Cell Segmentation from DIC Images - Tutorial

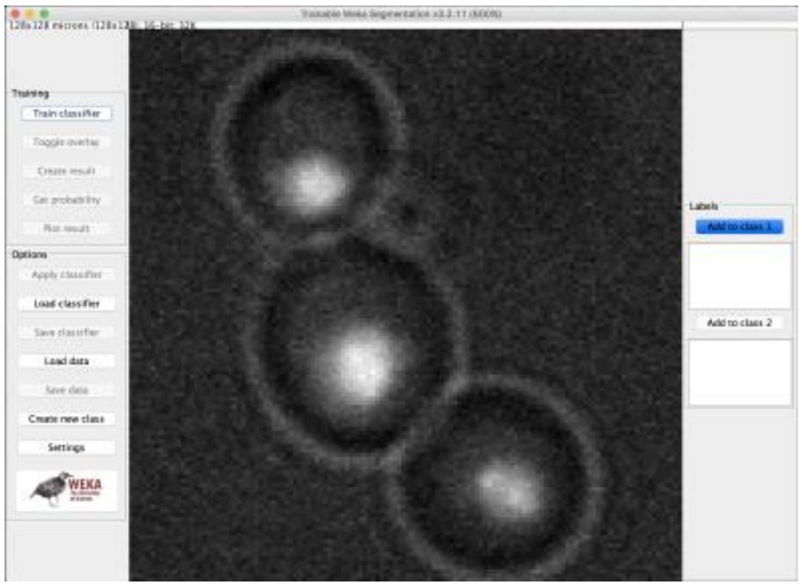
Johns Hopkins University, Wu Lab
Author: Sun Jay Yoo
Date: May 24, 2017

Cell segmentation from DIC images using Machine Learning relies upon the ***Trainable Weka Segmentation***[1] *plugin, which is part of the default libraries that come preinstalled with Fiji. Weka*[2] *(Waikato Environment for Knowledge Analysis) is the broader plugin being used, which contains a collection of visualization tools and machine learning algorithms for predictive modeling in a usable graphical user interface. It is freely available under the GNU General Public License*[3]*.*

Tutorial (with example):

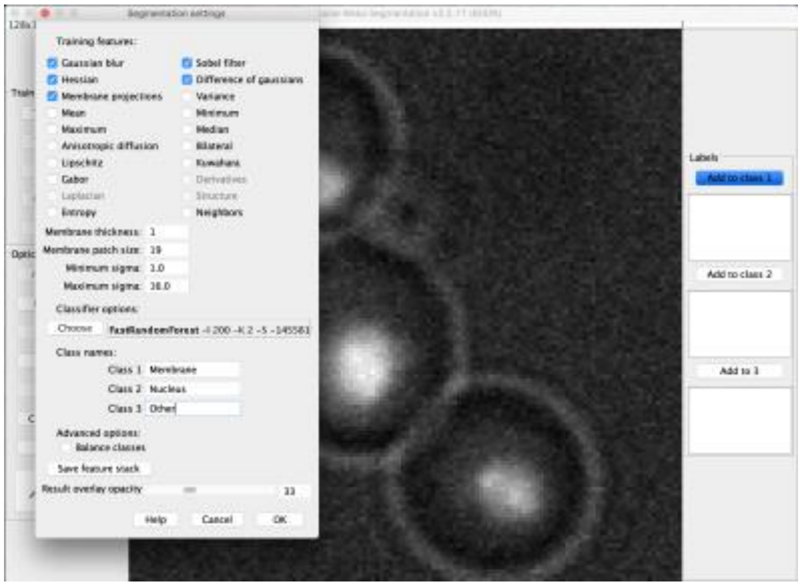
1. Open Fiji and select *Plugins > Segmentation > Trainable Weka Segmentation*. A file browser should open and select the first image (a *.tif* file) you desire in your training sample. Typically, you want to use an image with generic/homogenous features without any abnormal or outlier segments (e.g. cell division or patchy images due to noise).

2. A graphical user interface should pop up with the desired sample image in the center.



3. By default, there are two classes (representing two features in the image). Since we want three classes (cell membrane, nucleus, and other), select *Create new class* and give it any name (we can rename it later). If the image does not have visible nuclei and only the membrane needs to be segmented, skip this step as only two classes are then needed.

4. Next, select *Settings*, and change the *Class names* appropriately. *Training features* can be left untouched, but can be adjusted if feature extraction algorithms need to be changed (same goes for *Classifier options*). Select *OK* to save classifier settings.



Segmentation settings

Training features:

- [✓] Gaussian blur
- [✓] Hessian
- [✓] Membrane projections
- [ ] Mean
- [ ] Maximum
- [ ] Anisotropic diffusion
- [ ] Lipschitz
- [ ] Gabor
- [ ] Laplacian
- [ ] Entropy
- [✓] Sobel filter
- [✓] Difference of gaussians
- [ ] Variance
- [ ] Minimum
- [ ] Median
- [ ] Bilateral
- [ ] Kuwahara
- [ ] Derivatives
- [ ] Structure
- [ ] Neighbors

Membrane thickness: 1
Membrane patch size: 19
Minimum sigma: 1.0
Maximum sigma: 16.0

Classifier options:
Choose FastRandomForest -I 200 -K 2 -S -145581

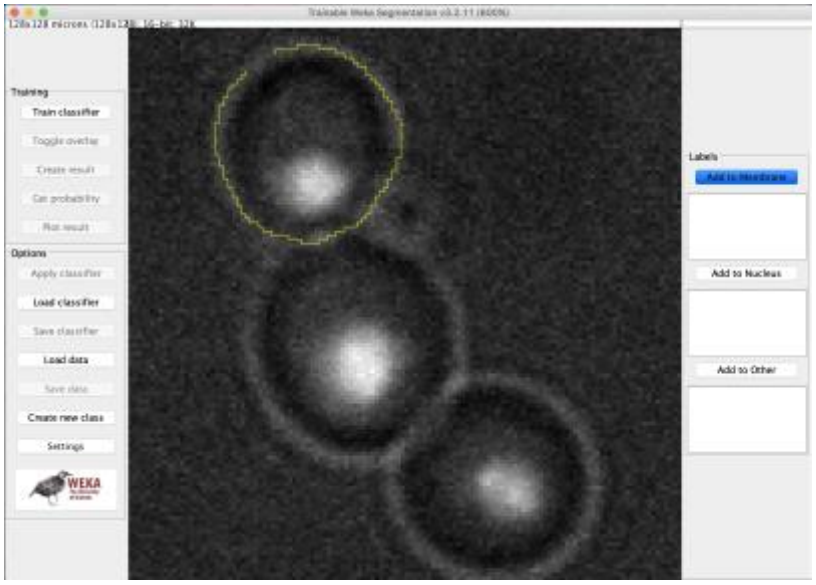
Class names:
Class 1 Membrane
Class 2 Nucleus
Class 3 Other

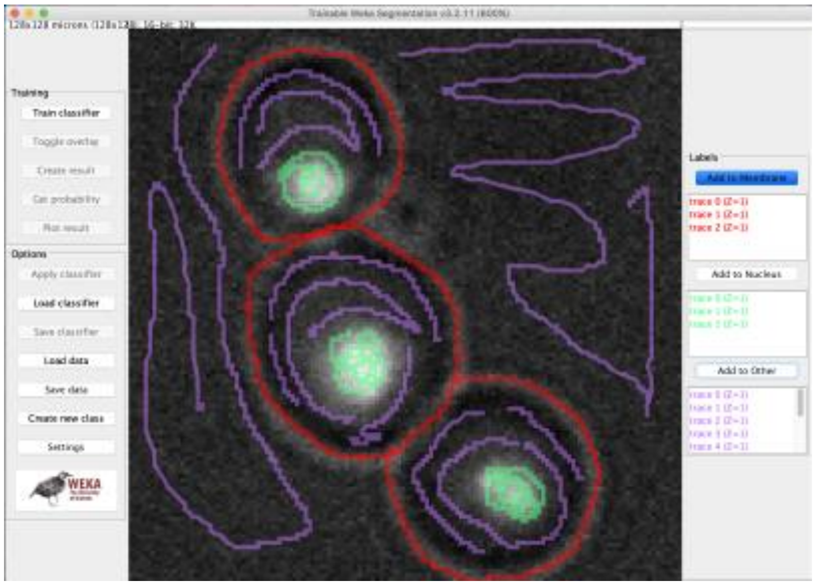
Advanced options:
- [ ] Balance classes

Save feature stack
Result overlay opacity 33

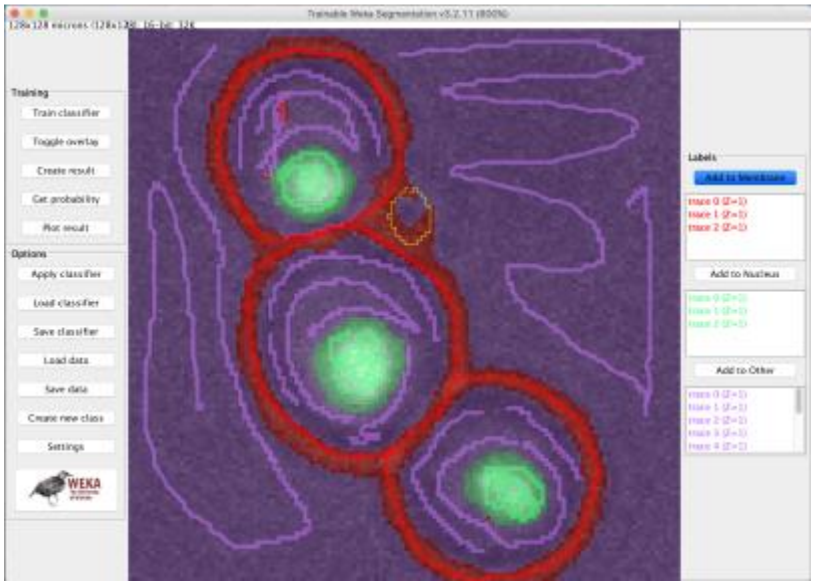
Help Cancel OK

5. Next, use your mouse to draw over sections of the image that match the corresponding class and select *Add to [CLASS NAME]* on the right side of the interface. Repeat for every class and try to add as many as possible. If an incorrect drawing was classified into a class, select that trace on the right and press *Enter* to delete it.





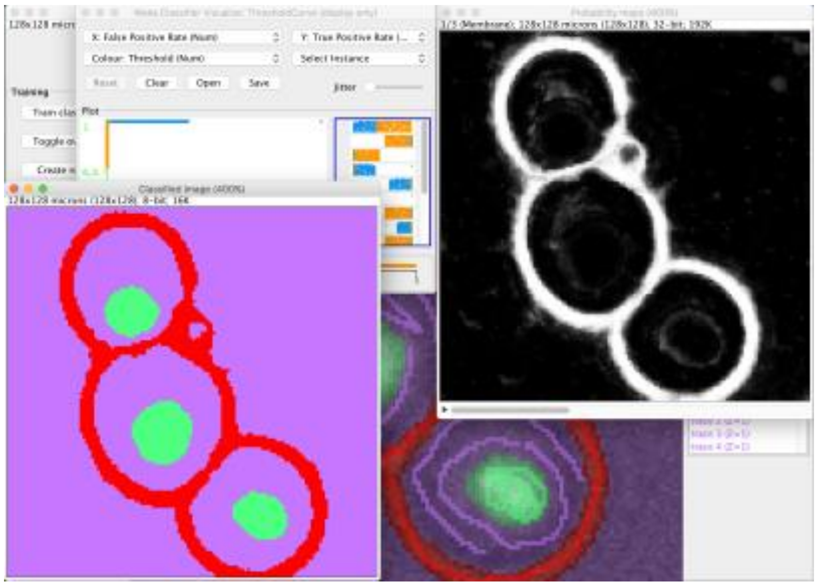
6. Select *Train classifier* to apply to machine learning algorithm to the training sample. A segmented output should display over the image. If there are any errors in segmentation, continue to draw over the appropriate parts and add them to the classes.



7. Select *Train classifier* again and repeat step 6 if there continues to be any errors or discrepancies in segmentation classification. Select the *Toggle overlay* to toggle between the original image and the segmentation overlay.

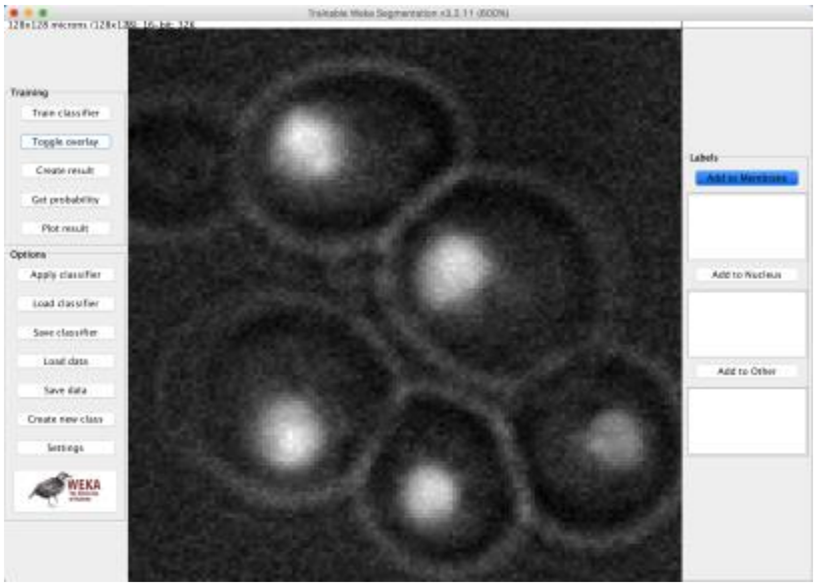
8. Once you have completed the classification for this test sample, select *Save data* to save the drawing data you have made (the classifications) to the classifier algorithm. Select *Save classifier* to save the settings and classification groups for this classifier. You should now have a *.arff* and a *.model* file saved somewhere in your directory – keep track of them.

9. Select *Create result* to view the image segmentation on its own and select *Get probability* to see individual class segmentations and probability maps. You can also select *Plot result* to obtain a threshold curve for the input classifications for the training sample.

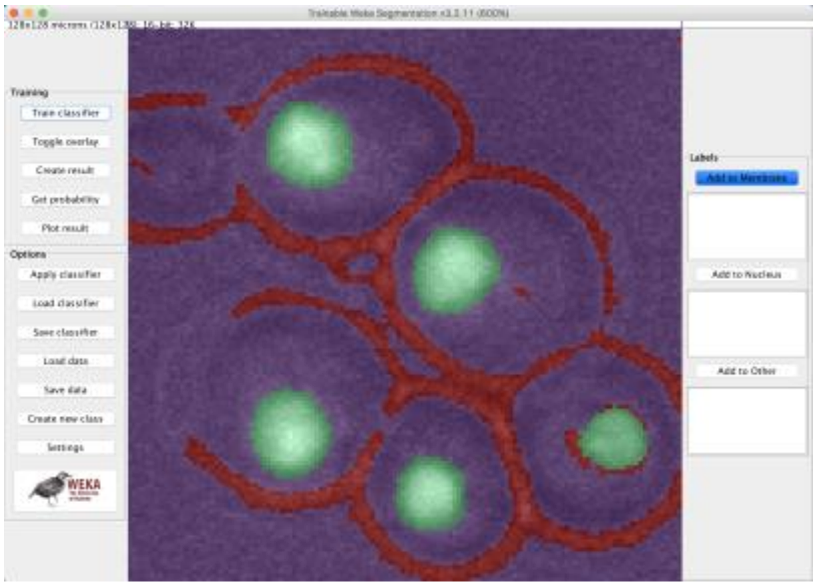


10. If only one image needs to be segmented, you can stop here. However, since this is a machine learning tool, you can make use of its abilities to predict the classification segmentation of new unseen images. However, before you can do that, a larger training sample is crucial. To add more images to the training samples of the classifier, follow these instructions below.

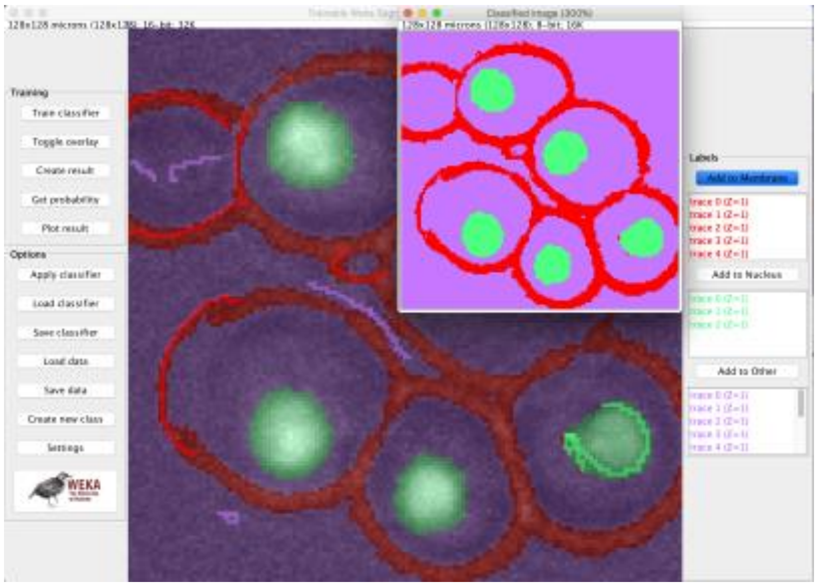
11. Close all menus and select the *Trainable Weka Segmentation* plugin again (step 1). Select the next image you want in your training sample. Note that the next image should preferably be taken with similar contrast settings and resolution. Select *Load classifier* and open the previously saved *.model* file. Select *Load data* and open the previously saved *.arff* file. The classes to the right should have loaded appropriately.



12. Select *Train classifier* to test the previously made classifier using the first image on this newly unseen image.



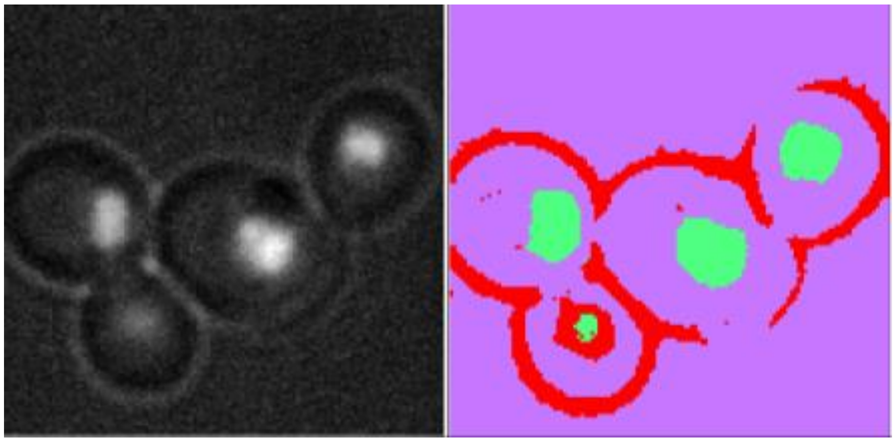
13. There will most likely be errors seen in this segmentation classification (toggle the overlay to see these errors). Select/draw over sections that the classifier has missed, classify them appropriately, and select *Train classifier* again. Repeat this drawing, classifying, and training of the sample until the correct classification of every feature in the image has been achieved. You can *Create result* if the classification needs to be verified.



14. To save this classification data along with the data from previous image(s), *Save data* and *Save classifier* after each image that you have completely classified (you can override previous saves).

15. To achieve better classifiers, load more images into the plugin, apply the classifier, fix/classify any errors, save, and repeat. You want as many images in the training sample to more accurately predict the segmentation classification of unseen images automatically.

16. With just the two classified images I loaded into the training sample. The trained algorithm predicted the following segmentations from this previously unseen image. As apparent in the result, there are clearly slight errors in classification, but it is also apparent that the classifier is doing is job. With much more images added to the training sample, an exponentially improved prediction can be made. With such outputs from the algorithm, post processing can be applied to further segment desired features and morphology segmentation can also be implemented.



References:

[1] http://imagej.net/Trainable_Weka_Segmentation
[2] http://www.cs.waikato.ac.nz/ml/weka/
[3] http://www.gnu.org/licenses/gpl.txt