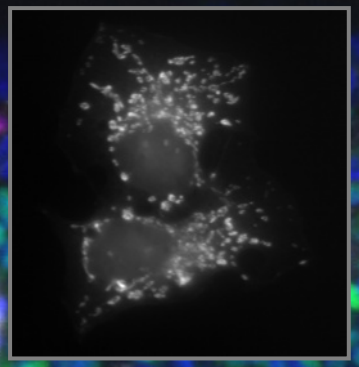
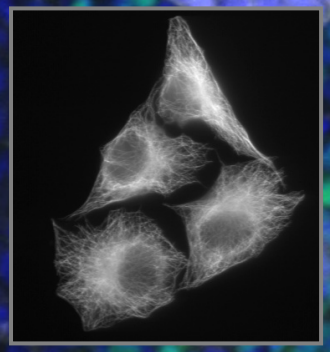
Datasets and slides: https://github.com/doktor-nick/intro-to-ml-for-imaging



Introduction to Machine Learning for Imaging

Dr Nick Hamilton
Institute Bio-Mathematician
Institute for Molecular Bioscience
Research Computing Centre
Queensland Cyber Infrastructure Foundation
n.hamilton@imb.uq.edu.au

www.imb.uq.edu.au



Images: Groups Little & Teasdale

Power corrupts.

PowerPoint corrupts absolutely.

- Edward Tufte

Outline

- Types of machine learning: supervised, unsupervised, reinforcement
- Support Vector Machine
- Classifying Images
- Image Segmentation: classifying pixels in an image
- Trainable Weka image segmentation in ImageJ/Fiji

What is machine learning?

- Machine learning is a subset of Artificial Intelligence that uses statistical techniques to “learn” to computationally solve a problem without explicit programming

- Types of Machine Learning Algorithms
  - Neural Networks, Support Vector Machines, Bayesian Networks, Genetic Algorithms, …
- Why now?
  - Deep learning : Neural Networks with many connected “layers”
  - Convolution Neural Networks – inspired by organization of visual cortex
  - GPUs make training the “deep” networks feasible
  - “Easy” to frameworks such as Tensorflow, Torch, Theano, Fiji / Trainable Weka …

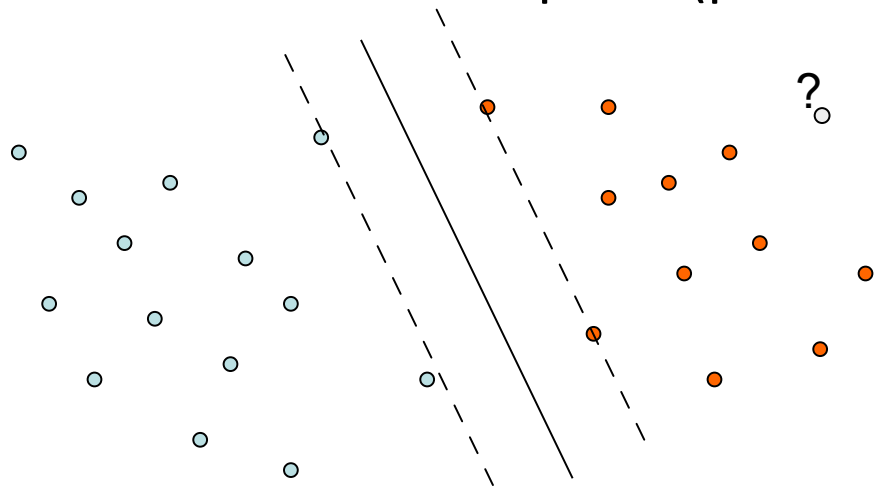
- Peak hype!
  - Machine learning is far from perfect
  - The feature that the ML is using to classify may in fact be an artifact
  - Is often a black box with little insight into how it is working

Types of machine learning

- **Supervised learning*** (e.g. regression, classification)
  - You have examples of expected outputs for a set of inputs
  - For instance, a set of **training images** where each image contains either a **hotdog** or **not a hotdog**, and you know which is which
  - These are then used to train a machine learning method to **classify images** where you don't know the answer already as either "hotdog" or "not a hotdog"
- Unsupervised learning (e.g. clustering)
  - You don't know the classes/types
  - You want the machine learning method to find the patterns for you and create classes
- Reinforcement learning
  - Software agents receive stimulus from an environment and produce behaviours to maximise some cost function
  - For instance, playing Pacman to maximise the score

Example: (Linear) Support Vector Machines

- Two or more classes of data point (potentially high dimensional)



- During **training** on points of known type the SVM finds a line (hyperplane) that separates them and maximises margin between the classes
- New data points are then **classified** accordingly

  e.g. any coordinate pair (x,y)

Images and Machine Learning

•Image Basics: Pixels

- An image is made up of a rectangular array of dots called **pixels**
- Each pixel has a value in the **range 0 to 2^n -1** for some integer n
- Typically, **n might be 8, 16 or 24** (8 bit, 16 bit or 24 bit)
- correspond to 0..255, 0..65535, 0..16777215
- A bright pixel would have a high value, e.g. 255
- A dark pixel would have a low value, e.g. 0
- **Colour** is usually represented as triples of numbers.
- For instance, in **RGB** format the triple (255,255,0) would represent maximum brightness **R**ed and Green, with no **B**lue.
  - Most people would perceive this as yellow.

• The Mathematics of Image Analysis

- At its core, image analysis applies **functions** to **matrices** of numbers.

- We denote the values (intensity) in row x, column y by **I(x,y)**

- E.g. A **brightness increase** might be performed by the function

  I'(x,y) = I(x,y) + 20

- E.g. an **object detector** assigns individual objects a unique number

  I'(x,y) = 255, if I(x,y) > 100

  0, otherwise

29	27	17	24	20
24	99	134	71	13
21	148	157	134	11
21	31	31	14	42

49	47	37	44	40
44	119	154	91	33
41	168	177	154	31
41	51	51	34	62

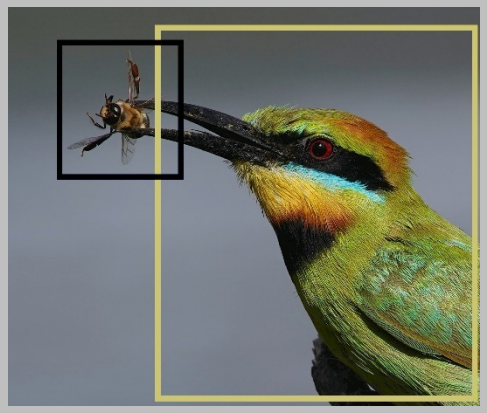
0	0	0	0	0
0	0	255	0	0
0	255	255	255	0
0	0	0	0	0

Applications of Computer Vision

- Classification

Rainbow bee-eater

- Object Detection



Bee-eater, bee

- Semantic Segmentation

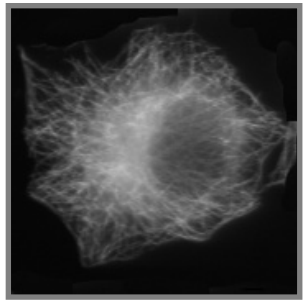
Bee-eater, bee, stick, background

- Instance Segmentation

Bee-eater 1 and 2

Outline of Image Classification using Machine Learning

Image Classification



=

```
(  0.000772532, 0.00154506, 0.00343348, 0.0139056, 0.0308155,
   0.0440343, 0.0337339, 0.0433476, 0.828412, 0.000235031,
   0.000352547, 0.00205653, 0.00593454, 0.0223867, 0.0351372,
   0.022093, 0.0312004, 0.880604, 0.000420663, 0.00117786,
   0.00496382, 0.0165741, 0.058388, 0.107017, 0.0904425,
   0.110803, 0.61024)
```

=

Cytoskeleton

Image Statistics Generation

Machine learning

The key is to generate image statistics that **distinguish** the classes of imaging

Image Classification

APPROACH

- Two sets of images:
  - A **training/test set** for which the **class of each image is known**
  - A large number of images that you want to classify

- Generate **image statistics** for each image

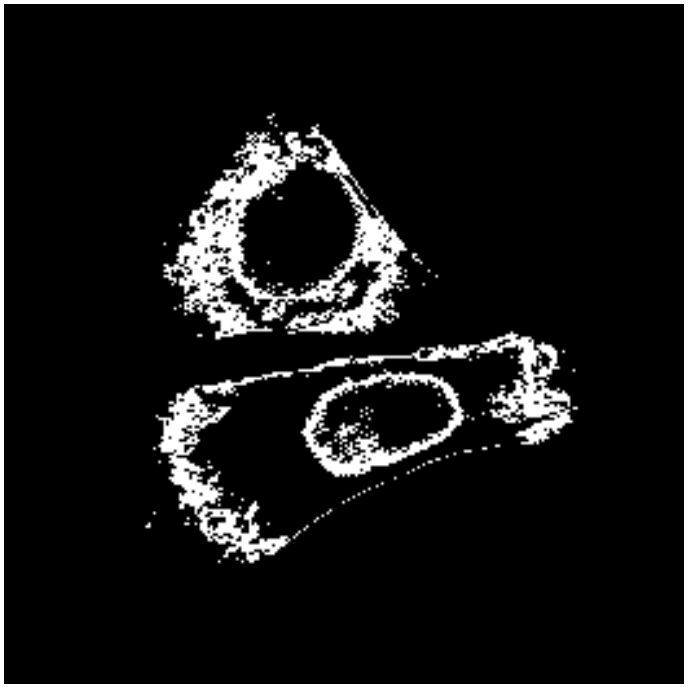
- Train a **machine learning** technique such as a Support Vector Machine (SVM) or Neural Network (NN) to *distinguish* the classes on the training/test set

- Use the trained SVM or NN to **classify** the other images

- Have a coffee while the machine does all the work

Threshold Adjacency Statistics

ER

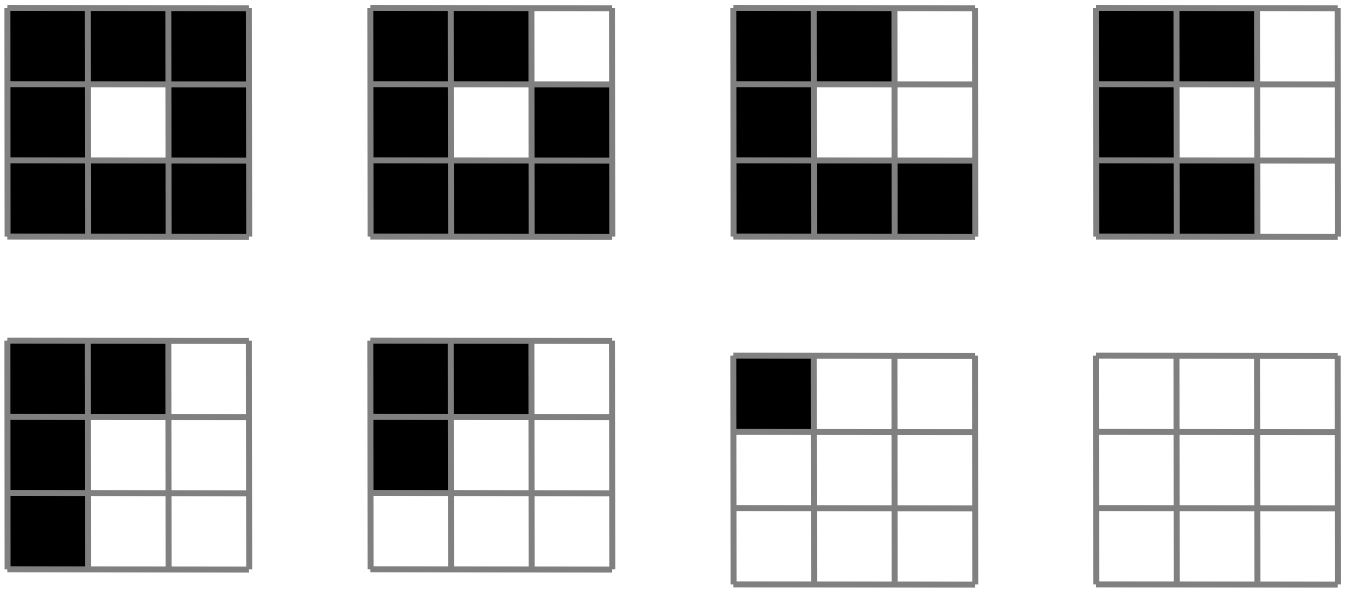


Microtubules

Let μ be the **mean intensity** in the image

(Band pass) **threshold intensity** in range (μ, μ+30)

Threshold Adjacency Statistics (TAS)

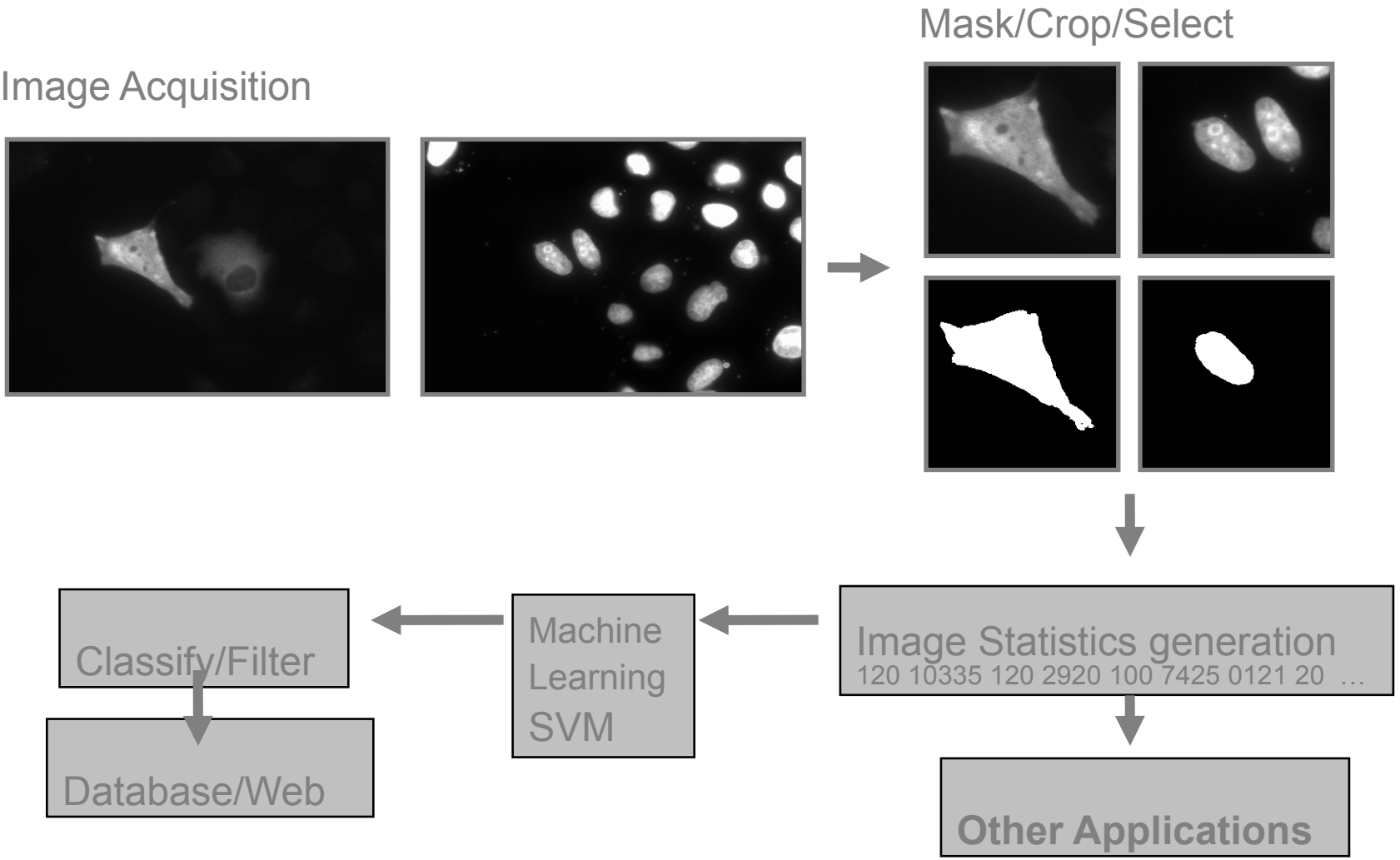


- Count number “on” pixels with
  - 0 on neighbours
  - 1 on neighbour
  - …
  - 8 on neighbours

→

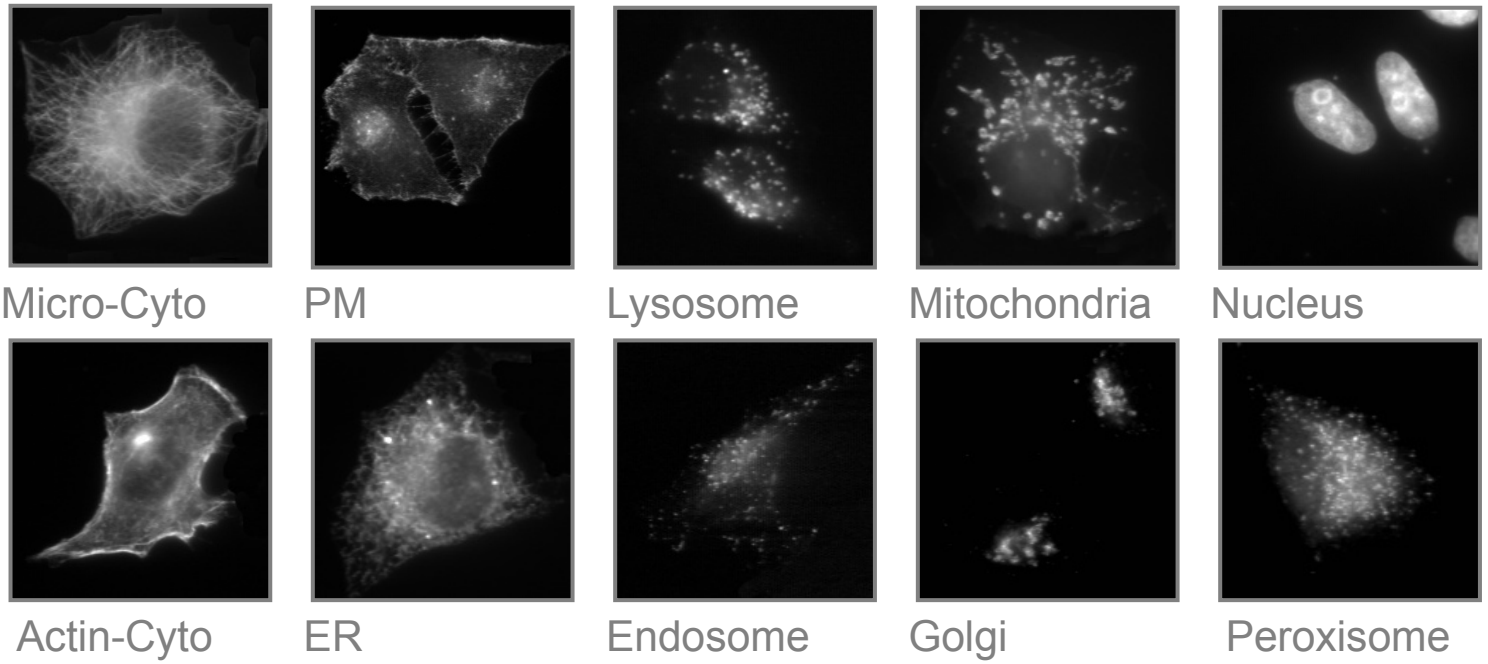
- Nine image statistics

(Protein Cell) Image Classification
- where is the protein in the cell



Hamilton, Pantelic, Hanson, Teasdale, *BMC Bioinformatics 2008*

(Protein Cell) Image Classification

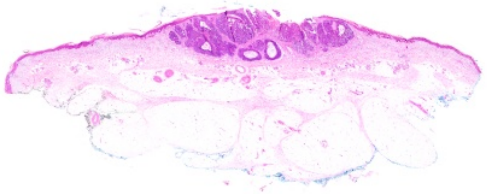


- HeLa cells, fluorescent imaging, 60X
- 10 locations * 50 plate images/class = **500 plates**

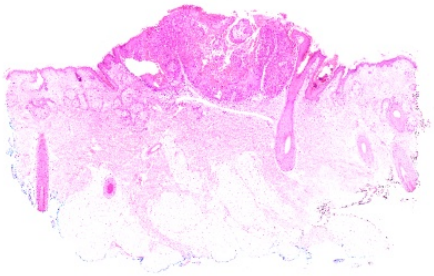
- Up to **95%** correctly classified using **TAS image stats + SVM**

- **CellProfiler** now has facilities for machine classification of cell imaging

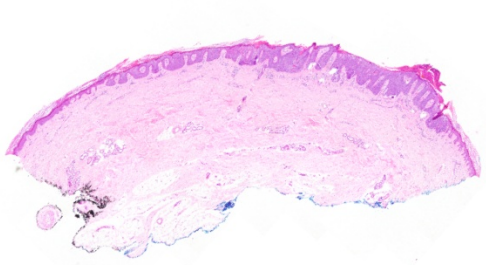
Non-Melanoma Skin Cancer Image Classification



Basal Cell Carcinoma



Squamous Cell Carcinoma



Intraepidermal Carcinoma

Histology Imaging

Simon Thomas (2018) imaged

284 Images

(x10 Magnification)

- 135 Excisions
- 91 Shave Biopsies
- 58 Punch Biopsies

of **BCC, SCC, IEC, normal** that had been classified by an expert pathologist

99% accuracy training a convolution neural network

Image Segmentation and Machine Learning

Image Segmentation and Machine Learning

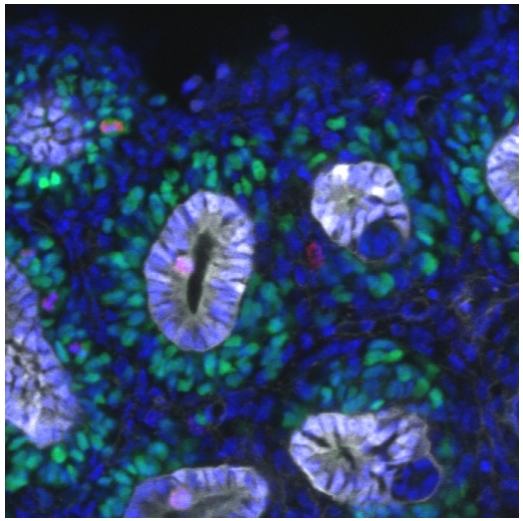
Definition

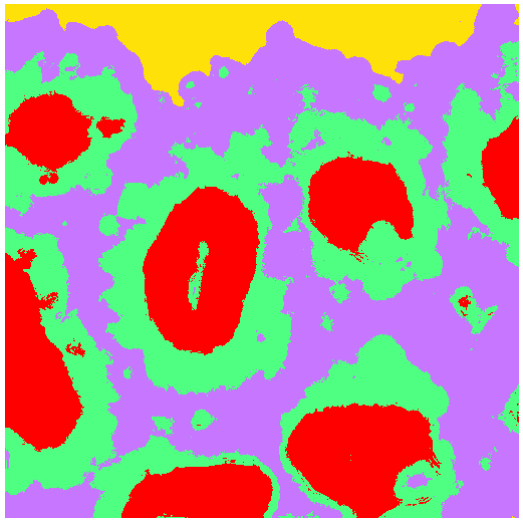
A segmentation is a partition of an image into regions

e.g. foreground/background

Approach

Instead of a whole image we want to classify individual pixels



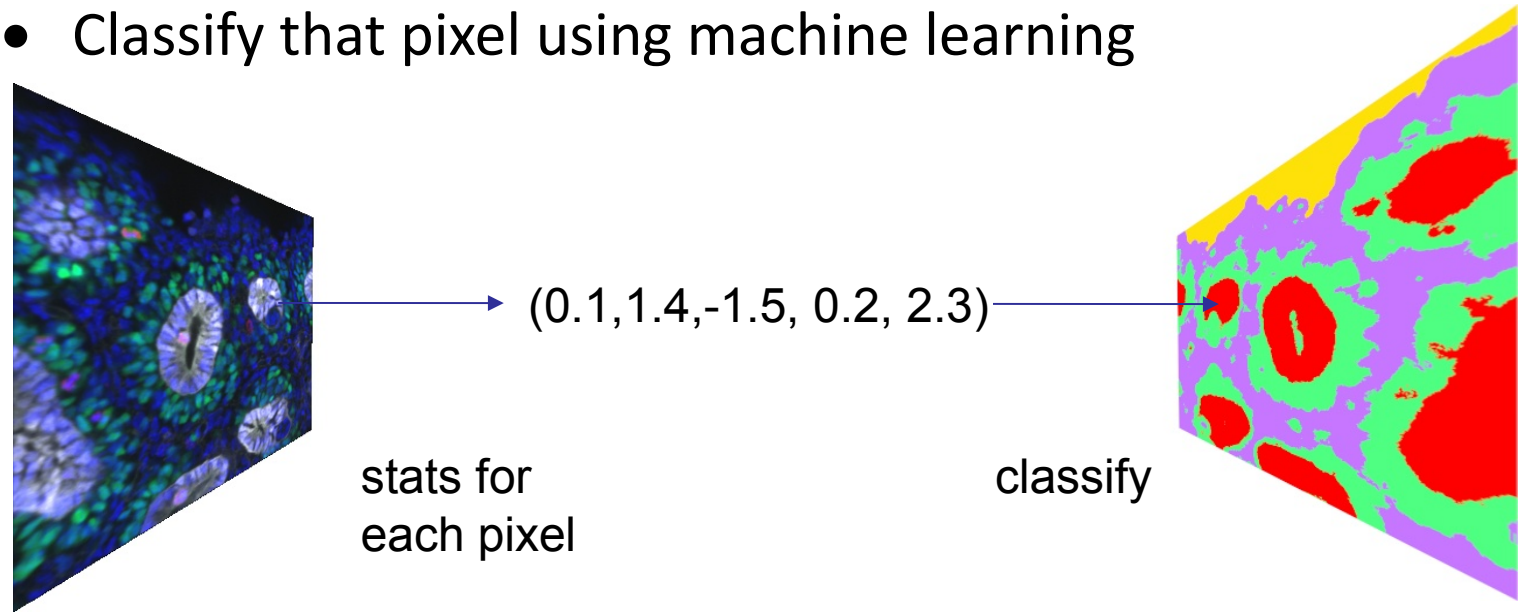


Ureteric tree
Cap mesenchyme
Stroma
Background

Image: Alex Combes / Melissa Little Lab

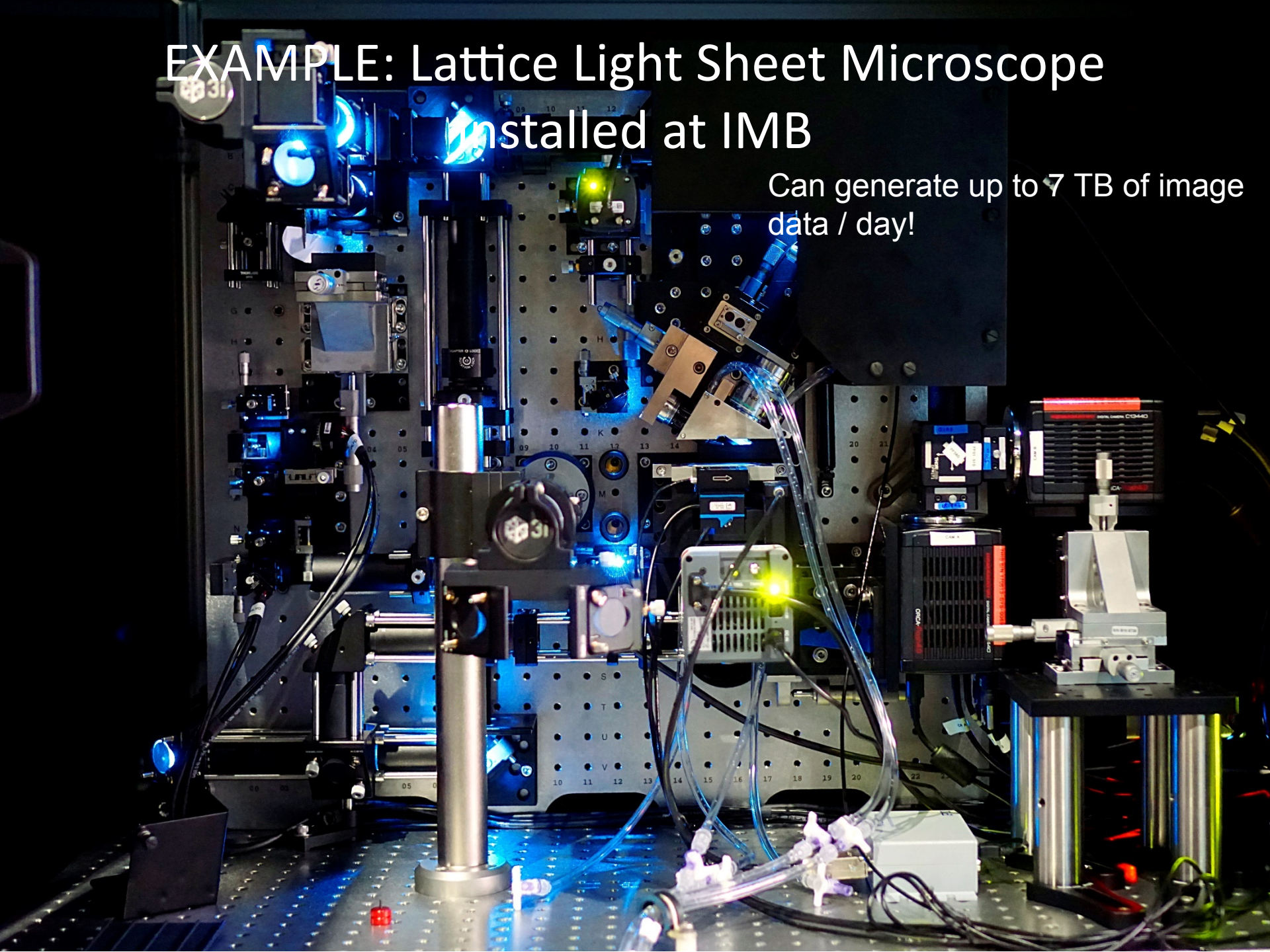
Image Segmentation and Machine Learning

- Generate statistics for **each pixel** in the image
- E.g. intensity, local median intensity, gradient, …
- Classify that pixel using machine learning

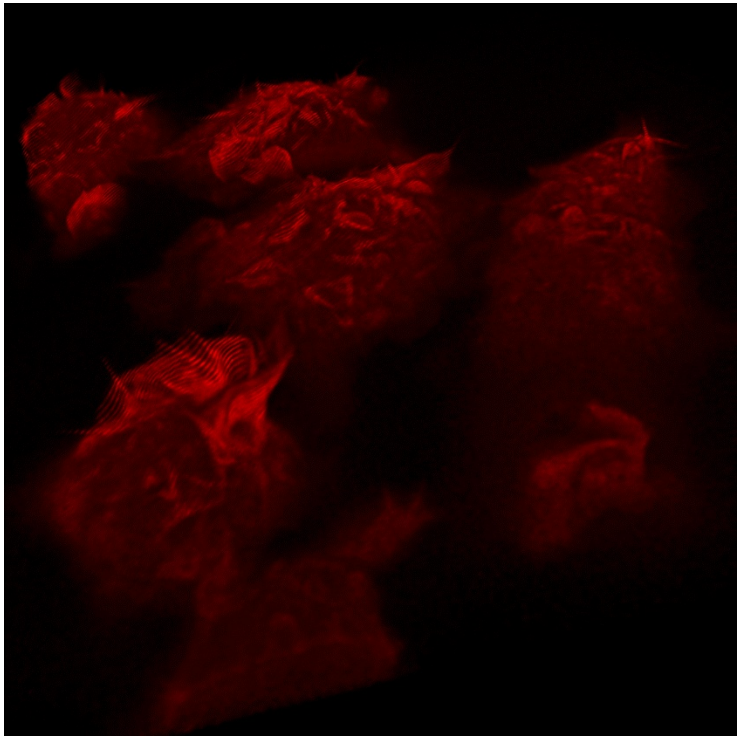


EXAMPLE: Lattice Light Sheet Microscope installed at IMB

Can generate up to 7 TB of image data / day!

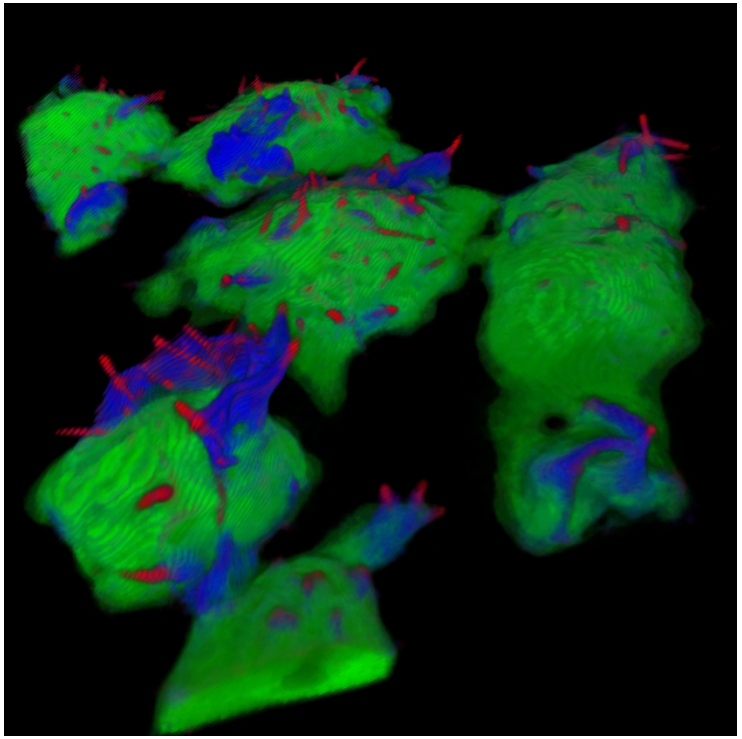


Segmentation of 3D LLSM imaging



Macrophage cells

(Image:Nick Condon / Stow Lab)

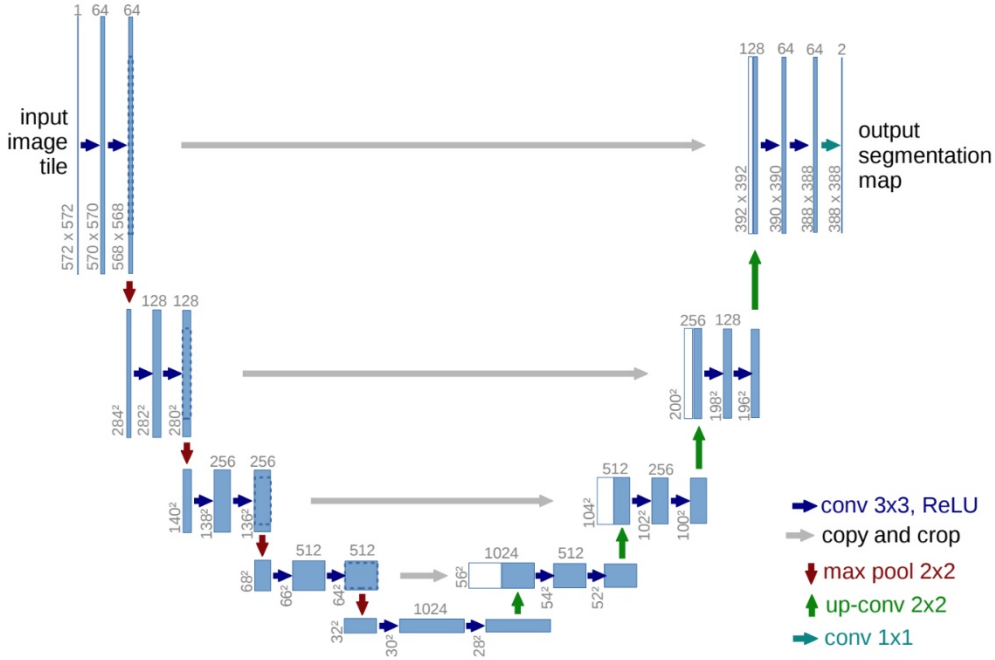


Segmentation into “tent poles” , ruffles, cell surface

(Image: James Lefevre / Hamilton Lab)

Another approach: convolutional deep learning with U-net

- **U-net** was a new type of neural network architecture introduced in 2015
- It down-scales then upscales an image to create a segmentation
- It uses convolutions, max-pooling and skipping



- U-Net and its variants are the state of the art in segmentation
- Later in the year we will run advanced U-net workshop
- U-net paper: https://arxiv.org/pdf/1505.04597.pdf

ImageJ and Trainable WEKA

- pixel classification for segmentation

Trainable Weka in ImageJ/Fiji

https://imagej.net/Trainable_Weka_Segmentation

- **ImageJ/Fiji** is open source image analysis software widely used in biology and many other fields
- **Trainable Weka** is a plugin for Fiji that provides a number of machine learning methods and image statistics generation to **segment imaging**
- It builds upon Weka, a powerful and general machine learning environment
- Users select example regions / classes in an image
  - These are used to train a machine learning model to predict the class of each pixel in an image
  - The model may then be applied to segment individual or whole directories of images
- Works on grey or RGB images

The Trainable Weka Pipeline

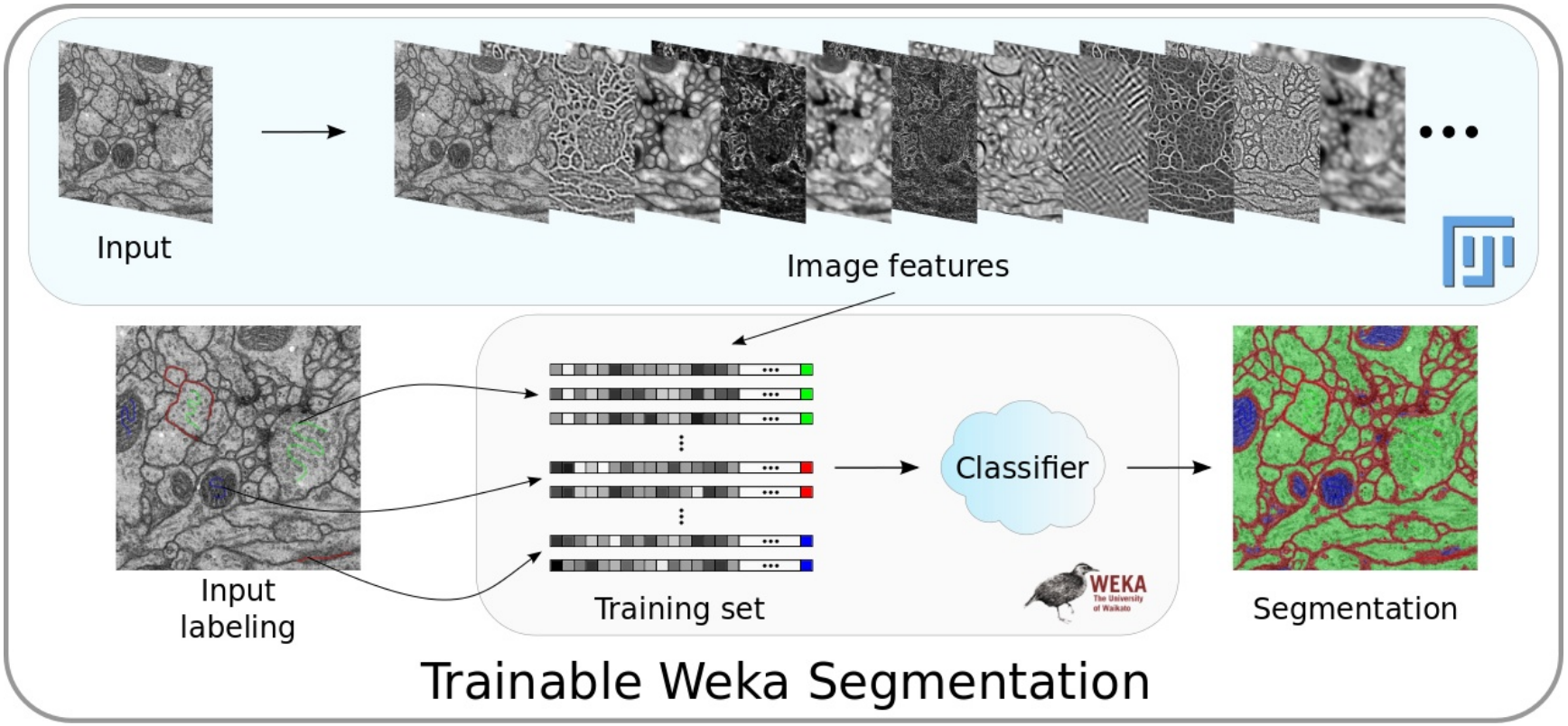


Image: https://imagej.net/Trainable_Weka_Segmentation

Demonstration of using Trainable Weka on a pretty kidney image

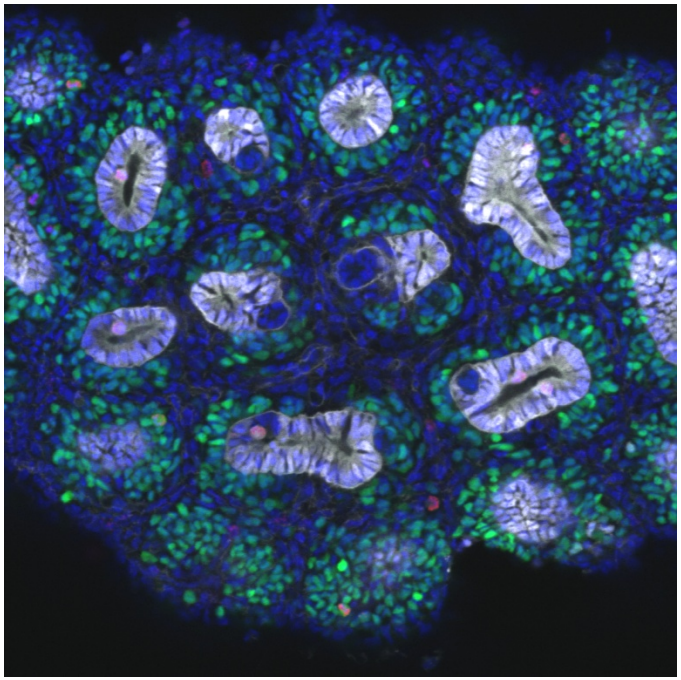


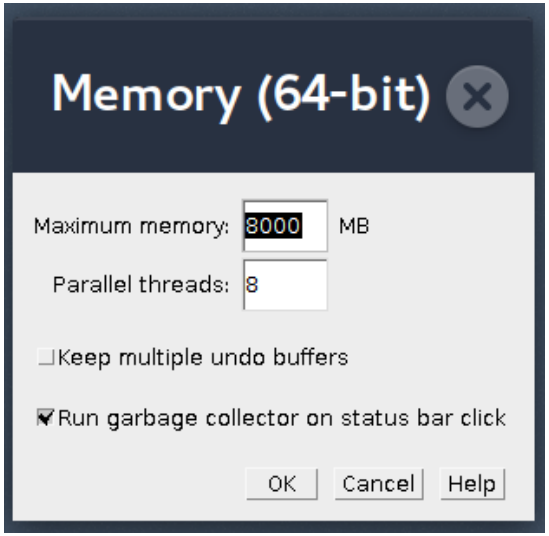
Image courtesy of Alex Combes / Little Lab

Trainable Weka Practical I

Creating a simple 2D image segmenter and checking the results

Before starting: marking sure you have enough memory available in Fiji

- Trainable Weka and machine learning generally use a lot of memory
- Almost every error you ever see will give some **unreadable garbage** which you have no idea what it means (this is my personal experience, anyway). It's almost always memory
- To increase memory in Fiji:
  - Go to Edit > Options > Memory and Threads
  - Select a "reasonable" amount of memory, e.g. on a 16GB system, use, say 8GB
  - If you have a multi-core system set the parallel threads parameter as well
  - Do click "Run garbage collector" as it cleans up memory every time you click the status bar



Getting Started with Trainable Weka (TW)

Open a **Salmonella image** in Fiji : File > Open
Start Trainable Weka : Plugins > Segmentation > Trainable Weka

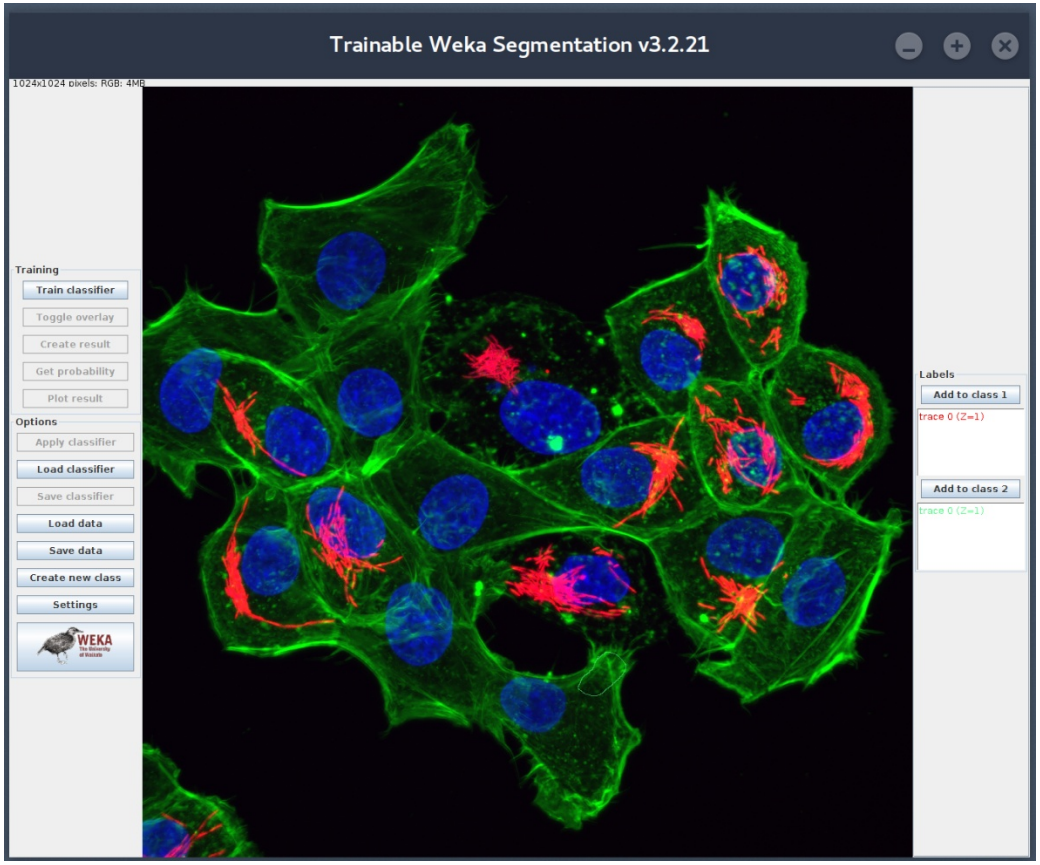
Start training

Toggle showing training results

Create segmentation

Add a new class of points

Settings
- Learning method
- Statistics



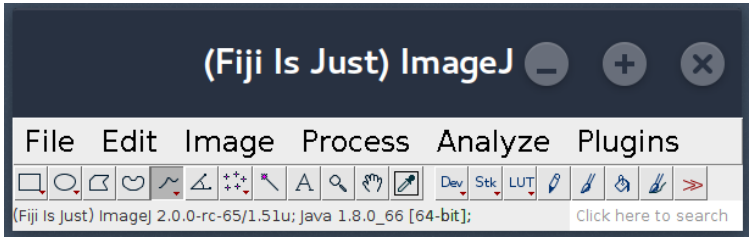
Classes of points

Zoom in/out: **Ctrl + scroll** or **Crtl +/-**

Image courtesy Dr Markus Kerr / Teasdale Lab

Selecting areas /classes and creating a classifier

Regions can be selected using the Fiji selection tools



Selection tools

In the TW window, create new classes “nucleus” and “background” (the 2 classes that are already there can be “salmonella” and “cell”)

On the image, select some points of classes **salmonella, cell, nucleus, background** and add them to their respective classes

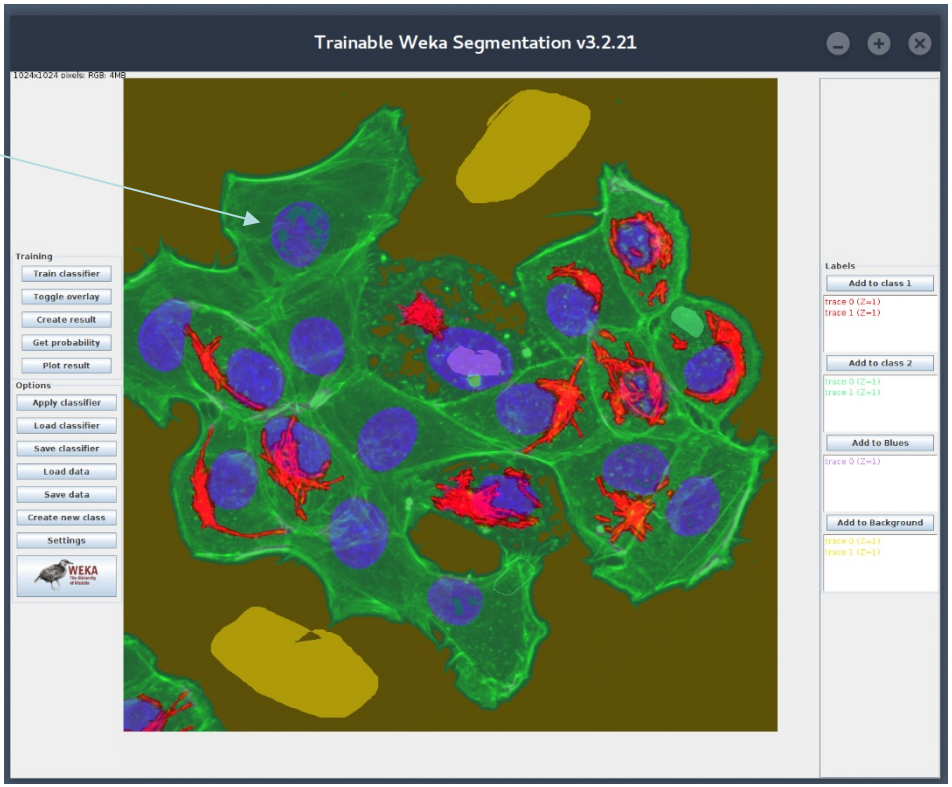
Click **Train Classifier** to start the training

Observe the results

Click **Toggle overlay** to compare the segmentation against the original image

Fixing and retraining the classifier

Some areas will probably be misclassified. For instance in this image, some of the nucleus is classified as cell membrane



Select some of these areas and add them to the correct class

Train the classifier again and observe the results

Repeat until you are happy with the results

Use **Create result** to get a final segmentation

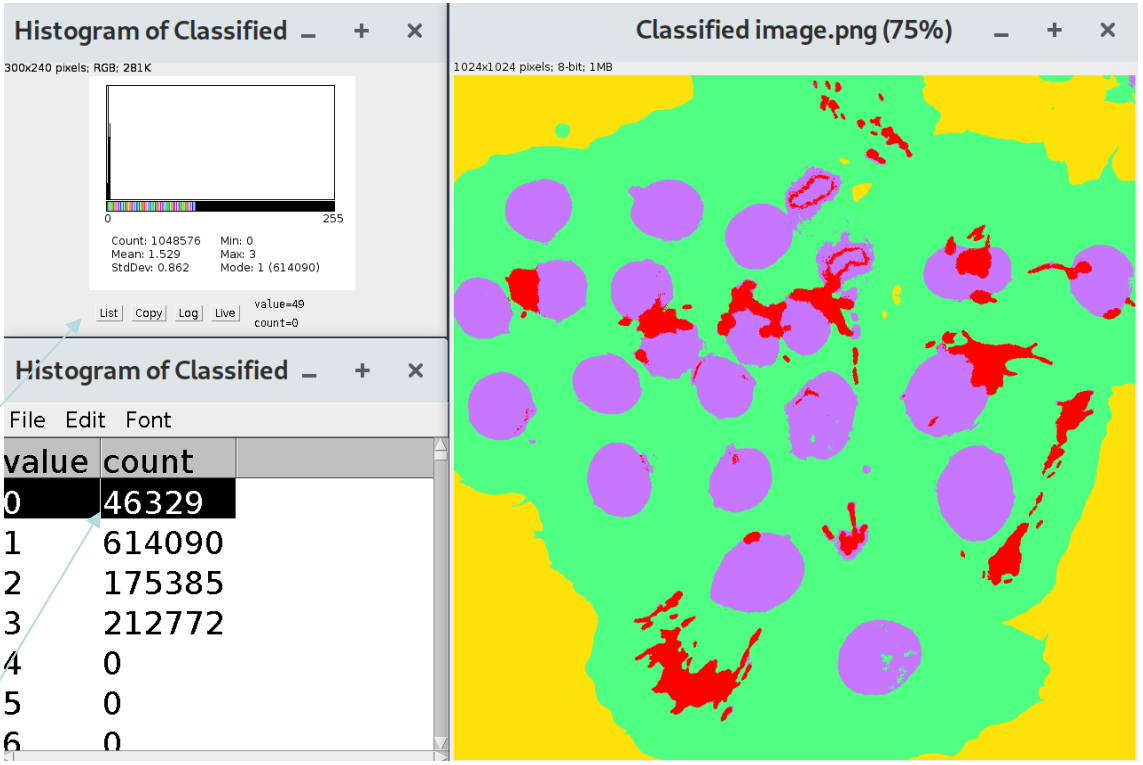
Counting the area of each region

In many cases you will want to know the area of each region type in the segmentation.

To do this, select the **Classified Image window**, and on the Fiji menu select
Analyse > Histogram

On the Histogram window click **list.**

A table listing the number of **pixels** for each of the labeled regions is then created. The order of the list is the same as for the class labels in the Trainable Weka interface.



Note: The numbers in the table are in **pixels** and you may want to convert these to an **area**, depending on the resolution of your imaging.

Applying your classifier to other images

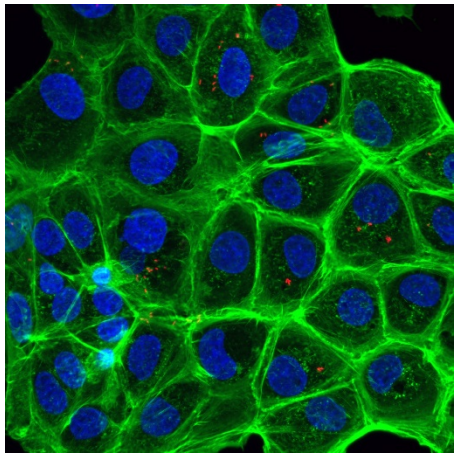
1. Click **Apply classifier** and select **a single image** from the Salmonella folder
(select “no” to probability map, when asked)

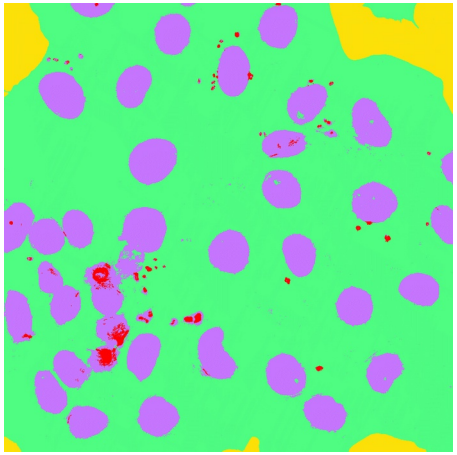
How do the results look? Is the classifier generalising?

2. Click **Apply classifier** and **select all the images** from the Salmonella folder. Create a new folder when asked to put the results images in
(select “no” to probability map, when asked)

3. Open each source image and its segmentation & compare

How do the results look? Is the classifier generalising?





Trainable Weka Practical II

A closer look at the image statistics classifier types and problems such as overfitting

What do the image statistics look like?

1. Open the 2D kidney image and start Trainable Weka

2. Create a classifier to distinguish the five classes of points shown in the image here

Note: a “fat” line also defines a set of points.
Line width can be set by double clicking the line tool

3. Save the Feature Stack to disk:
   - In TW click **Settings**
   - Click **Save Feature Stack** and save it to a location on your computers disk

4. Drag the Feature Stack file from disk onto Fiji to open it

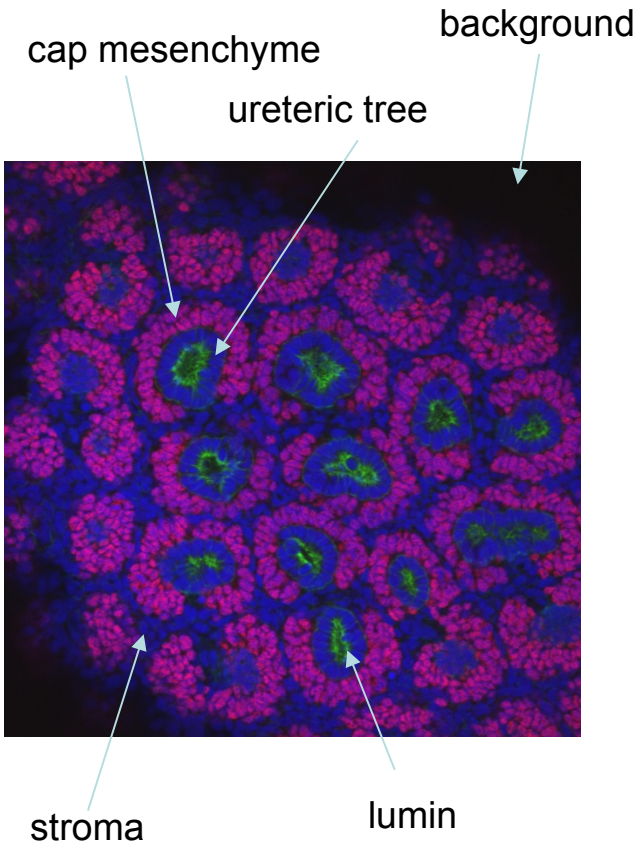


Image courtesy Dr Alex Combes / Little Lab

What do the image statistics look like?

This stack of images is image representations of the statistics that TW calculated for each pixel

The first image is the original image

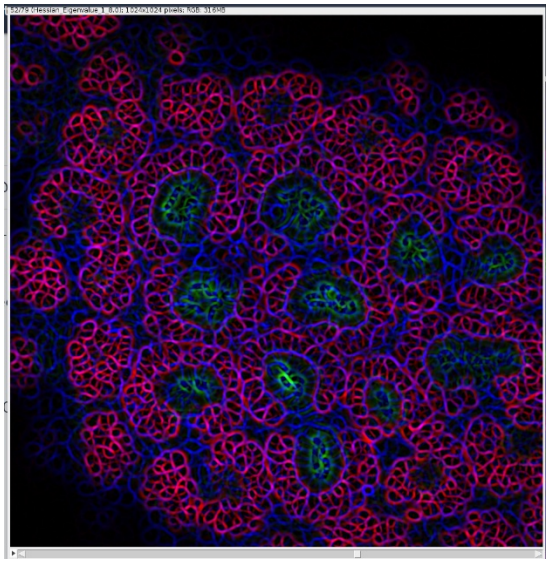
The 78 other slices correspond to the 78 (default) statistics that TW calculates

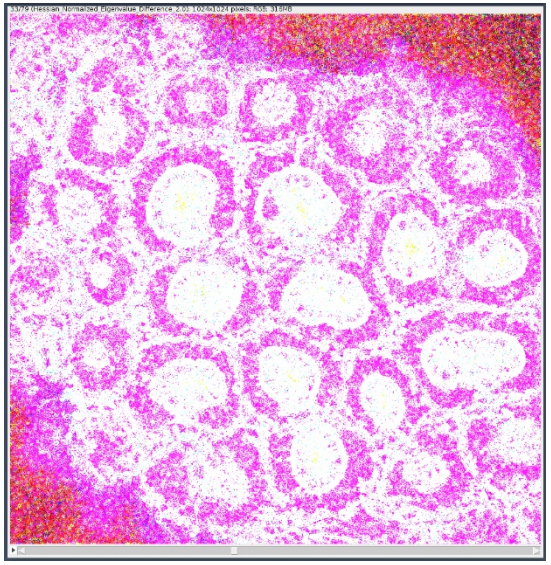
Use the slider at the bottom of the image to move through the stack

Each slice is labelled with the statistic and parameters

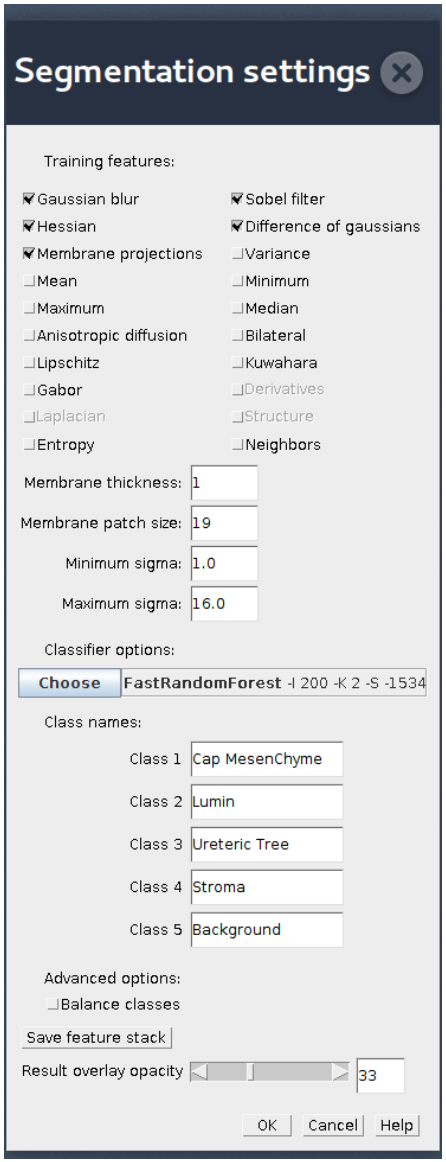
Can you see how certain statistics distinguish certain regions?

More information on the statistics calculated on TW site:
https://imagej.net/Trainable_Weka_Segmentation





A closer look at the Image Statistics in TW



At the top are the currently selected statistics used for classification

More information on the statistics on the TW site:
https://imagej.net/Trainable_Weka_Segmentation

The Sobel filter, for instance, is an edge detector

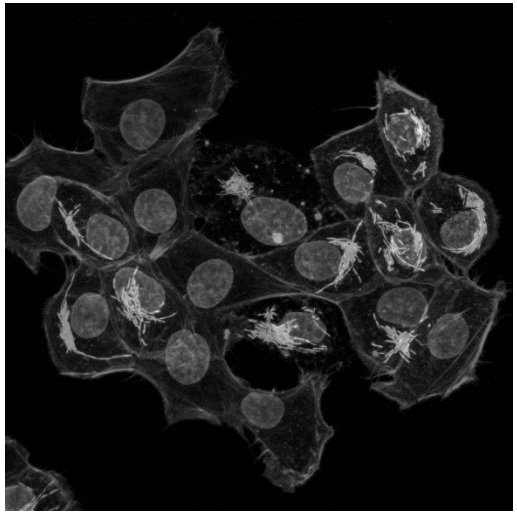
Parameters to some of the statistics, such as what radius (sigma) to act over, are in the middle

The default sigmas range over 1, 2, 4, 8, 16

The larger sigma, the more computationally expensive it will be.

Some statistics are greyed out as they require the ImageScience library to be installed

Trying other Image Statistics in TW



Open the Salmonella-BW image in TW

Create a classifier that distinguishes, **nucleus, cell membrane, salmonella, and background**

Click **Create Result** so you have a reference to compare

Some things to try:

- In Settings, select only **Mean statistics**, click ok and then **Train Classifier** in the main TW window. How did it do?
- In Settings, select **Save Feature Stack** and then drag the saved stack into Fiji and examine the slices. These are the images obtained by averaging each pixel with those around them at radii =1, 2, 4, 8 and 16 pixels
- In Settings, try setting the maximum sigma (the averaging radius) to either 1, 2, 4, or 8, and run Train Classifier again. How did it do?

- Try doing all of the above, but using the Sobel feature instead (remembering to set sigma to 16 initially)

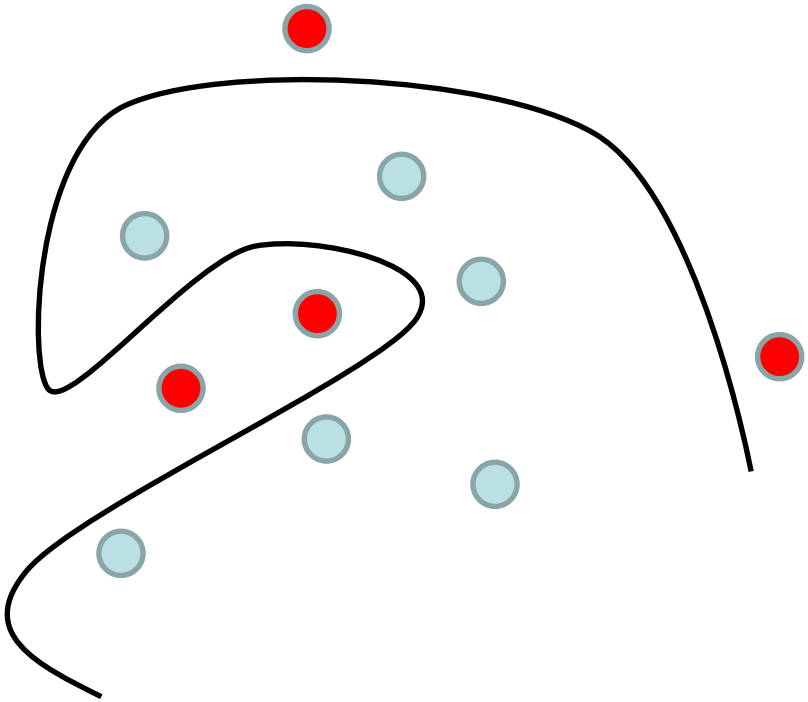
It can be surprising how little information is needed for a reasonable classification
Sometimes statistics can be costly to calculate, but do not necessarily add much

Overfitting the data

Sometimes a machine learning model can fit the data it is trained on very well, but it fits so well that it does not generalise to other data sets. This is called overfitting.

For instance, the black line here is a “classifier” that separates the blue and red dots.

But it is questionable whether the two types of dots are really distinguishable.

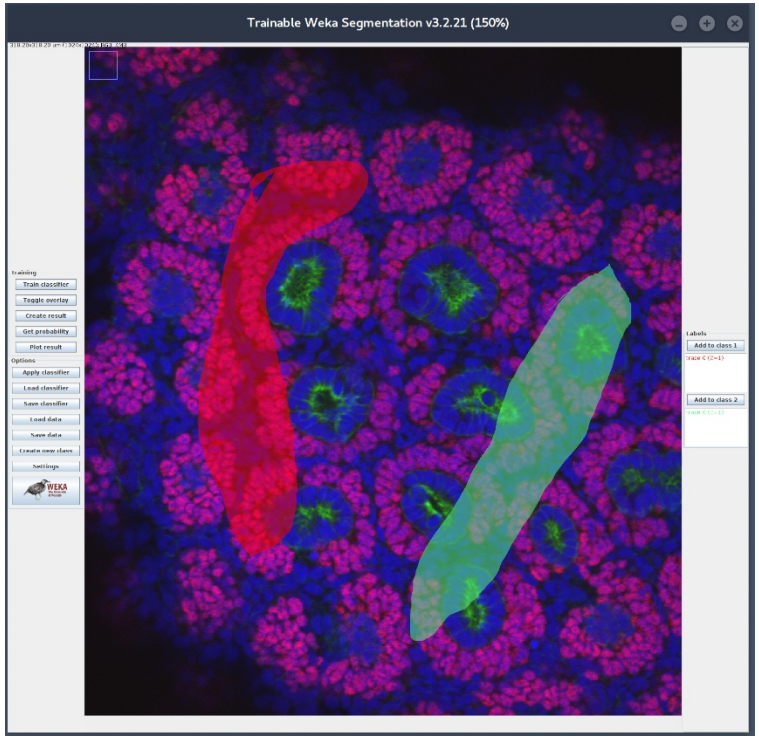


Maybe the statistics do not properly distinguish the classes we want to distinguish.

In the next exercise we will create a crazy image segmenter, then see how well it generalises

Creating a crazy classifier

1. Close your current Trainable Weka session, and open up the 2D kidney image again
2. Start TW again
3. Create a two class classifier with two regions where it is clearly not possible to reasonably distinguish them (see right), adding those regions to distinct classes, then clicking **Train Classifier**



How does it look? Click **Toggle Overlay** to see how well it has correctly classified points in
(a) the regions you selected?
(b) outside those regions?

It may be easier to see if you click **Create Result** to create the segmentation mask

Some methods of machine learning are more prone to overfitting that others
That is what we will look at next

Trying other machine learning methods

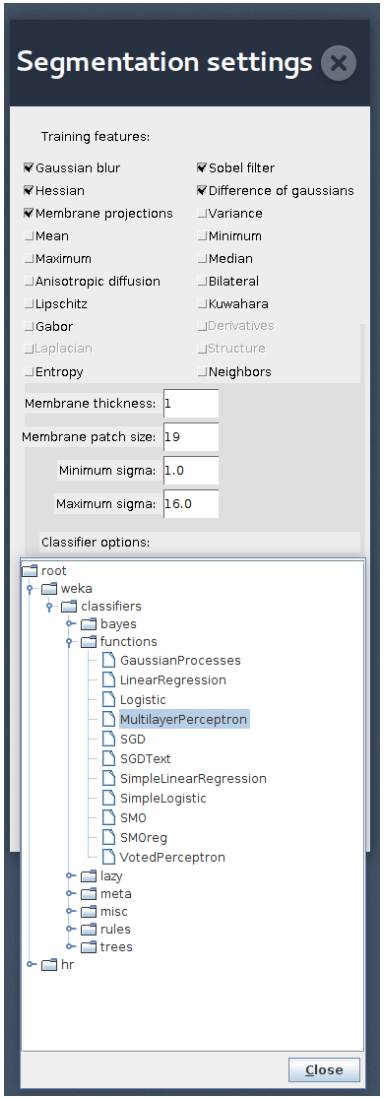
The default machine learning method in TW is called **random forests**

There are many other methods of machine learning available in TW

For instance, MultilayerPerceptron is what is often called a **Neural Network**

Depending on your data, they may be more or less prone to overfit and not generalise well. They may also take more or less time to train

In TW these may be selected in Settings > Choose



Trying other machine learning methods

Keeping the crazy classifier you created before, lets try some other learning methods.

1. Select Settings > Choose > MultilayerPerceptron and click OK
   Click **Train Classifier** to create a classifier

   After a while you will get **bored** waiting for it to finish: training **Neural Networks** can be slow!

   Click **Stop** to kill the Neural Network training

2. Instead select Settings > Choose > SGD and click OK
   This is a type of **Support Vector Machine,** another popular training method.

   Click **Train Classifier** to create a classifier

   Look at the results, possibly using **Create Results**

   Did this correctly classify the regions you defines as Class 1 and Class 2? Does this look like overfitting? Is this better or worse than the (default) Random Forests learning method?

Trainable Weka Practical III

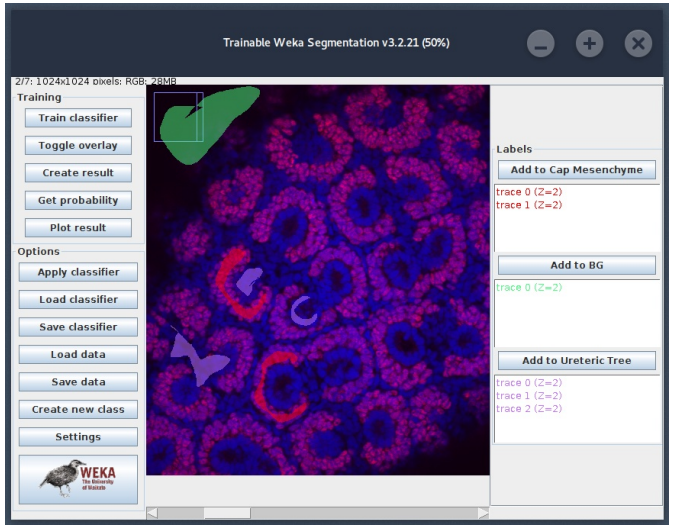
Moving into 3D

3D segmentation in Trainable Weka

- 3D segmentation *could* be done in TW by segmenting each of the 2D slices individually
- However, many structures in 3D would be difficult to distinguish only looking at 2D slices
- Hence TW has several 3D image statistics that consider the 3 dimensional region around each pixel
- For instance, the 3D mean or median intensity around a pixel

- The interface for 3D segmentation in Trainable Weka is similar 2D, though there is now a slider under the image to select different slices

3D segmentation of kidney data



Load the 3D kidney dataset into Fiji and start **Trainable Weka 3D**

We will create a classifier that distinguishes the cap mesenchyme (pink), the rest of the kidney (blue), and background (black regions)

Select a slice and add regions for each class. You will need to **Create New Class** so that you have 3 classes to use.

Click **Train Classifier** to begin the classification.

Use the slider under the image to look at the results. Fix any errors you see by selecting them, adding them to the correct class, and retraining the classfier.

Click **Create Result** to see the final segmentation. How did it do?

Save the feature stack (Settings > Save feature stack) and open the stack(s) in Fiji. Why are there multiple images?

You will notice that it takes a lot longer to both generate the statistics (because they are now 3D) and classify the images (because there are more of them).

Other things to try in Trainable Weka

- Try TW on **your own data**!
- Have a closer look at what each of the statistics TW uses does (see https://imagej.net/Trainable_Weka_Segmentation). For instance, membrane projections is interesting!
- **Get probability** shows the probability each pixel is in each class rather than forcing it into a particular class.
- **Plot results** gives a true positive / false positive ROC that gives an indication of how well the classifier is doing (hard to read!)
- **Balanced training.** If a training set has many more examples of one class than another, this can bias it to performing better on classification. Try creating a unbalanced training set and testing the performance of the classifier. TW has

  Setting > Balance Classes

  to avoid this problem by resampling

Other things to try in Trainable Weka

For power users:

- Data can be exported from Trainable Weka into Weka to provide an enhanced training and analysis environment

  The Weka button in the bottom left corner starts this environment
- There is an API for programming TW directly in Fiji using the macro language. See https://imagej.net/Trainable_Weka_Segmentation