DeepImageJ: A user-friendly plugin to run deep learning models in ImageJ

Estibaliz Gómez-de-Mariscal[1,**], Carlos García-López-de-Haro[1,**], Laurène Donati[2], Michael Unser [2], Arrate Muñoz-Barrutia[1,*] and Daniel Sage[2,*]

[1]Bioengineering and Aerospace Engineering Department, Universidad Carlos III de Madrid, 28911 Leganés, and Instituto de Investigación Sanitaria Gregorio Marañón, 28007 Madrid, Spain
[2]Biomedical Imaging Group, École polytechnique fédérale de Lausanne (EPFL), Switzerland
[*]Corresponding authors mamunozb@ing.uc3m.es, daniel.sage@epfl.ch
[**]Equally contributed

Example of a complete image analysis pipeline

In the following lines, we describe a generic workflow for image analysis using DL models and DeepImageJ plugin (see Figure 1). We use a toy example in which two U-Net [1] models are trained to segment cells on 2D phase contrast microscopy images. The entire code was written in Python using the Keras library[1] and it was run in the Google Colaboratory environment, which supplies a free Tesla K80 GPU service. The code is freely distributed[2].

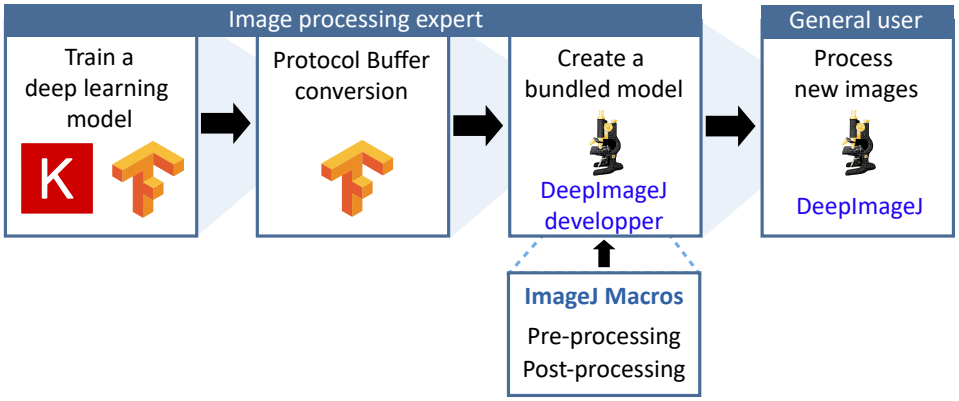

Figure 1: Proposed pipeline for image analysis using deep-learning models and DeepImageJ plugin.

The data used was made available by the Cell Tracking Challenge initiative (CTC)[3] [2, 3]. They are 2D phase contrast microscopy videos of HeLa cells cultured on a flat glass and Glioblastoma-astrocytoma U373 cells grown on a polyacrylamide substrate. We refer to the model trained on the HeLa cells as `U-Net HeLa segmentation` and on the U373 cells as `U-Net glioblastoma segmentation`. Only a small portion of the data was chosen to train and test the models (see Table 1 for details). Moreover, the image size was halved to shorten the computational time during training. The Keras `ImageDataGenerator` class was used to perform data augmentation with random rotations of ±40°, shifts, shear deformations, zooms, vertical and horizontal flips.

[1]https://github.com/zhixuhao/unet
[2]https://github.com/deepimagej/python4deepimagej/
[3]http://celltrackingchallenge.net/

Name of the bundled model	Images (training)	Images (test)	Loss (training)	Loss (test)	Accuracy (training)	Accuracy (test)	SEG (CTC)	Python runtime* [sec]
`U-Net HeLa segmentation`	8	9	0.245	0.231	0.900	0.905	0.830	4.33
`U-Net glioblastoma segmentation`	24	10	0.061	0.054	0.985	0.984	0.795	9.72

Table 1: Summary of U-Net training. *Run on an Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz, 32.0GB (RAM), 64-bit Operating System, x64 based processor machine with Windows 10 operating system.

Models were trained with the binary cross-entropy loss function, a learning rate of $1e^{-04}$ and a weight decay of $5e^{-07}$ during 10 epochs of 500 steps each. Altogether, it took 17 minutes to train each of them. The model finally chosen in both cases was the one that resulted in the lowest validation loss during training. The probability output maps from the inference were thresholded at 0.5 to get the final binary masks. The segmentation accuracy was assessed using the percentage of correct pixel assignments (accuracy) and the Jaccard index as computed by the code provided at the CTC web page (SEG) [2, 3]. Table 1 summarizes the segmentation accuracy results.

A short script along with the code translates `U-Net HeLa segmentation` and `U-Net glioblastoma segmentation` models from the widely used Keras format HDF5 (.hdf5) to the TensorFlow SavedModel one. Note that the script can be easily adapted to translate other models given in Keras format. Then, the generated models were easily converted to DeepImageJ bundled models using the provided builder module (DeepImageJ Build Bundled Model). Subsequently, the models were loaded in FIJI/ImageJ so they could be applied to the rest of the images. An optional post-processing macro to analyze all segmented objects was also implemented and it is provided with the rest of the code. Due to the large overlap between cells, the output binary mask from the `U-Net HeLa segmentation` model was first processed using a Watershed transform to split cellular clusters. Then, both models output an image of uniquely labelled cells that were further processed using the implemented user interface for object analysis. The described DeepImageJ workflow with the `U-Net HeLa segmentation` model is illustrated in panel B of Figure **??**.

Data availability statement

The web page: https://deepimagej.github.io/deepimagej/ provides free access to the plugin, along with the bundled models and user guide for image processing.

Acknowledgements

We would like to thank João Luis Soares Lopes, Rémy Pétremand and Halima Hannah Schede from École polytechnique fédérale de Lausanne (EPFL) for writing the complete Python pipeline and providing ready-to-use U-Net models. Daniel Wüstner from the University of Southern Denmark, Odense, provided membrane fluorescence images to test Noise2Void model. We would like also to thank Pedro M. Gordaliza, Ignacio Arganda-Carreras and Thomas Pengo for fruitful discussions.

This work is partially supported by the Spanish Ministry of Economy and Competitiveness (TEC2015-73064-EXP, TEC2016–78052-R) and by a 2017 Leonardo Grant for Researchers and Cultural Creators, BBVA Foundation. This work is part of the EPFL initiative "imaging@EPFL". We thanks the program "Short Term Scientific Missions" of NEUBIAS (network of European bioimage analysists). We also want to acknowledge the support of NVIDIA Corporation with the donation of the Titan X (Pascal) GPU card used for this research.

Author contributions

E.G.M. and C.G.L.H. contributed to the design of the experimental framework, reviewed, trained and exported existing image processing methods. C.G.L.H. and D.S. developed and implemented the plugin and worked on the supporting documentation with input from the rest of the authors. E.G.M. and L.D. wrote the manuscript with help from A.M.B. and D. S.. E.G.M., A.M.B. and D.S. created the web page dedicated to the plugin. All the authors

contributed to the conception of the study, the design of the experimental framework and took part in the literature review. All authors revised the manuscript.

Competing interests

The authors declare that they have no competing interests.

References

[1] Thorsten Falk, Dominic Mai, Robert Bensch, Özgün Çiçek, Ahmed Abdulkadir, Yassine Marrakchi, Anton Böhm, Jan Deubner, Zoe Jäckel, Katharina Seiwald, Alexander Dovzhenko, Olaf Tietz, Cristina Dal Bosco, Sean Walsh, Deniz Saltukoglu, Tuan Leng Tay, Marco Prinz, Klaus Palme, Matias Simons, Ilka Diester, Thomas Brox, and Olaf Ronneberger. U-Net: deep learning for cell counting, detection, and morphometry. *Nat. Methods*, 16(1):67–70, jan 2019.

[2] Martin Maška, Vladimír Ulman, David Svoboda, Pavel Matula, Petr Matula, Cristina Ederra, Ainhoa Urbiola, Tomás España, Subramanian Venkatesan, Deepak MW Balak, et al. A benchmark for comparison of cell tracking algorithms. *Bioinformatics*, 30(11):1609–1617, 2014.

[3] Vladimír Ulman, Martin Maška, Klas EG Magnusson, Olaf Ronneberger, Carsten Haubold, Nathalie Harder, Pavel Matula, Petr Matula, David Svoboda, Miroslav Radojevic, et al. An objective comparison of cell-tracking algorithms. *Nature methods*, 14(12):1141, 2017.