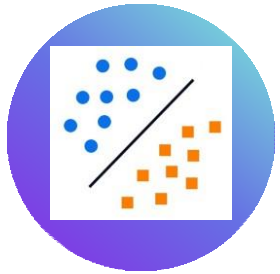
Cell nuclei segmentation: support vector machine

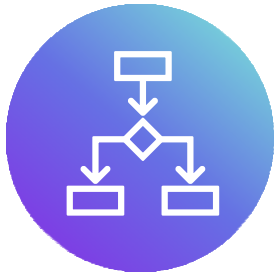
Final presentation by:
Michelle Emmert, Juan Hamdan, Laura Sanchis, und Gloria Timm




Our results



Our algorithm



Evaluation

Our data

Our Dataset

28 images of nuclei :

N2DH-GOWT1

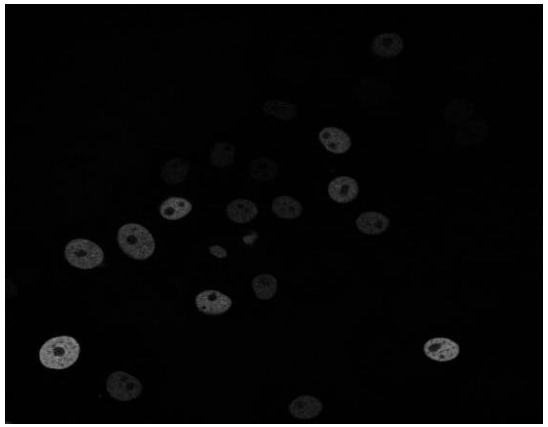
GFP transfected GOWT1 mouse embryonic stem cells

N2DL-HeLa

Histone 2B (H2B)-GFP expressing HeLa cells

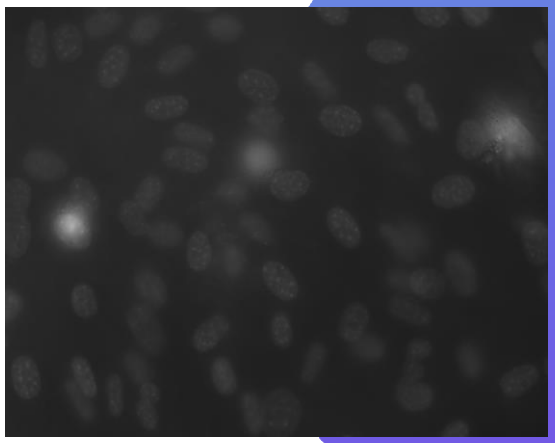
NIH3T3

mouse embyonic fibroblast – CD tagged (EGFP)



N2DH-GOWT1

N2DL-HeLa



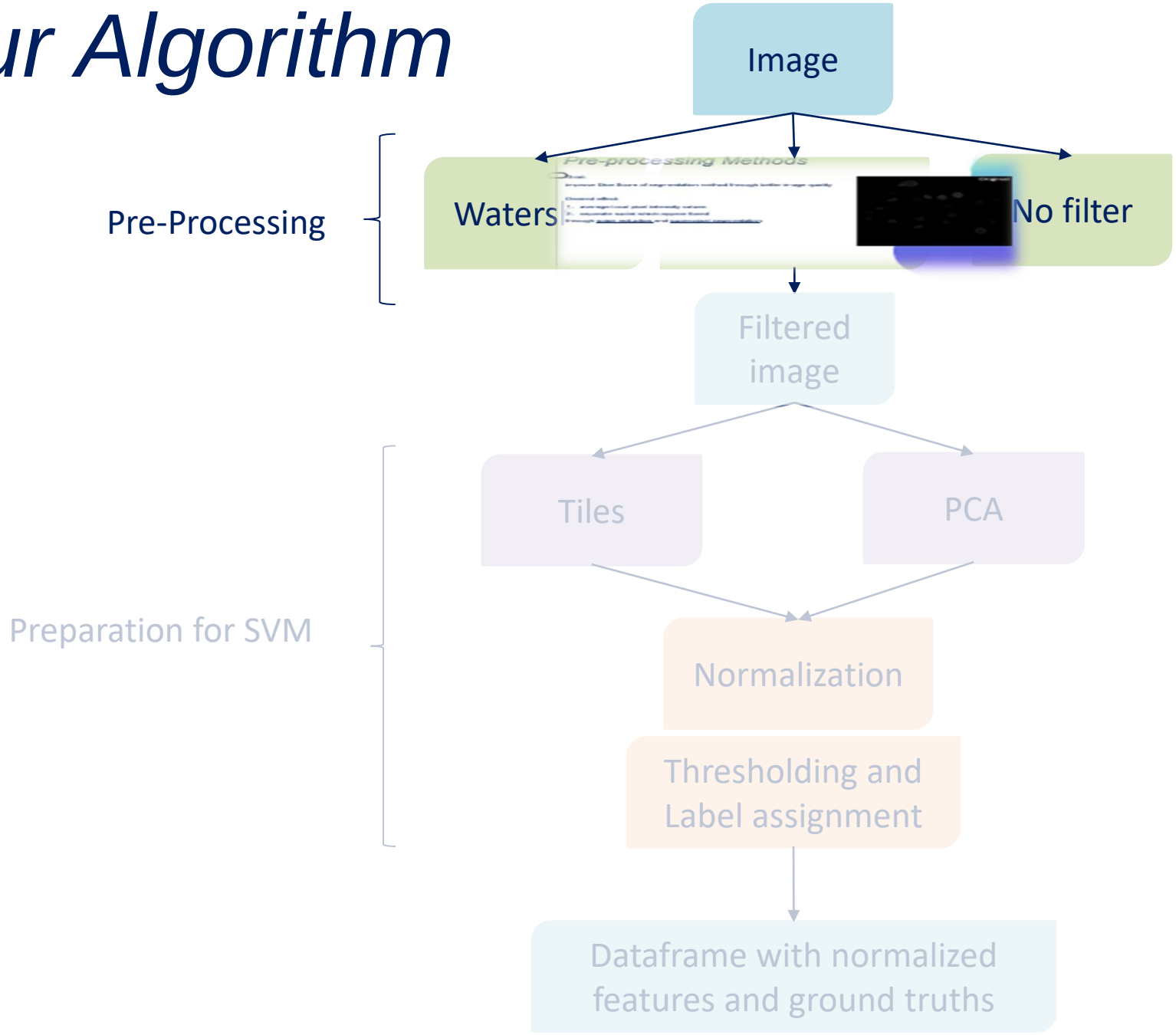
NIH3T3

(1) Osuna, E. et al. 2007. Large-Scale Automated Analysis of Location Patterns in Randomly Tagged 3T3Cells

(2) Maska, M. et al. 2014. A benchmark for comparisonof cell tracking algorithms

Synthetic images

Our Algorithm



Pre-processing Methods

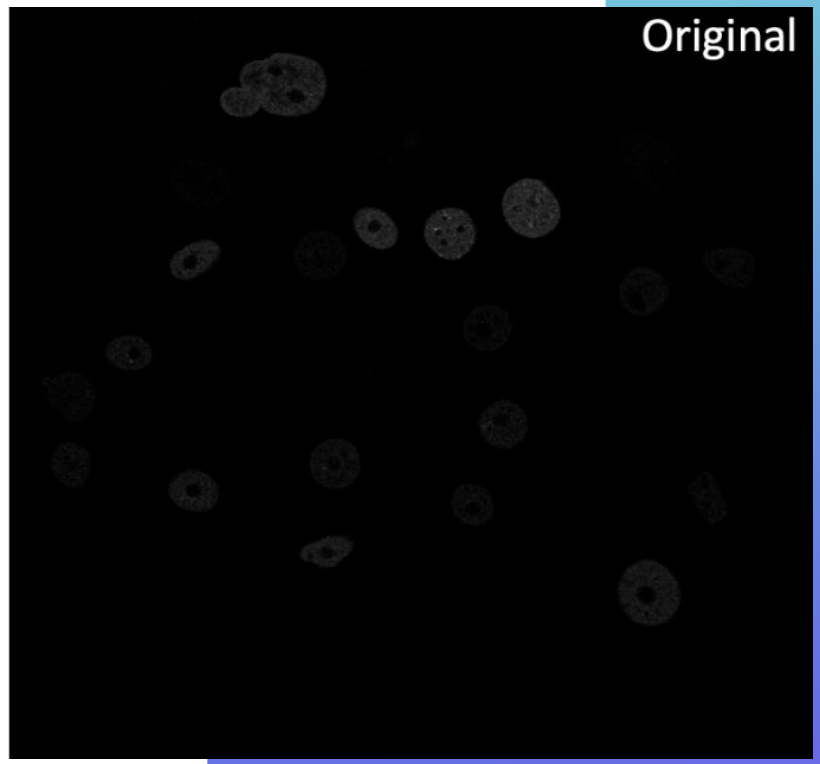
Goal:

improve Dice Score of segmentation method through better image quality

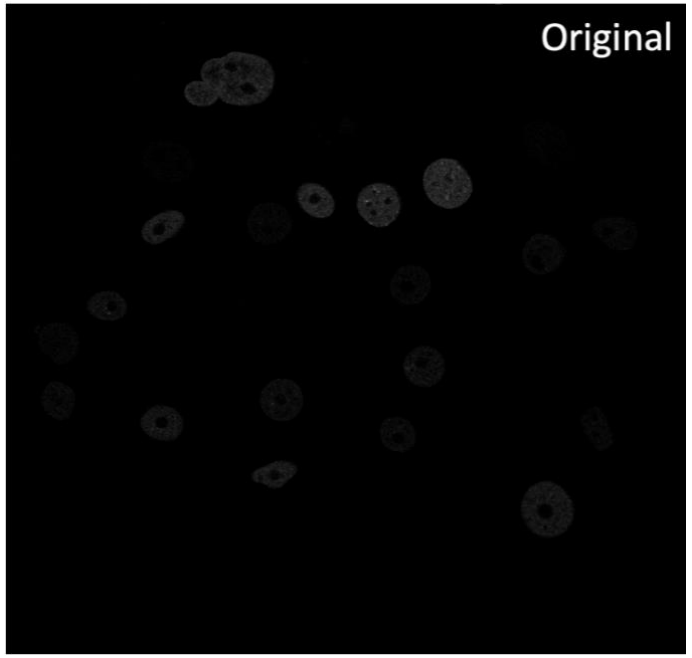
Desired effect:

1. average local pixel intensity values
2. separate nuclei which appear fused

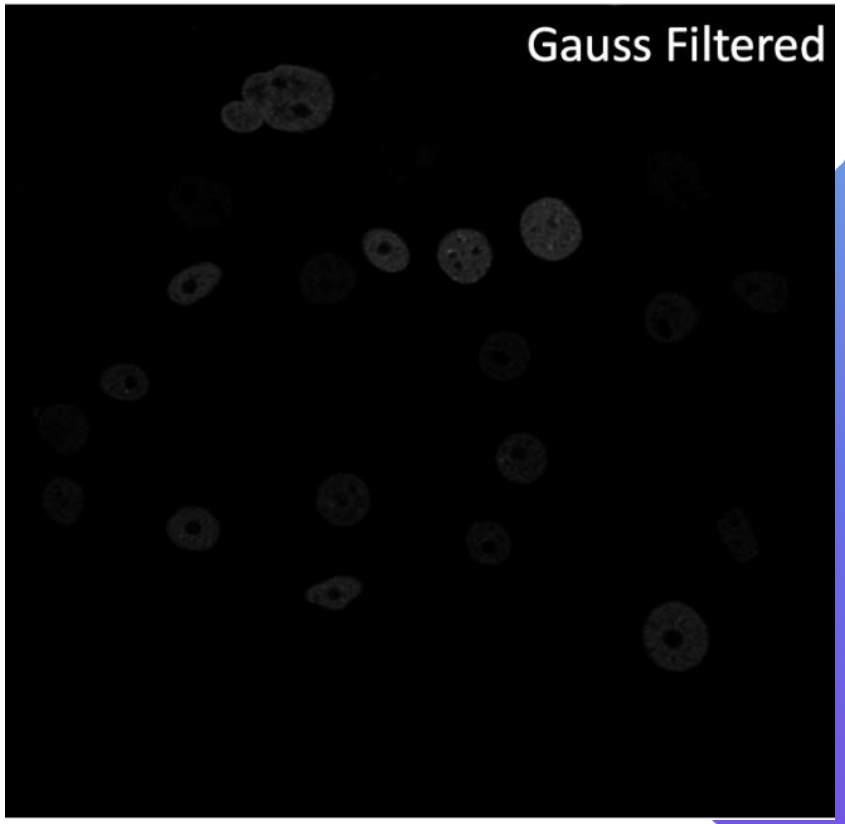
through noise reduction and super-pixel segmentation



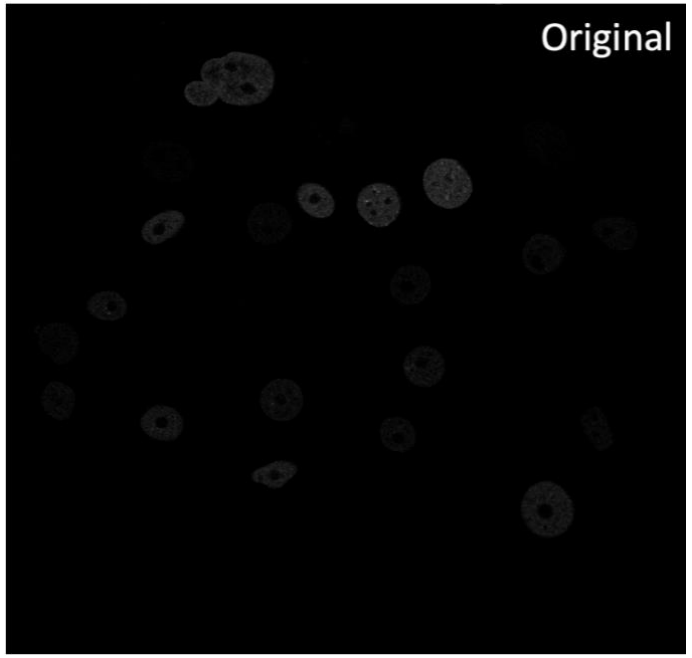
Pre-processing Methods



Gaussian Filtering

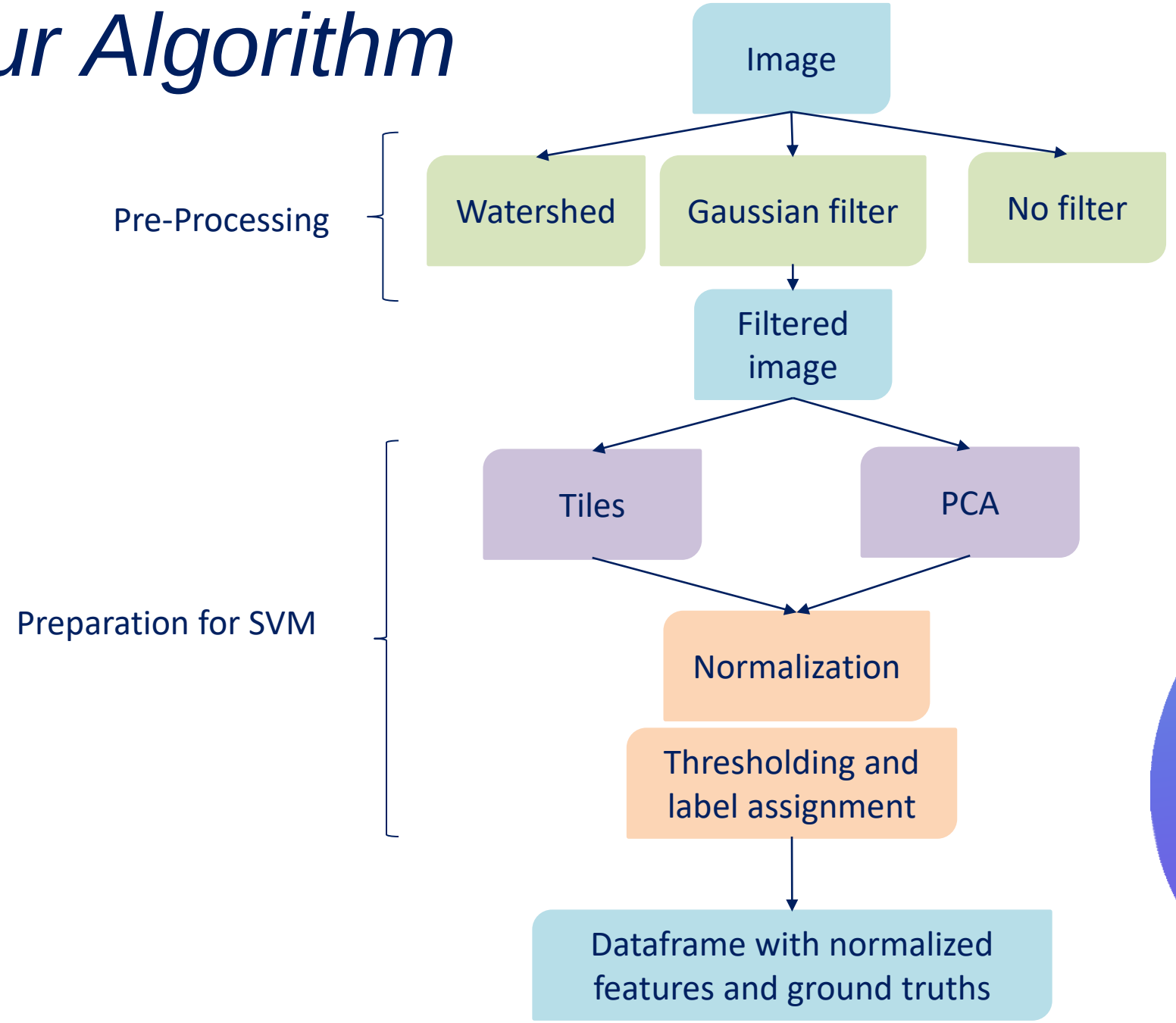


Pre-processing Methods



Watershed Filtering

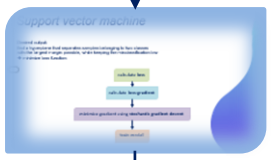
Our Algorithm



Our Algorithm

Dataframe with normalized features and ground truths

Segmentation



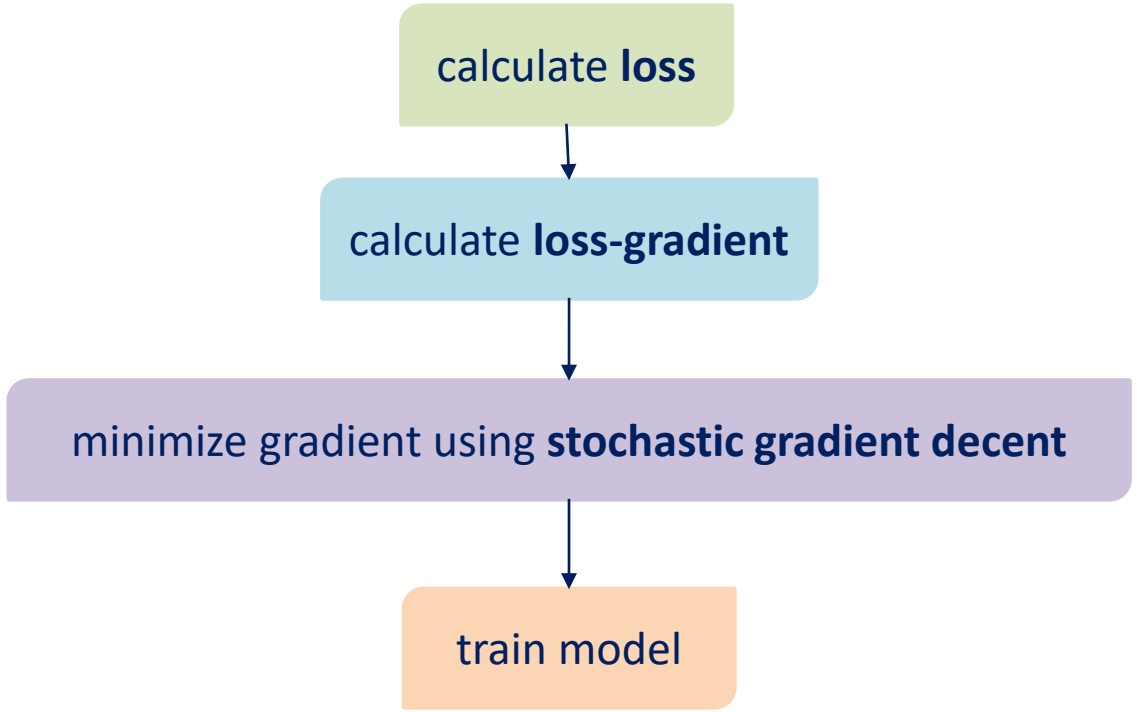
Segmented image

Evaluation

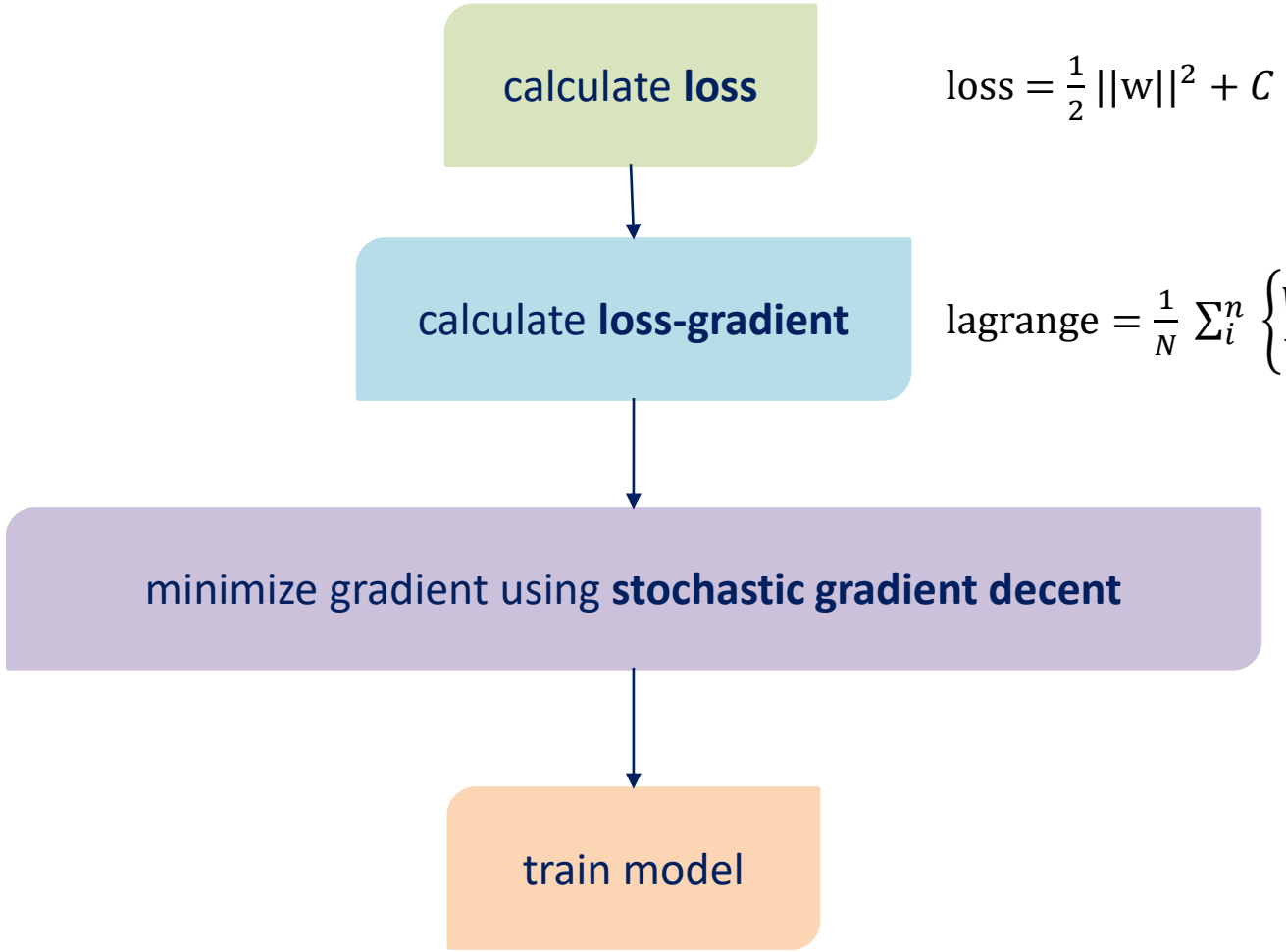
Dice score

Support vector machine

Desired output:
find a hyperplane that separates samples belonging to two classes
with the largest margin possible, while keeping the misclassification low
→ minimize loss function



Support vector machine



$$\text{loss} = \frac{1}{2}||\text{w}||^2 + C\left[\frac{1}{N}\sum_i^n max\,(0, 1 - y_i * (w * x_i + b))\right]$$

$$\text{lagrange} = \frac{1}{N}\sum_i^n \begin{cases} w & if\ max\,(0, 1 - y_i * (w * x_i)) = 0 \\ w - C\,y_i x_i & otherwise \end{cases}$$

Support vector machine

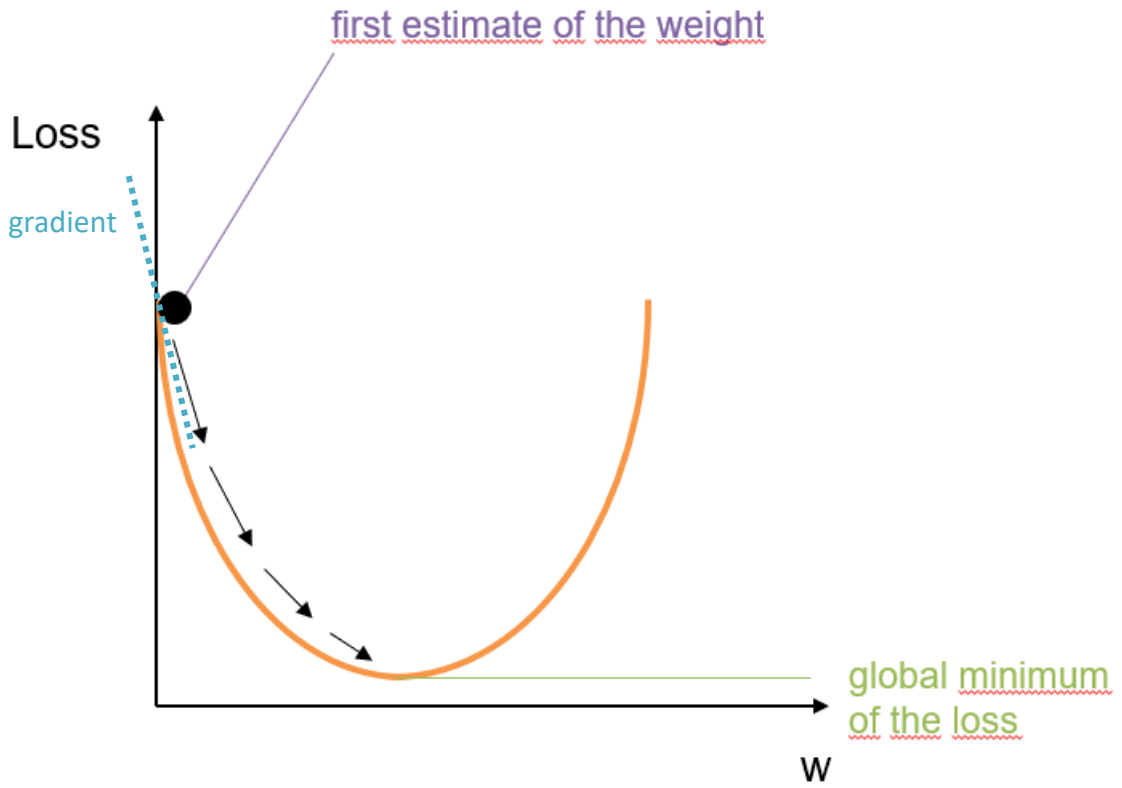
Code SVM

calculate **loss**

calculate **loss-gradient**

minimize gradient using **stochastic gradient decent**

train model



Results

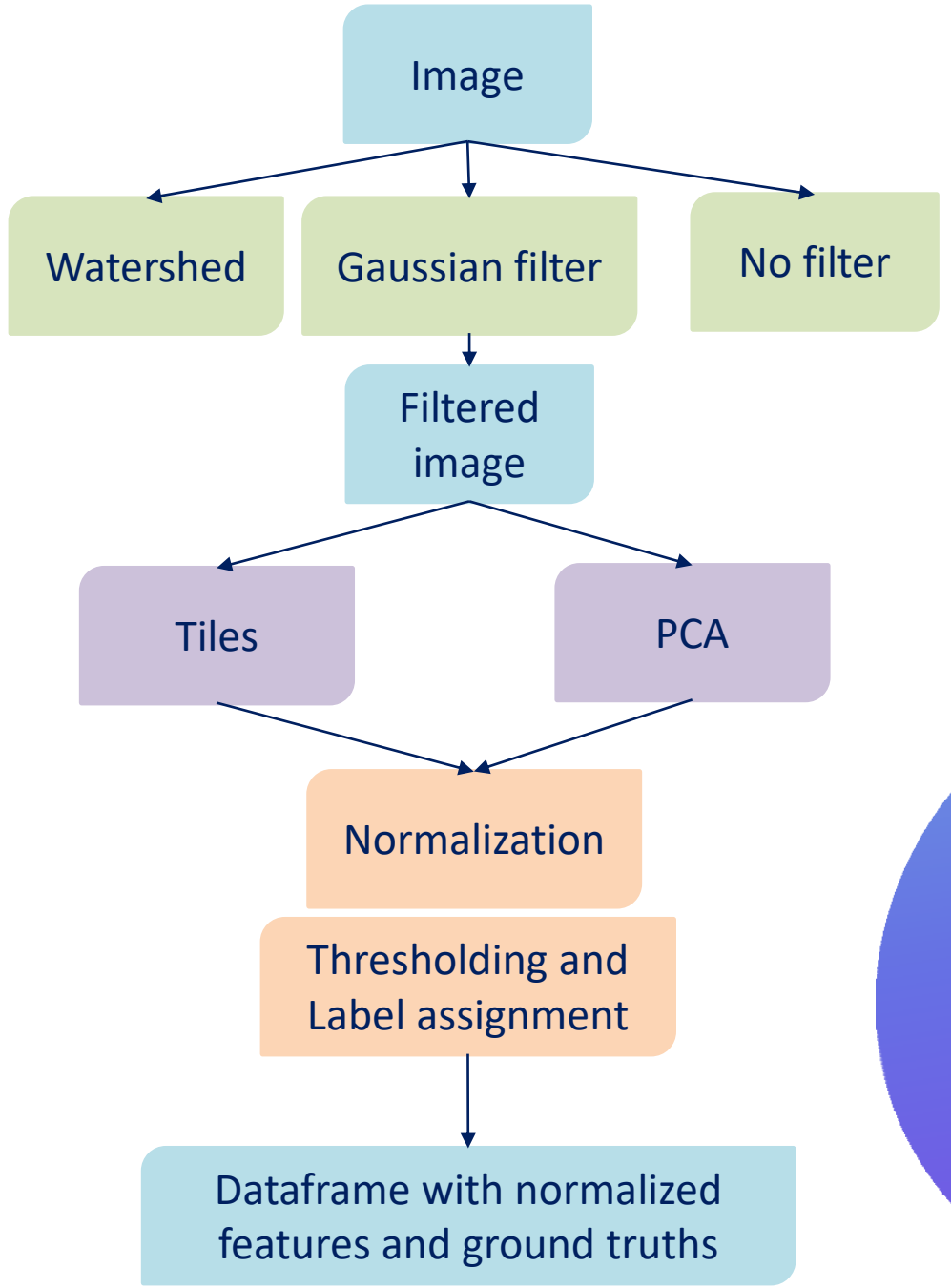
- Segmentiertes Bild + Original + Nuclei count?

Optimal setting for our algorithm

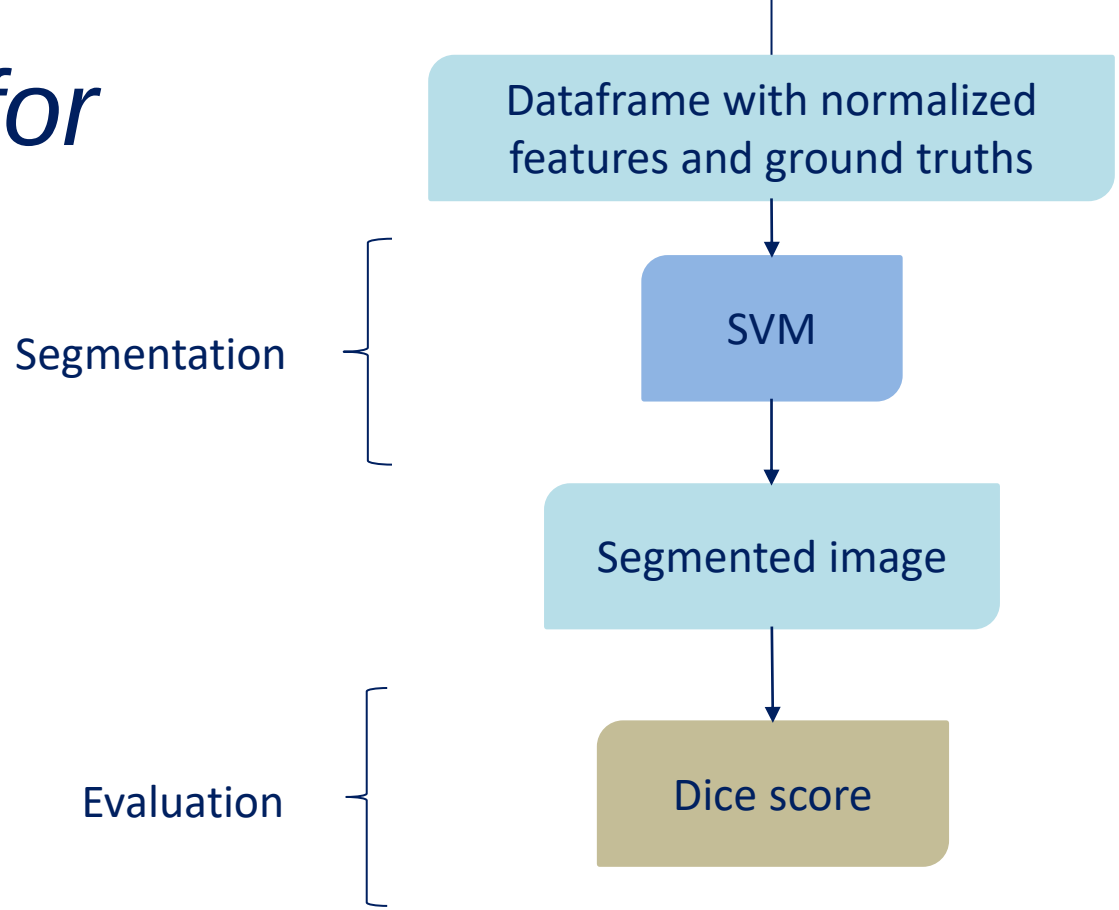
Pre-Processing

Das ausblenden was nicht gemacht wurde
Tiles: number angeben
Parameter filter angeben

Preparation for SVM



Optimal setting for our algorithm



Evaluation using the dice score

```python
def dice_score(pred, gt):
    """
    This function calculates the similiarity between two arrays.
    :param pred: an array of predicted labels
    :param gt: the ground truth of this array
    :return: a value between 0 and 1, describing the similiarity between those arrays. 1 is the dice score of similiar arrays.
    """
    dice = np.sum(pred[gt == pred]) * 2.0 / (np.sum(gt) + np.sum(pred))
    print(dice)
```

$$\text{Dice} = \frac{2*Intersection}{Union+Intersection} = F_1 = \frac{1}{\frac{1}{Prec}+\frac{1}{Recall}} = \frac{2\,TP}{2\,TP+FP+FN}$$

Evaluation of our optimal settings

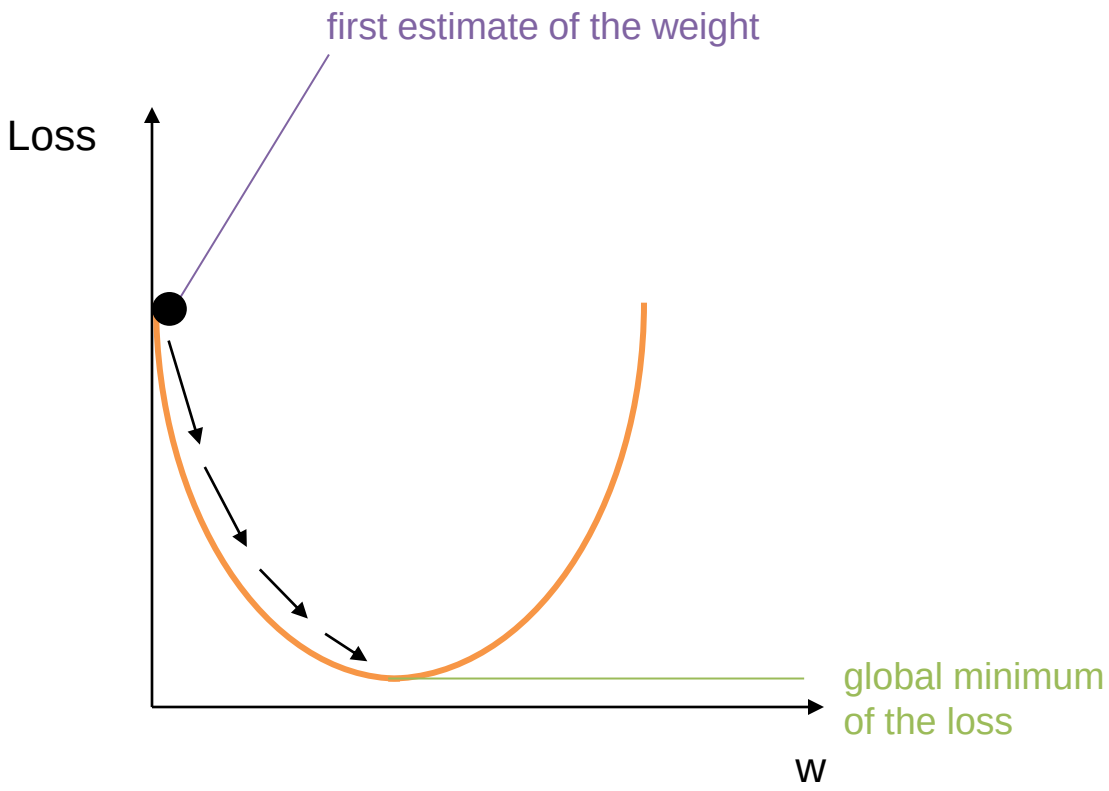
Evaluation of different settings

Vergleiche (z.B. in Boxplot Dice Scores vgl.):

- Mit Gauss vs. Ohne Gauss
- Mit Watershed vs. Ohne Watershed
- Watershed vs. Gauss
- Tiles vs. PCA

Summary/What we've learned

Thank you for listening!


first estimate of the weight
Loss
global minimum
of the loss
w