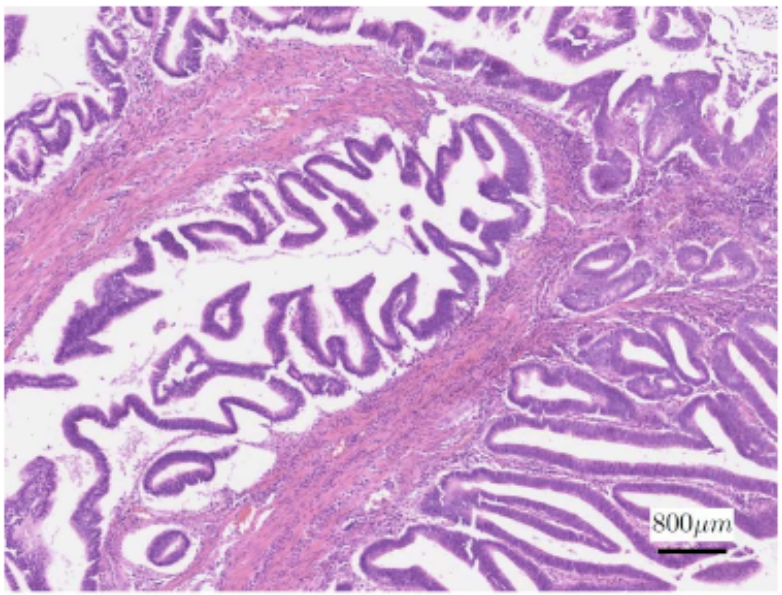
# Locality sensitive deep learning for detection and classification of nuclei

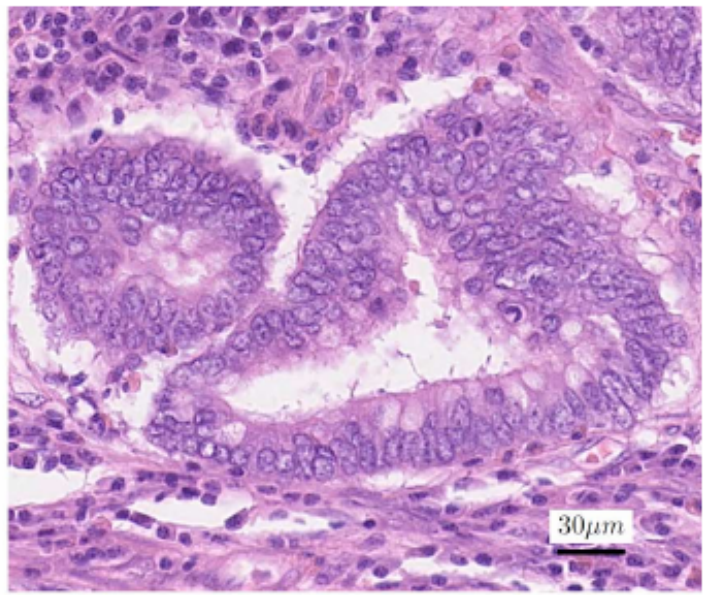
Sirinukunwattana et al., IEEE TMI 2016

Jan Kybic

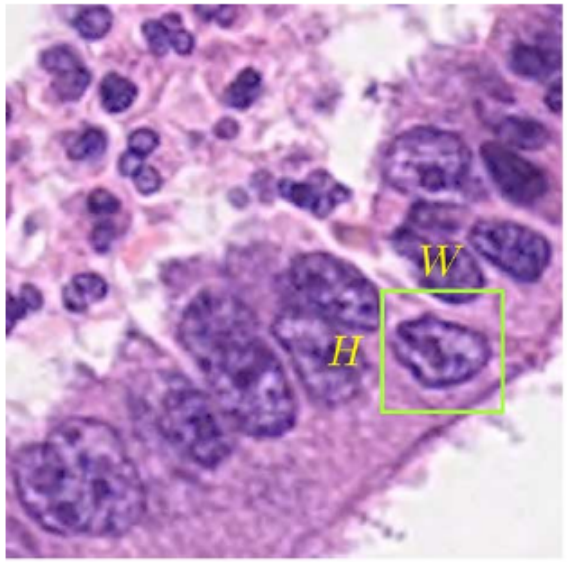
2020

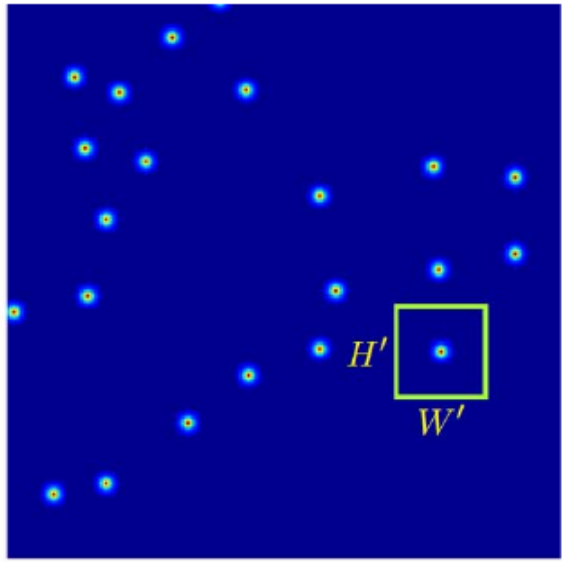
# Input images



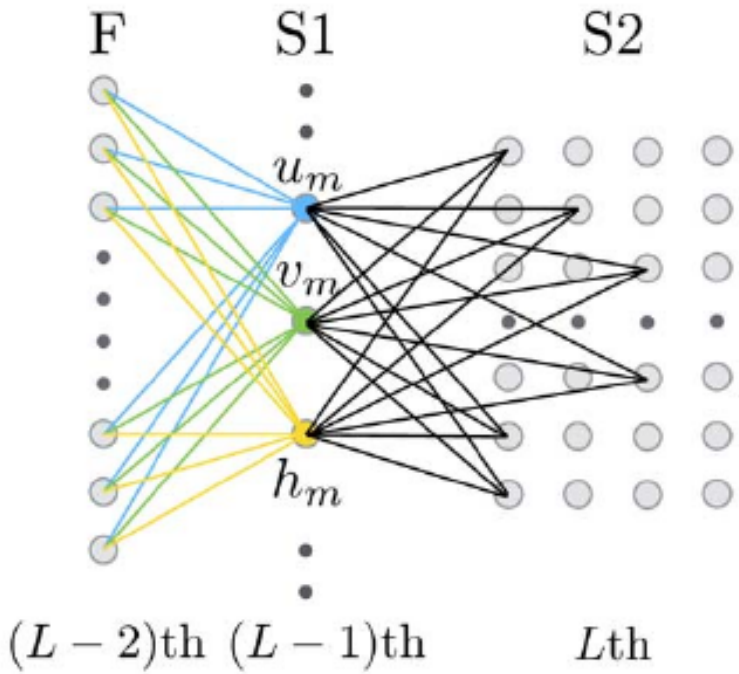
# Target density




$$y_j = \begin{cases} \dfrac{1}{1+\frac{(\|\mathbf{z}_j - \mathbf{z}_m^0\|_2^2)}{2}} & \text{if } \begin{array}{c} \forall m \neq m', \|\mathbf{z}_j - \mathbf{z}_m^0\|_2 \\ \leq \|\mathbf{z}_j - \mathbf{z}_{m'}^0\|_2 \leq d, \end{array} \\ 0 & \text{otherwise,} \end{cases}$$

# Output mapping




$$\hat{y}_j = g(\mathbf{z}_j; \hat{\mathbf{z}}_1^0, \ldots, \hat{\mathbf{z}}_M^0, h_1, \ldots, h_M)$$

$$= \begin{cases} \left( \frac{1}{1+\frac{(\|\mathbf{z}_j - \hat{\mathbf{z}}_m^0\|_2^2)}{2}} \right) h_m & \text{if } \begin{array}{l} \forall m \neq m', \|\mathbf{z}_j - \hat{\mathbf{z}}_m^0\|_2 \\ \leq \|\mathbf{z}_j - \hat{\mathbf{z}}_{m'}^0\|_2 \leq d, \end{array} \\ 0 & \text{otherwise,} \end{cases}$$

# Output mapping (2)

$$u_m = (H' - 1) \cdot sigm(\mathbf{W}_{L-1,u_m} \cdot \mathbf{x}_{L-2} + b_{u_m}) + 1,$$
$$v_m = (W' - 1) \cdot sigm(\mathbf{W}_{L-1,v_m} \cdot \mathbf{x}_{L-2} + b_{v_m}) + 1,$$
$$h_m = sigm(\mathbf{W}_{L-1,h_m} \cdot \mathbf{x}_{L-2} + b_{h_m}),$$

## Loss function

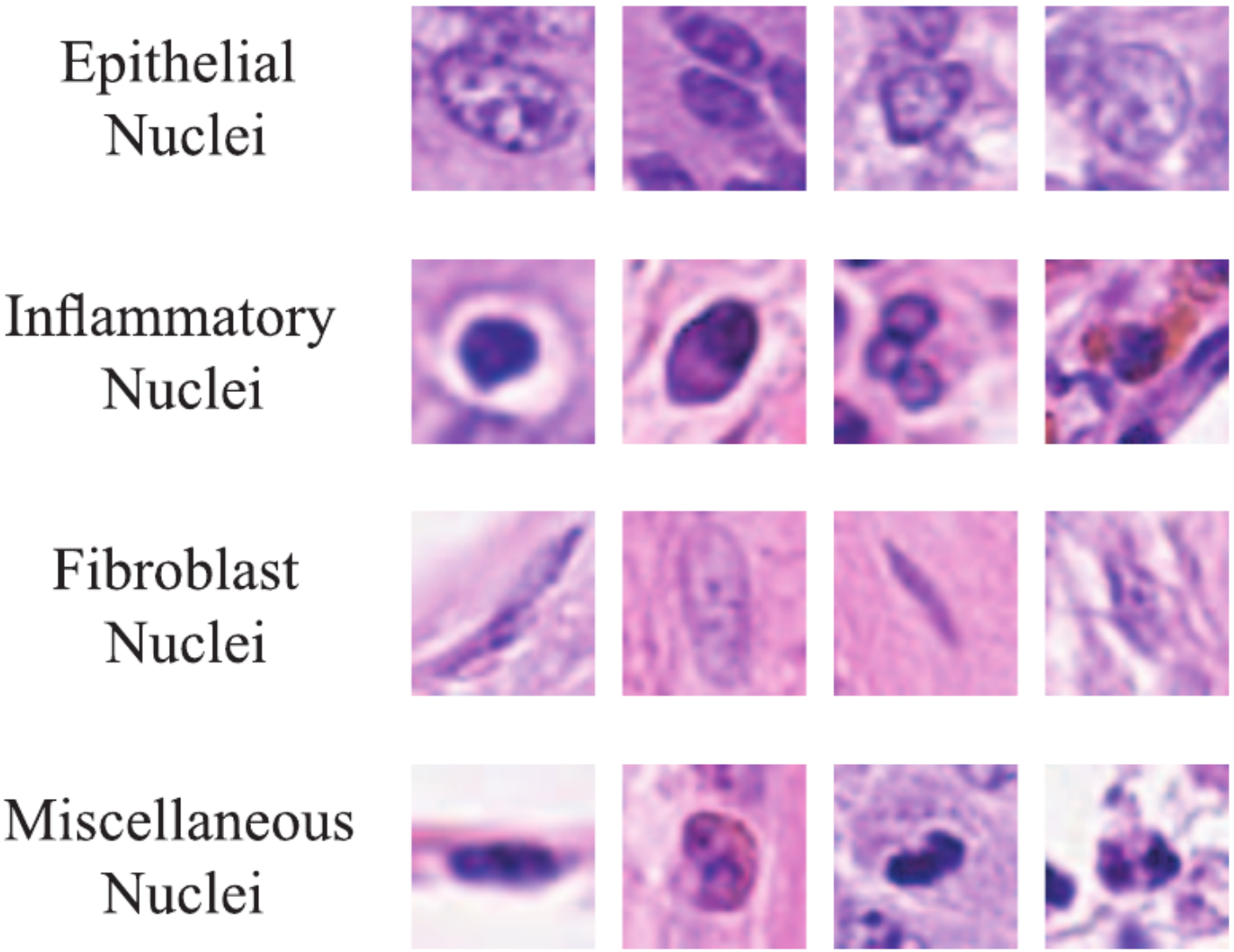
$$\ell(\mathbf{y}, \hat{\mathbf{y}}) = \sum_j (y_j + \epsilon) H(y_j, \hat{y}_j), \qquad (10)$$

where $H(y_j, \hat{y}_j)$ is the cross-entropy loss [29]–[31] defined by

$$H(y_j, \hat{y}_j) = -\Big[ y_j \log(\hat{y}_j) - (1 - y_j) \log(1 - \hat{y}_j) \Big], \qquad (11)$$

## Sliding windows

# Nucleus types



# Nucleus classification

- Softmax

$$p_j(\hat{\mathbf{y}}(\mathbf{x})) = \frac{\exp(\hat{y}_j(\mathbf{x}))}{\sum_k \exp(\hat{y}_k(\mathbf{x}))}.$$

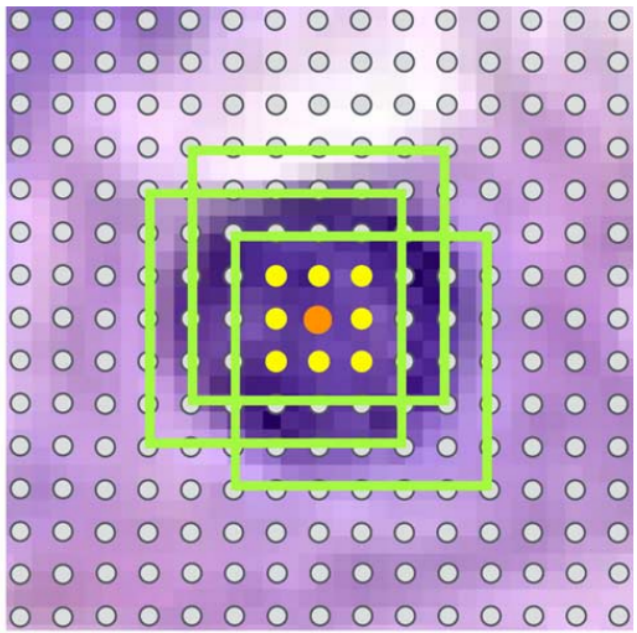
- Logarithmic loss

$$\ell(c, \hat{\mathbf{y}}(\mathbf{x})) = -\log(p_c(\hat{\mathbf{y}}(\mathbf{x})))$$

- Final classification

$$c = \arg\max_j \hat{y}_j(\mathbf{x}),$$

# Neighboring Ensemble Predictor



is centered. We now define a set of neighboring patches of $\mathbf{x}$ as

$$\mathcal{B}(\mathbf{x}) = \left\{ \mathbf{x}^{(i)} \in \mathcal{X}_{\mathrm{I}} : \|\mathbf{z}(\mathbf{x}^{(i)}) - \mathbf{z}(\mathbf{x})\|_2 \leq d_\beta \right\}, \qquad (16)$$

which contains all patches centered in the ball of radius $d_\beta$ centered at $\mathbf{z}(\mathbf{x})$.

Predicted class $\hat{c}$ for an input patch $\mathbf{x}$ with corresponding network output $\hat{\mathbf{y}}$ is given by

$$\hat{c} = \arg\max_j \sum_{\mathbf{x}^{(i)} \in \mathcal{B}(\mathbf{x})} w(\mathbf{z}(\mathbf{x}^{(i)})) p_j(\hat{\mathbf{y}}(\mathbf{x}^{(i)})), \qquad (17)$$

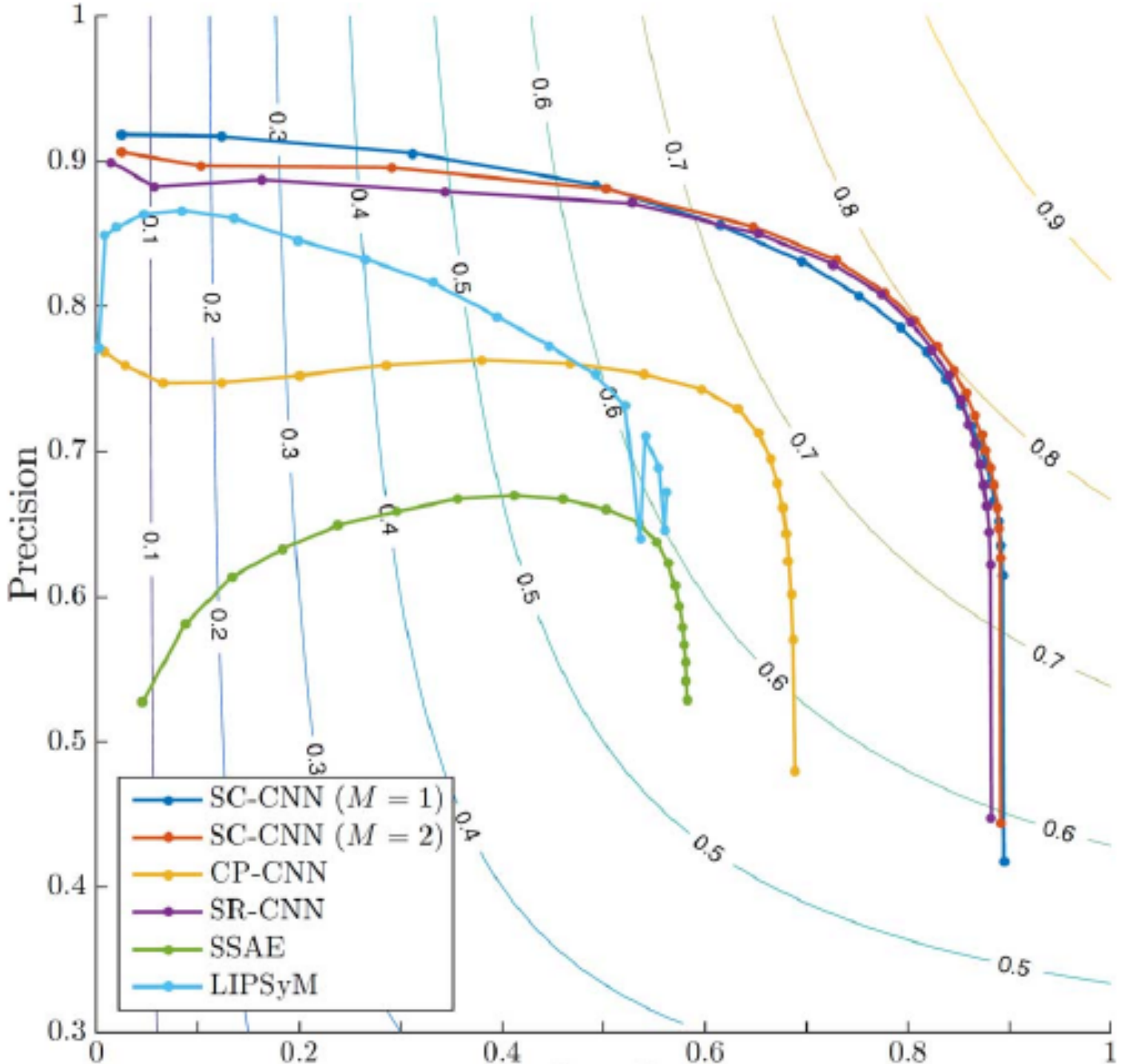
# CNN parameters

ARCHITECTURES OF THE SPATIALLY CONSTRAINED CNN (SC-CNN) FOR NUCLEUS DETECTION AND SOFTMAX CNN FOR NUCLEUS CLASSIFICATION. THE NETWORKS CONSIST OF INPUT (I), CONVOLUTION (C), MAX-POOLING (M), FULLY-CONNECTED (F), PARAMETER ESTIMATION (S1), AND SPATIALLY CONSTRAINED (S2) LAYERS

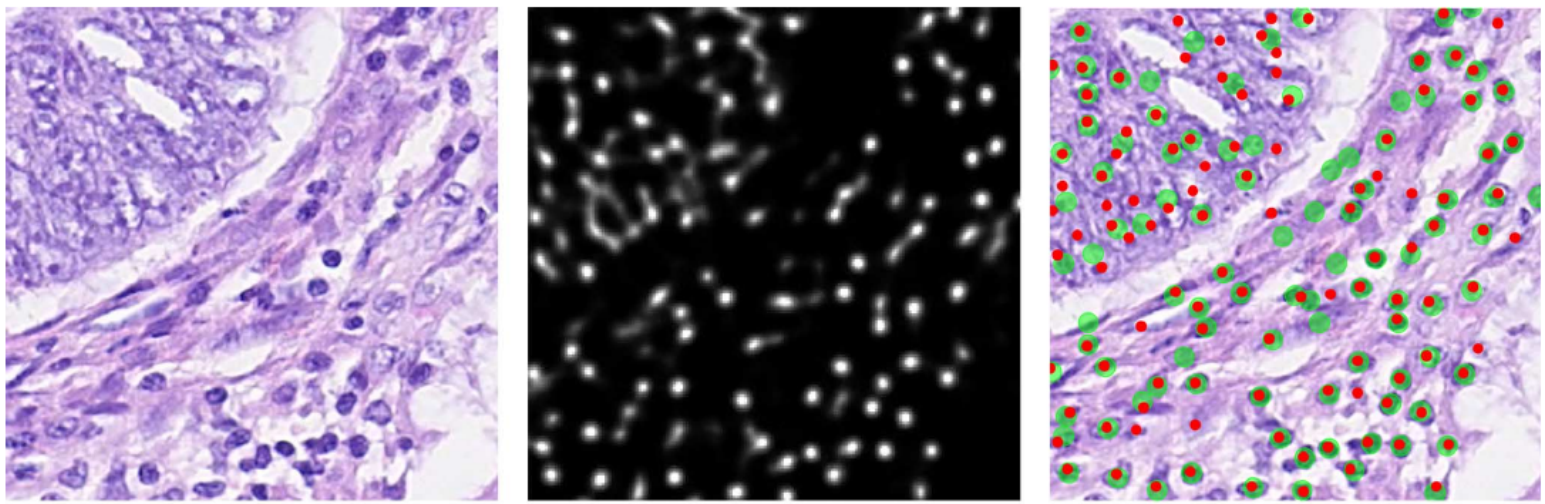
| Layer | SC-CNN for detection | | | softmax CNN for classification | | |
|---|---|---|---|---|---|---|
| | Type | Filter Dimensions | Input/Output Dimensions | Type | Filter Dimensions | Input/Output Dimensions |
| 0 | I | | 27 × 27 × 1 | I | | 27 × 27 × 1 |
| 1 | C | 4 × 4 × 1 × 36 | 24 × 24 × 36 | C | 4 × 4 × 1 × 36 | 24 × 24 × 36 |
| 2 | M | 2 × 2 | 12 × 12 × 36 | M | 2 × 2 | 12 × 12 × 36 |
| 3 | C | 3 × 3 × 36 × 48 | 10 × 10 × 48 | C | 3 × 3 × 36 × 48 | 10 × 10 × 48 |
| 4 | M | 2 × 2 | 5 × 5 × 48 | M | 2 × 2 | 5 × 5 × 48 |
| 5 | F | 5 × 5 × 48 × 512 | 1 × 512 | F | 5 × 5 × 48 × 512 | 1 × 512 |
| 6 | F | 1 × 1 × 512 × 512 | 1 × 512 | F | 1 × 1 × 512 × 512 | 1 × 512 |
| 7 | S1 | 1 × 1 × 512 × 3 | 1 × 3 | F | 1 × 1 × 512 × 4 | 1 × 4 |
| 8 | S2 | | 11 × 11 | | | |

- **Input:** hematoxylin (color deconvolution) for detection, RGB for classification
- ReLU, dropout, augmentation (90°rotation, flip, multiplicative color noise in HSV, shift), random initialization

# Nucleus detection precision/recall

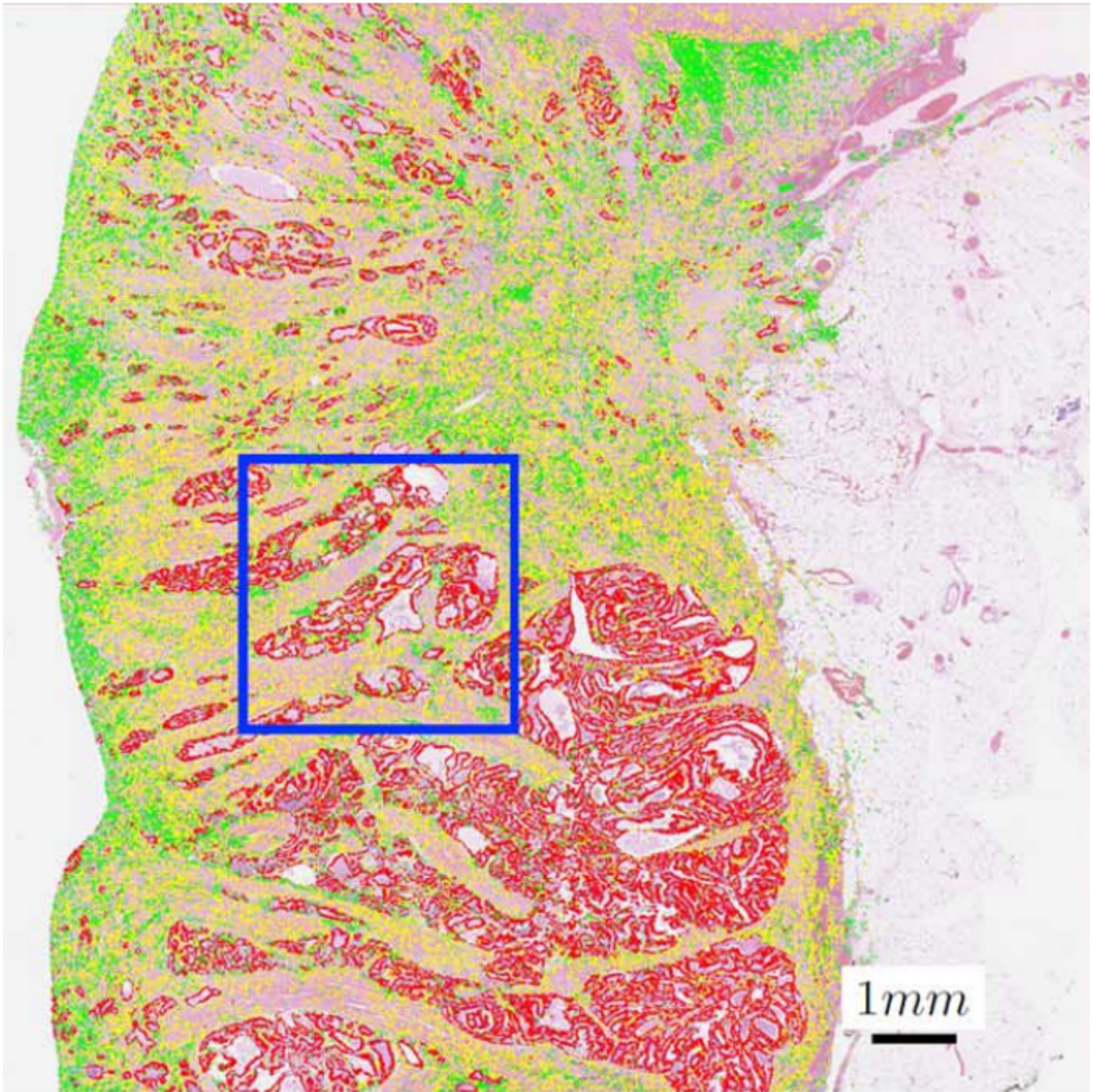


# Nucleus detection example

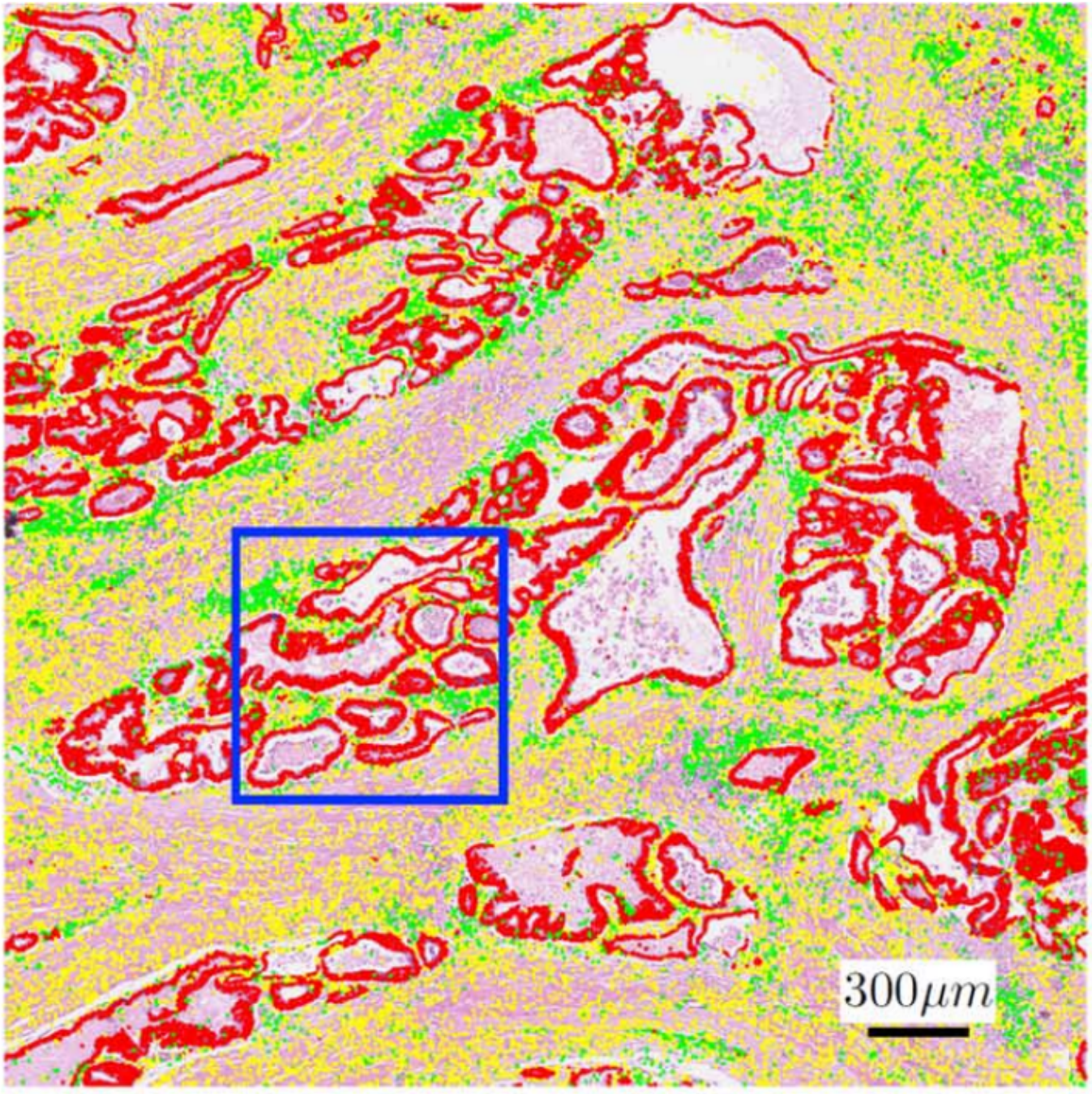


(a) (b) (c)

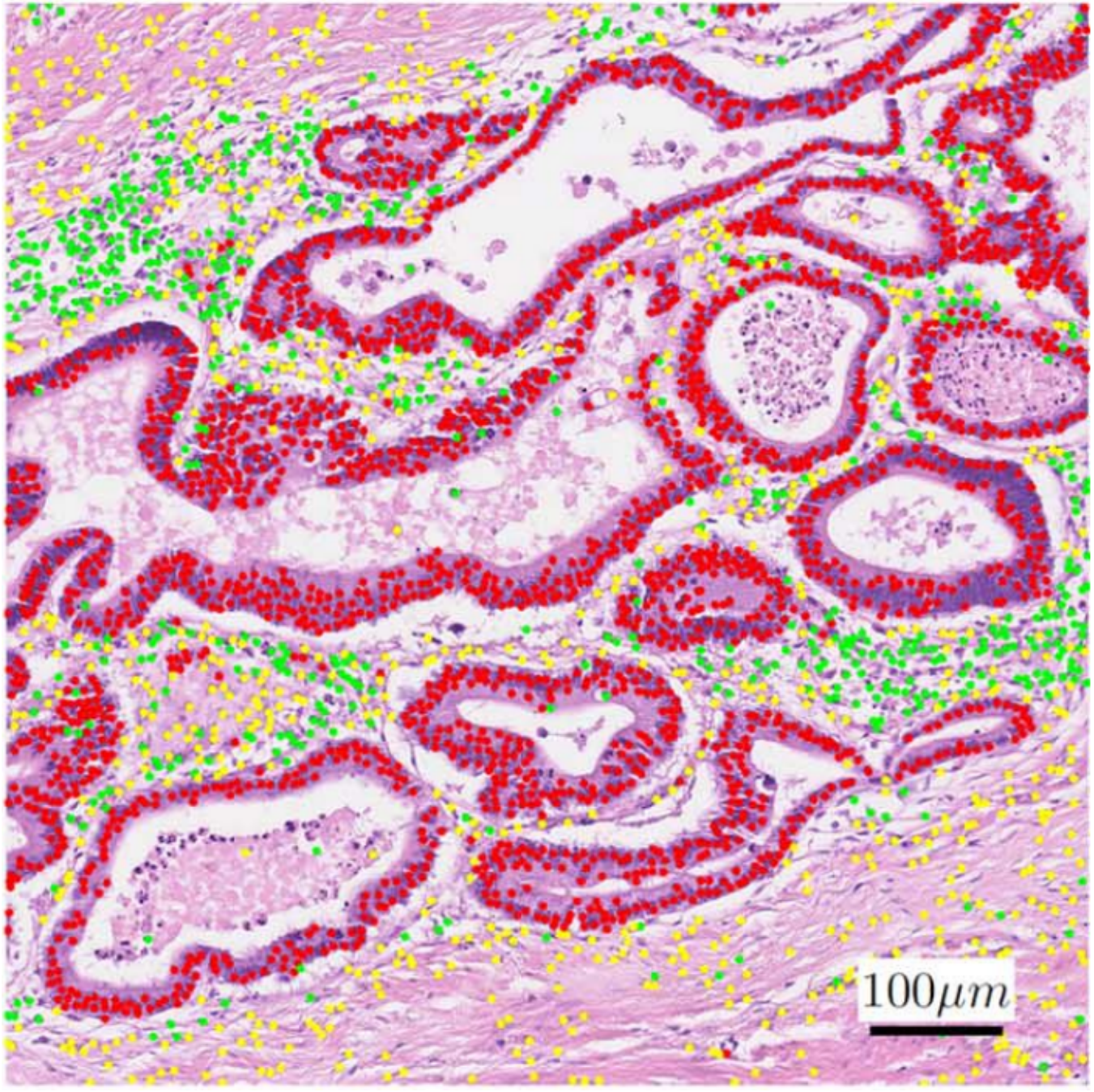
# Classification and detection example



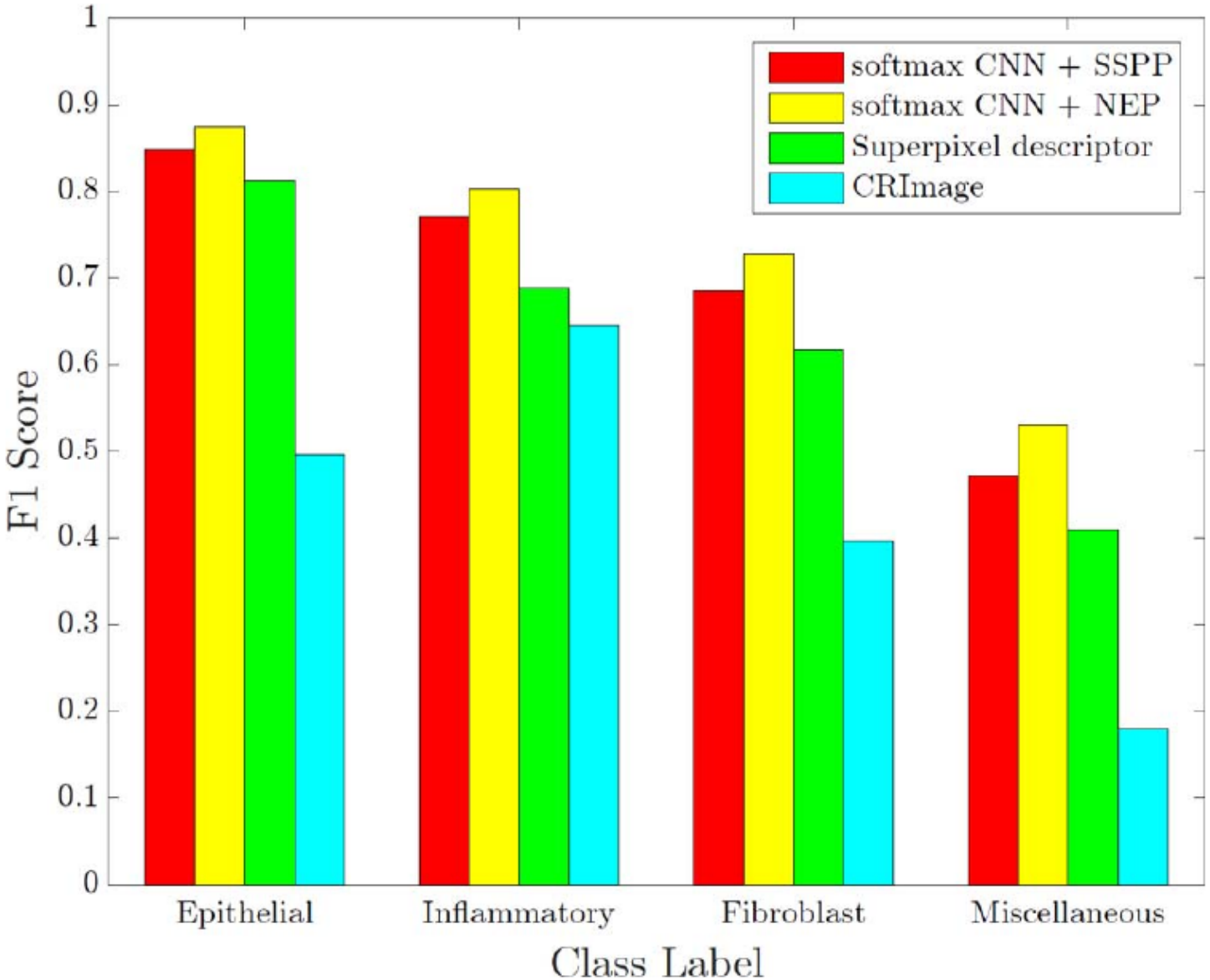
# Classification and detection example (2)



# Classification and detection example (3)



# Nucleus classification F1 score



# Combined detection and classification results

| Method | | Weighted |
|---|---|---|
| Detection | Classification | Average F1 score |
| SC-CNN ($M = 1$) | softmax CNN + SSPP | 0.664 |
| | softmax CNN + NEP | 0.688 |
| SC-CNN ($M = 2$) | softmax CNN + SSPP | 0.670 |
| | softmax CNN + NEP | **0.692** |
| SR-CNN [27] | softmax CNN + SSPP | 0.662 |
| | softmax CNN + NEP | 0.683 |

# Timing

- $1000 \times 1000$ pixels takes 47.6 s on a CPU
- $60000 \times 50000$pixels $=$ 750 tiles $=$ 50 min/slide
- likely much faster on a GPU