAM can output multiple valid masks and associated confidence scores.

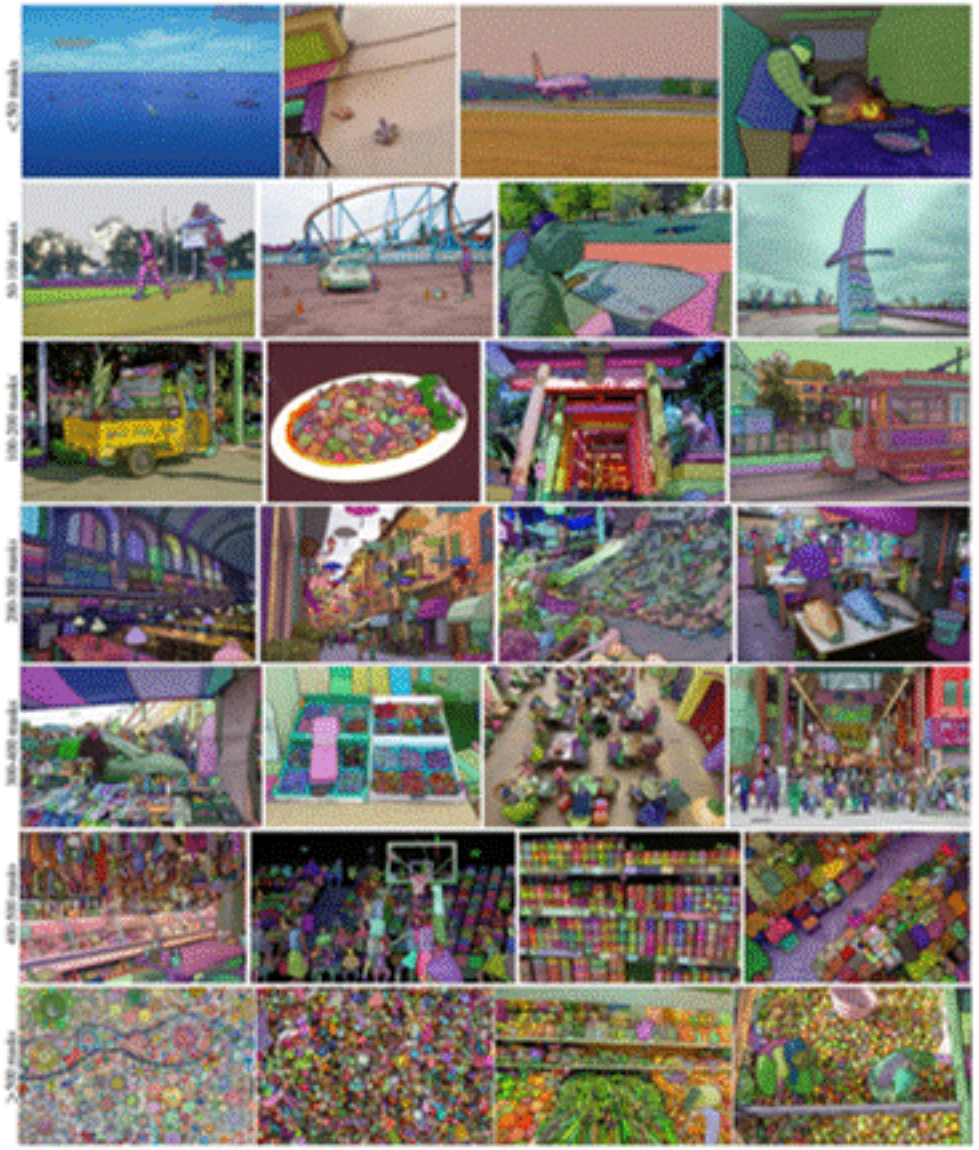


Figure 2: Example images with overlaid masks from our newly introduced dataset, **SA-1B**. SA-1B contains 11M diverse, high-resolution, licensed, and privacy protecting images and 1.1B high-quality segmentation masks. These masks were annotated *fully automatically* by SAM, and as we verify by human ratings and numerous experiments, are of high quality and diversity. We group images by number of masks per image for visualization (there are ~100 masks per image on average).

FOUNDATION MODEL FOR INSTANCE SEGMENTATION

Segment Anything

Alexander Kirillov[1,2,4] Eric Mintun[2] Nikhila Ravi[1,2] Hanzi Mao[2] Chloe Rolland[3] Laura Gustafson[3]
Tete Xiao[3] Spencer Whitehead Alexander C. Berg Wan-Yen Lo Piotr Dollár[4] Ross Girshick[4]
[1]project lead [2]joint first author [3]equal contribution [4]directional lead
Meta AI Research, FAIR

(a) **Task**: promptable segmentation (b) **Model**: Segment Anything Model (**SAM**) (c) **Data**: data engine (top) & dataset (bottom)

Figure 1: We aim to build a foundation model for segmentation by introducing three interconnected components: a promptable segmentation *task*, a segmentation *model* (SAM) that powers data annotation and enables zero-shot transfer to a range of tasks via prompt engineering, and a *data* engine for collecting SA-1B, our dataset of over 1 billion masks.

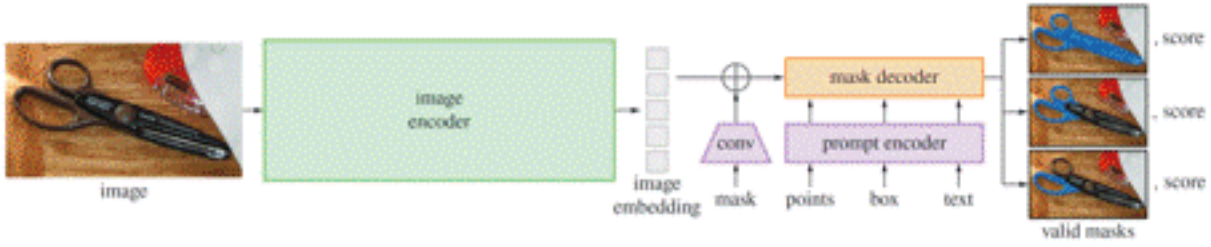
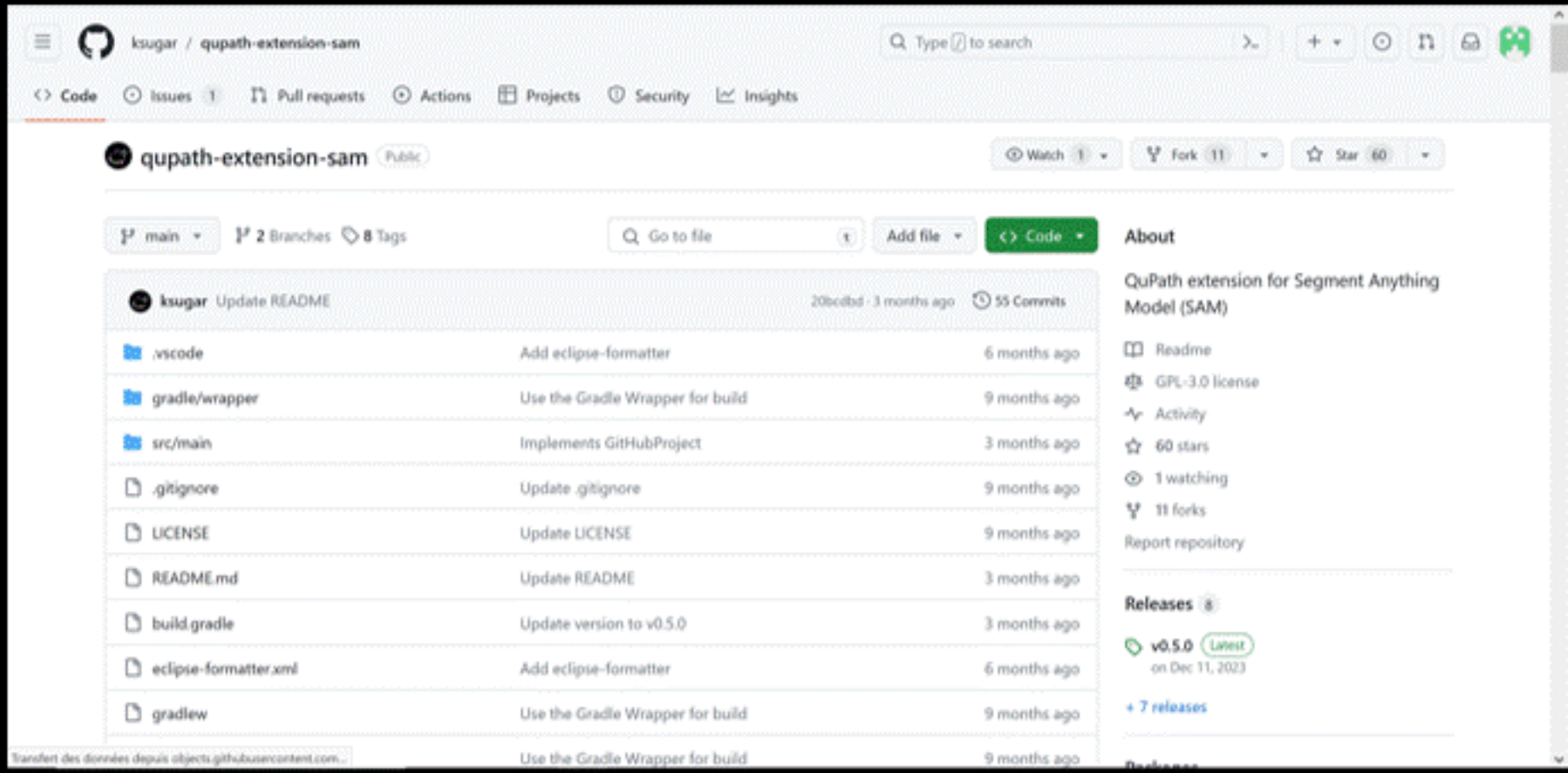


Figure 4: Segment Anything Model (SAM) overview. A heavyweight image encoder outputs an image embedding that can then be efficiently queried by a variety of input prompts to produce object masks at amortized real-time speed. For ambiguous prompts corresponding to more than one object, SAM can output multiple valid masks and associated confidence scores.

Figure 3: Each column shows 3 valid masks generated by SAM from a single ambiguous point prompt (green circle).

Download the latest **SAM extension** for QuPath and **drag it into QuPath**

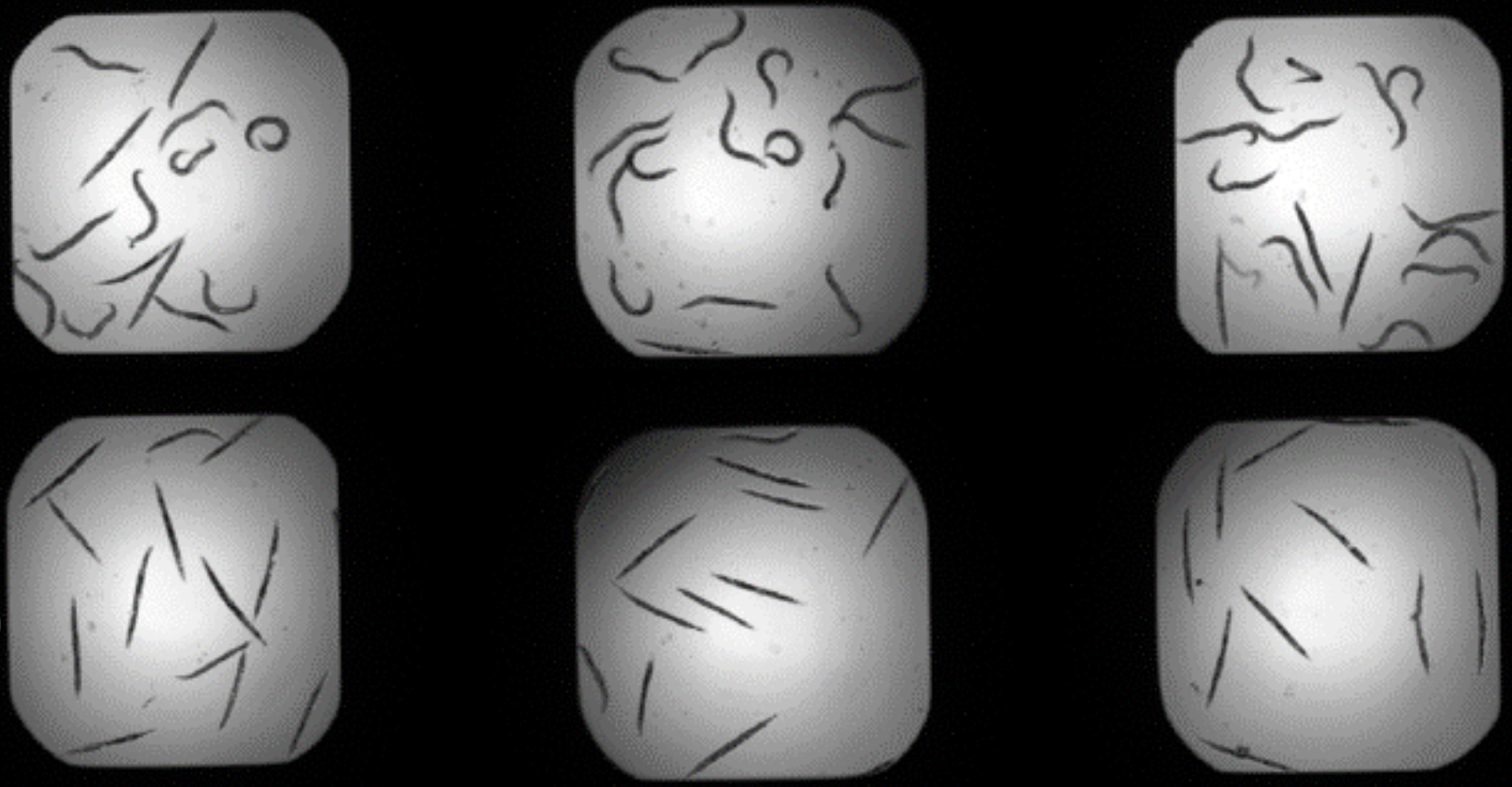


Install git (https://git-scm.com/download/win)

In an **Anaconda prompt**:

- conda create –n samapi –y python=3.10 // create a virtual environment called samapi
- conda activate samapi // activate the virtual environment samapi
- python -m pip install git+https://github.com/ksugar/samapi.git // install SAM API
- uvicorn samapi.main:app --workers 2 // process SAM API

- Create a training image with 6 images
- Segment images with omnipose model
- Use SAM to annotate each image
- Fine-tune an omnipose model
- Segment other images
- Train an object classifier to identify dead and alive worms



1 mm

- P. Bankhead *et al.* **QuPath: Open source software for digital pathology image analysis**. *Scientific Reports* (2017). https://doi.org/10.1038/s41598-017-17204-5
- U. Schmidt *et al.* **Cell Detection with Star-convex Polygons.** *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (2018). https://arxiv.org/abs/1806.03535
- https://github.com/qupath/qupath-extension-stardist
- N.F. Greenwald *et al.* **Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning.** *Nature Biotechnology (2021).* https://doi.org/10.1038/s41587-021-01094-0
- T. Pécot *et al.* **A deep learning segmentation strategy that minimizes the amount of manually annotated images.** *F1000 Research (2022)* https://doi.org/10.12688/f1000research.52026.2
- T. Pécot *et al.* **Deep learning tools and modeling to estimate the temporal expression of cell cycle proteins from 2D still images.** *PLOS Computational Biology (2022)* https://doi.org/10.1371/journal.pcbi.1009949
- C. Stringer *et al.* **Cellpose: a generalist algorithm for cellular segmentation.** *Nature methods (2021)* https://doi.org/10.25378/janelia.13270466
- K.J. Cutler *et al.* **Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation**. *Nature Methods (2022)* https://doi.org/10.1038/s41592-022-01639-4
- https://github.com/BIOP/qupath-extension-cellpose
- A. Kirillov *et al.* **Segment anything.** *arXiv:2304.02643 (2023)* https://doi.org/10.48550/arXiv.2304.02643
- K. Sugawara **Training deep learning models for cell image segmentation with sparse annotations.** *bioRxiv (2023)* https://doi.org/10.1101/2023.06.13.544786