



A Guide for Open-Source Deep Learning Tools in Bioimaging

Prateek Verma, Hao Van, Xintao Wu

Aug 14, 2024

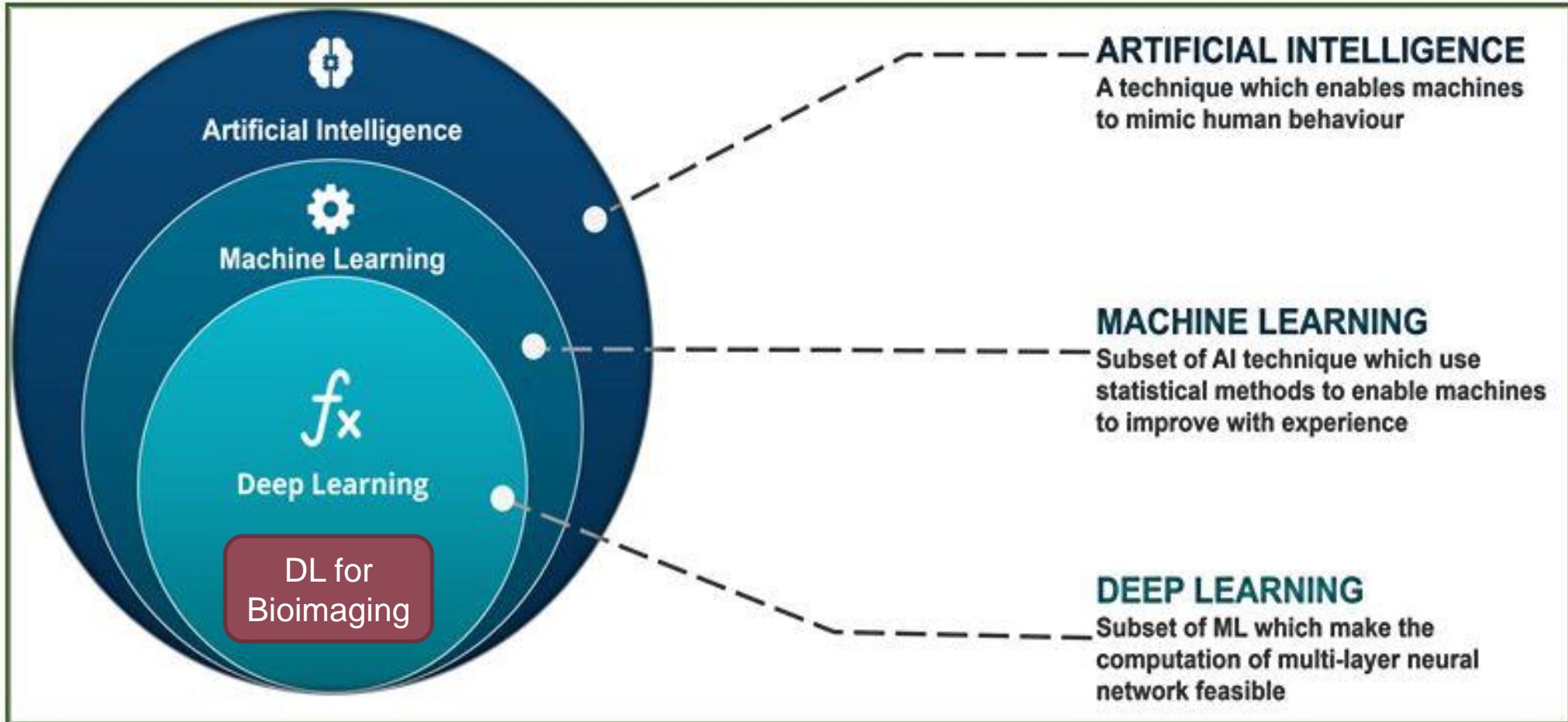
Arkansas Integrative Metabolic Research Center

Outline

- Introduction to deep learning
 - [Exercise 1](#)
- Deep learning tools/platforms in bioimaging
 - Traditional ML tools: ImageJ, ImageJ2 and Fiji
 - DL tools: BioImage Model Zoo, DeepImageJ
 - [Exercise 2](#)
- Overview of DL algorithms in bioimaging
 - Segmentation ([Exercise 3](#))
 - Denoising and image restoration
 - Super-resolution microscopy
 - Object detection
 - Image-to-image translation
- More tools
 - ZeroCostDL4Mic ([Exercise 4](#))
 - CSBDeep, etc.

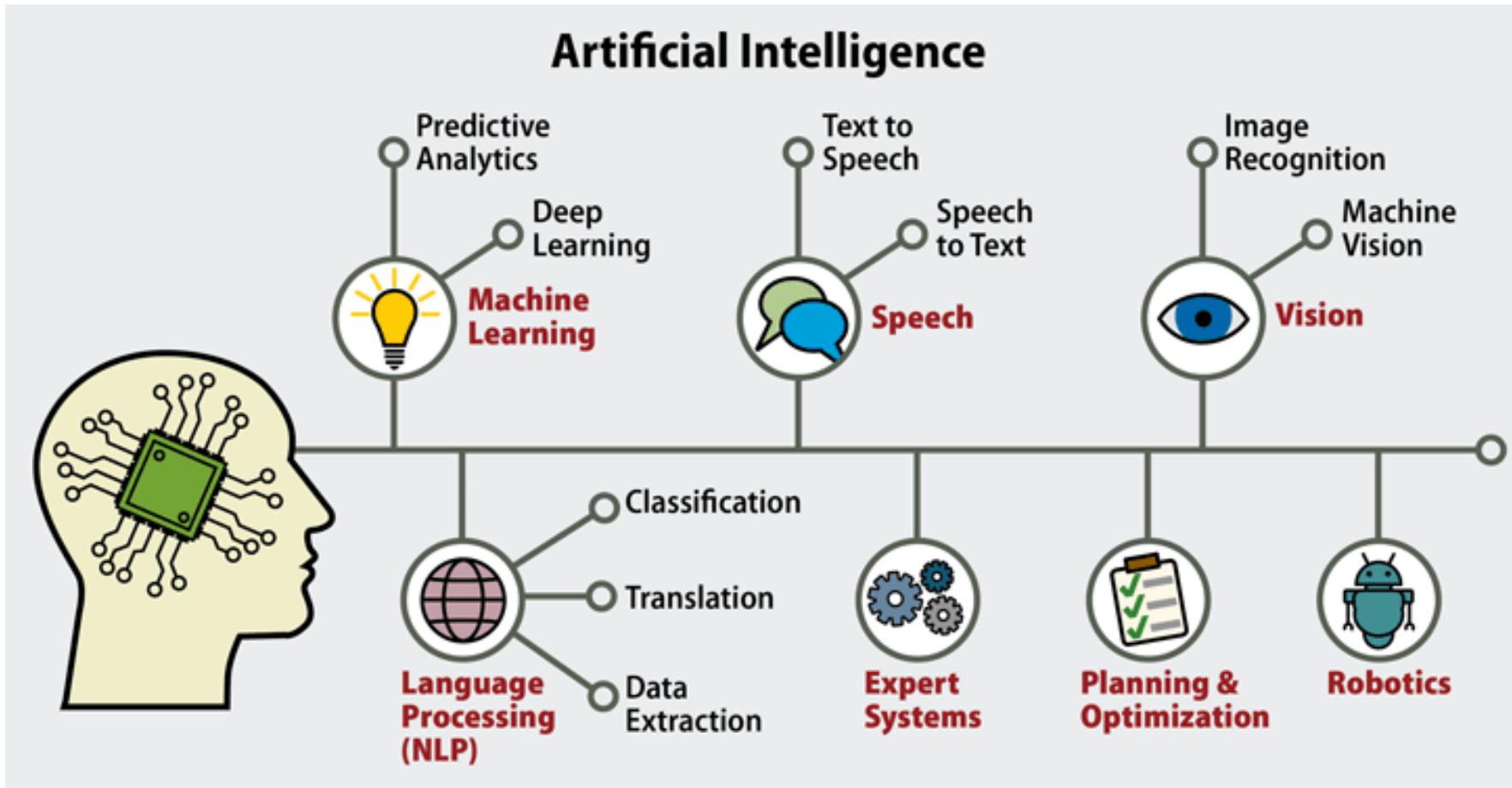


AI vs. ML vs. DL



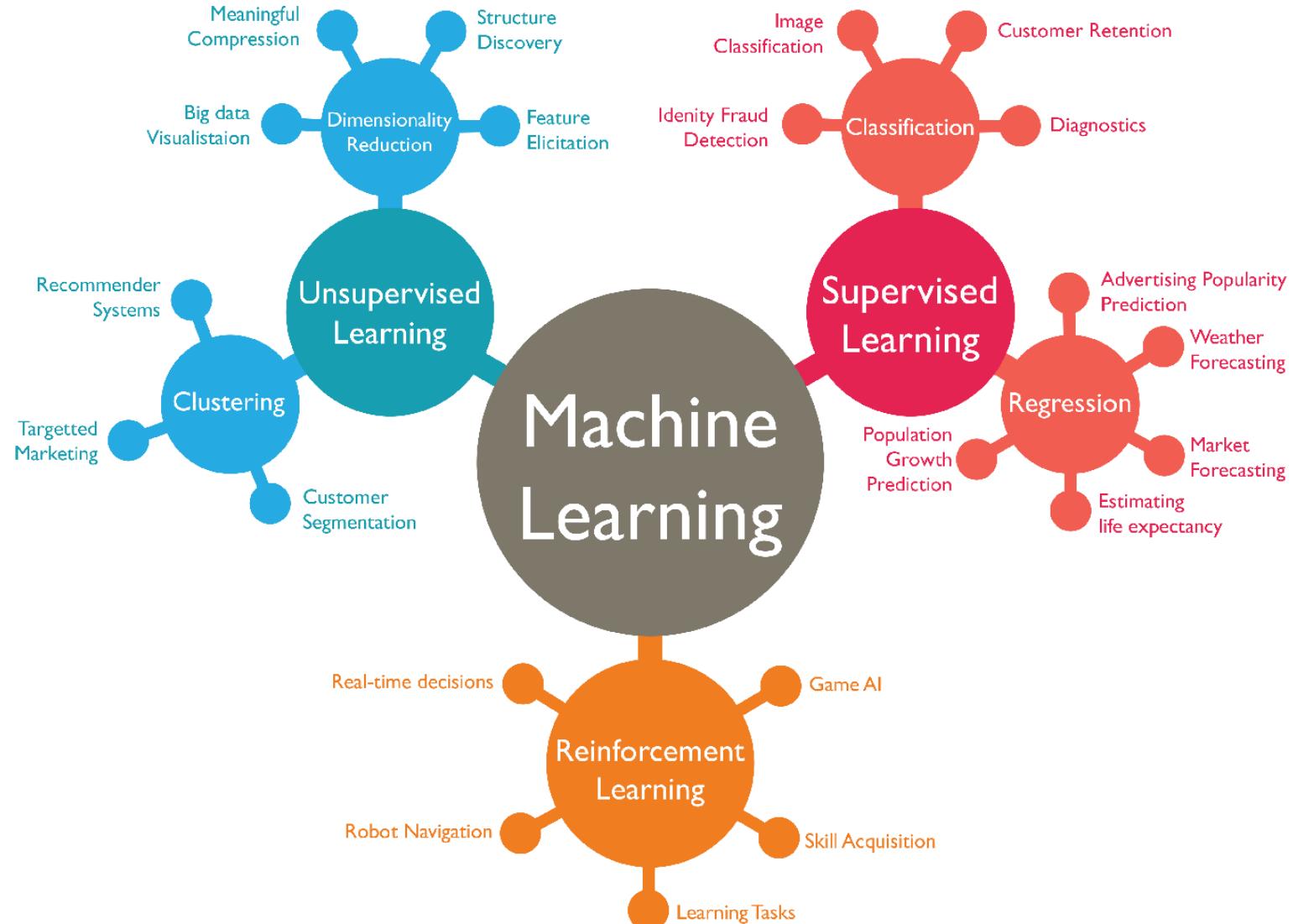


Artificial Intelligence



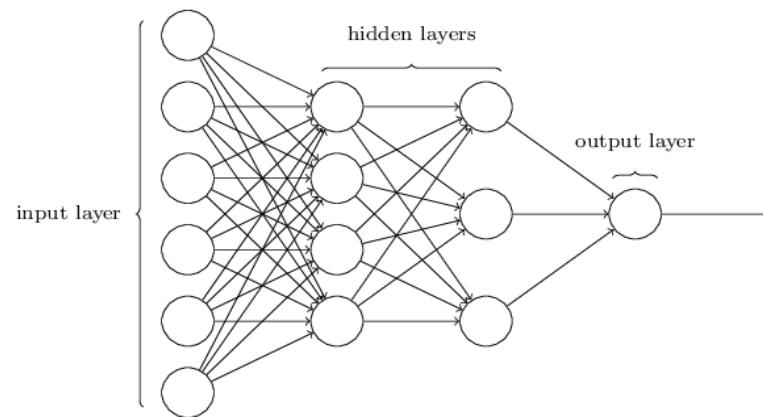


Machine Learning

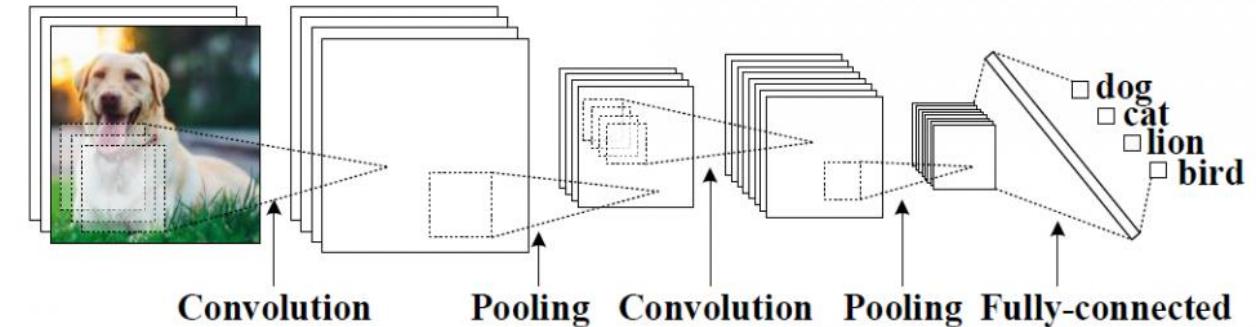


Basic Deep Learning Structures

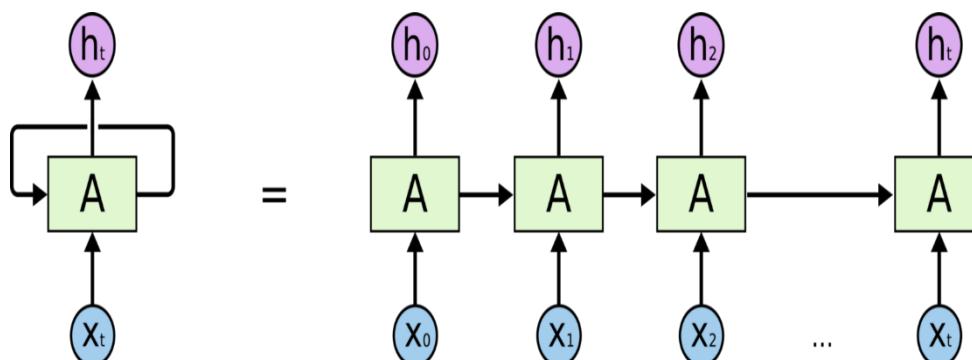
Feedforward Neural Network



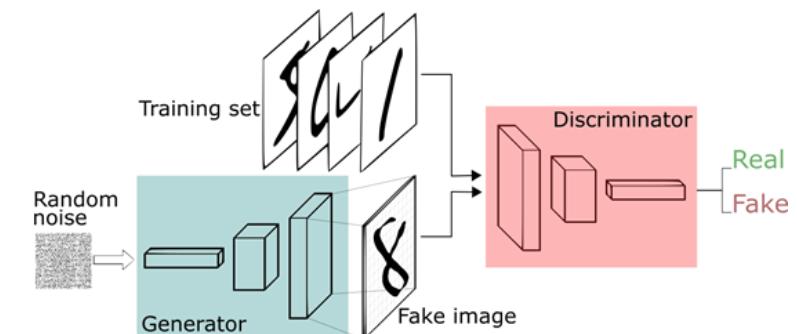
Convolutional Neural Network



Recurrent Neural Network

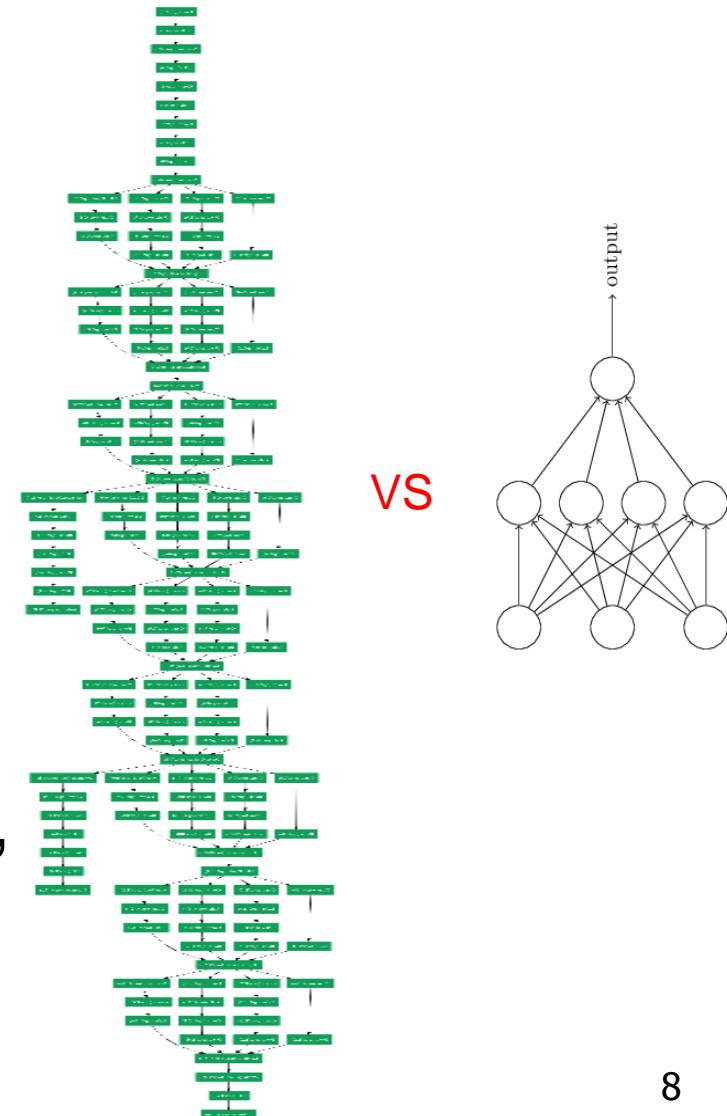


Generative Adversarial Network



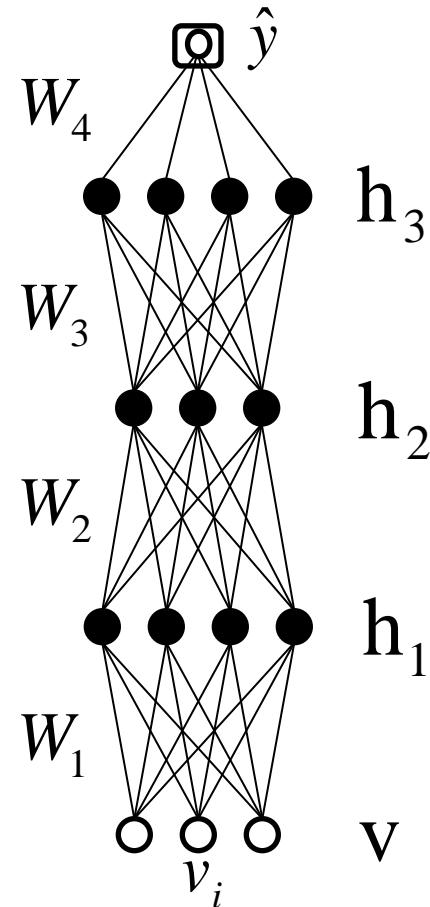
Why does DL work so well?

- lots of data (Big Data)
- Very flexible models
- GPGPU (powerful machines)
- Advanced algorithms for optimization, activation, regularization
- Huge research society (vision, speech, NLP, bioimaging, etc.)

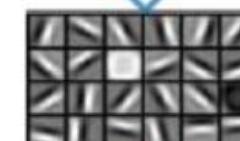




Learning of Representations



3rd Layer
“Objects”



1st Layer
“Edges”



Pixels
[Andrew Ng]



What's this workshop for

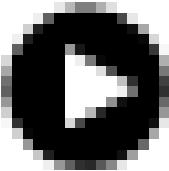
- ✓ how deep-learning algorithms work
- ✓ a variety of deep-learning tasks

- ✗ learn Python
- ✗ learn the inner workings of deep-learning



Before we begin the exercises

How to run a Jupyter Notebook?



we just read the instructions
And press this button

Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
✓ 0s   seconds_in_a_day = 24 * 60 * 60  
    seconds_in_a_day
```

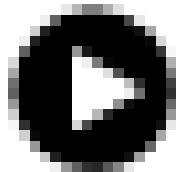
```
→ 86400
```



Before we begin the exercises

So, don't worry if you don't understand the code.

- familiarize yourself with different tasks and algorithms
- use this code base for future reference

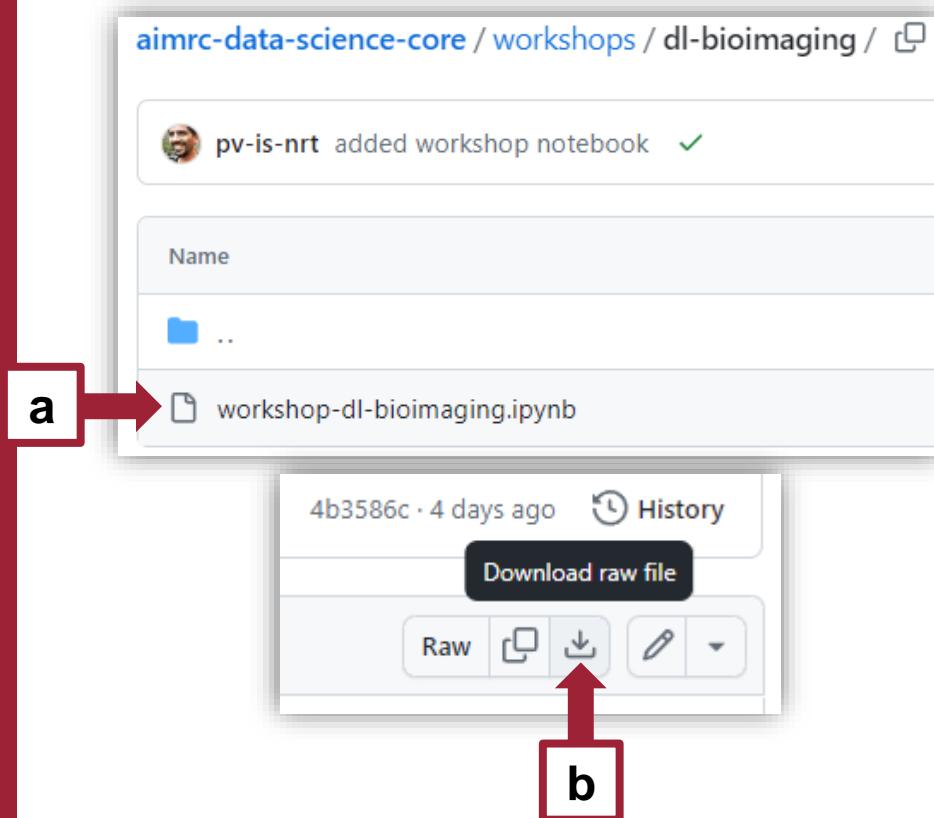


Just keep pressing the play button!

Exercise 1 – Classification

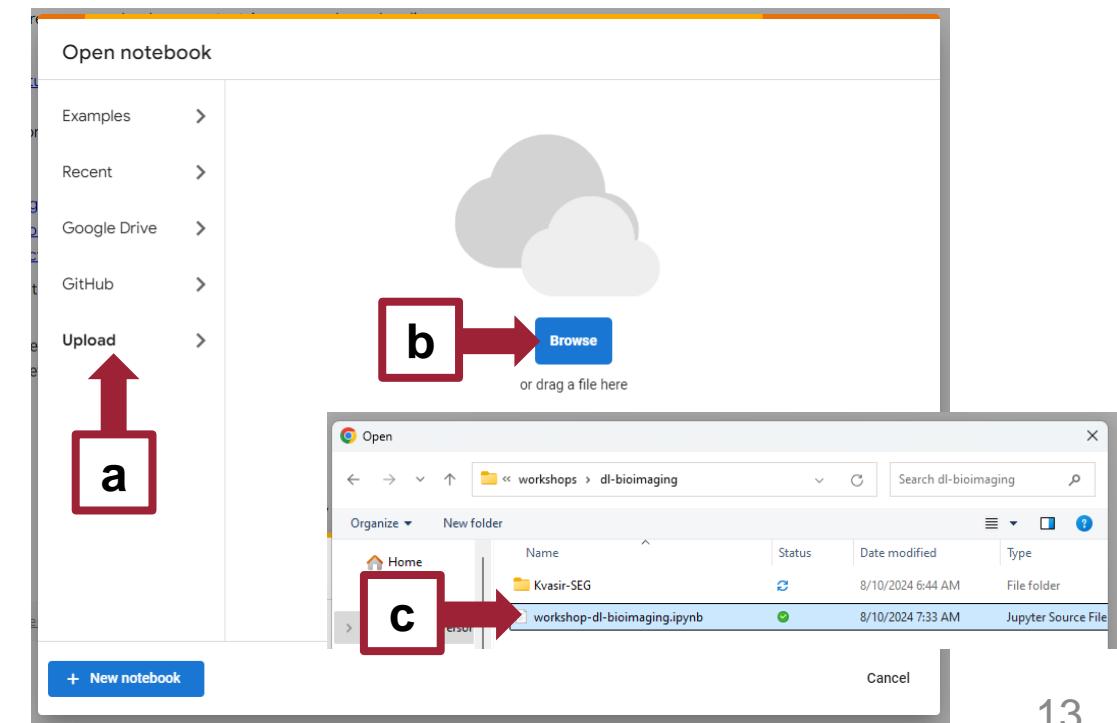
1. Download the Jupyter Notebook

- AIMRC [GitHub](#) > workshops > dl-bioimaging
- Click on “Download raw file” button in the upper right corner.



2. Upload the Notebook to Google Colab

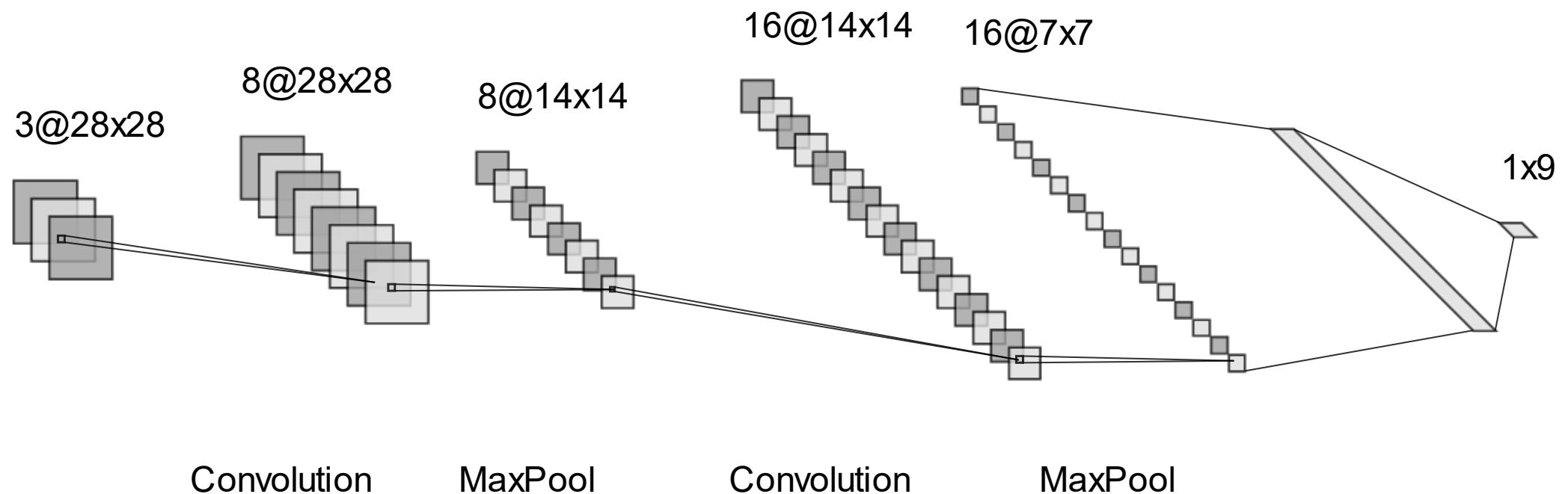
- Open [Google Colab](#) in your browser.
- Upload > Browse. Upload the Notebook file (.ipynb) you downloaded in the previous step.
- Follow steps written in the Notebook.





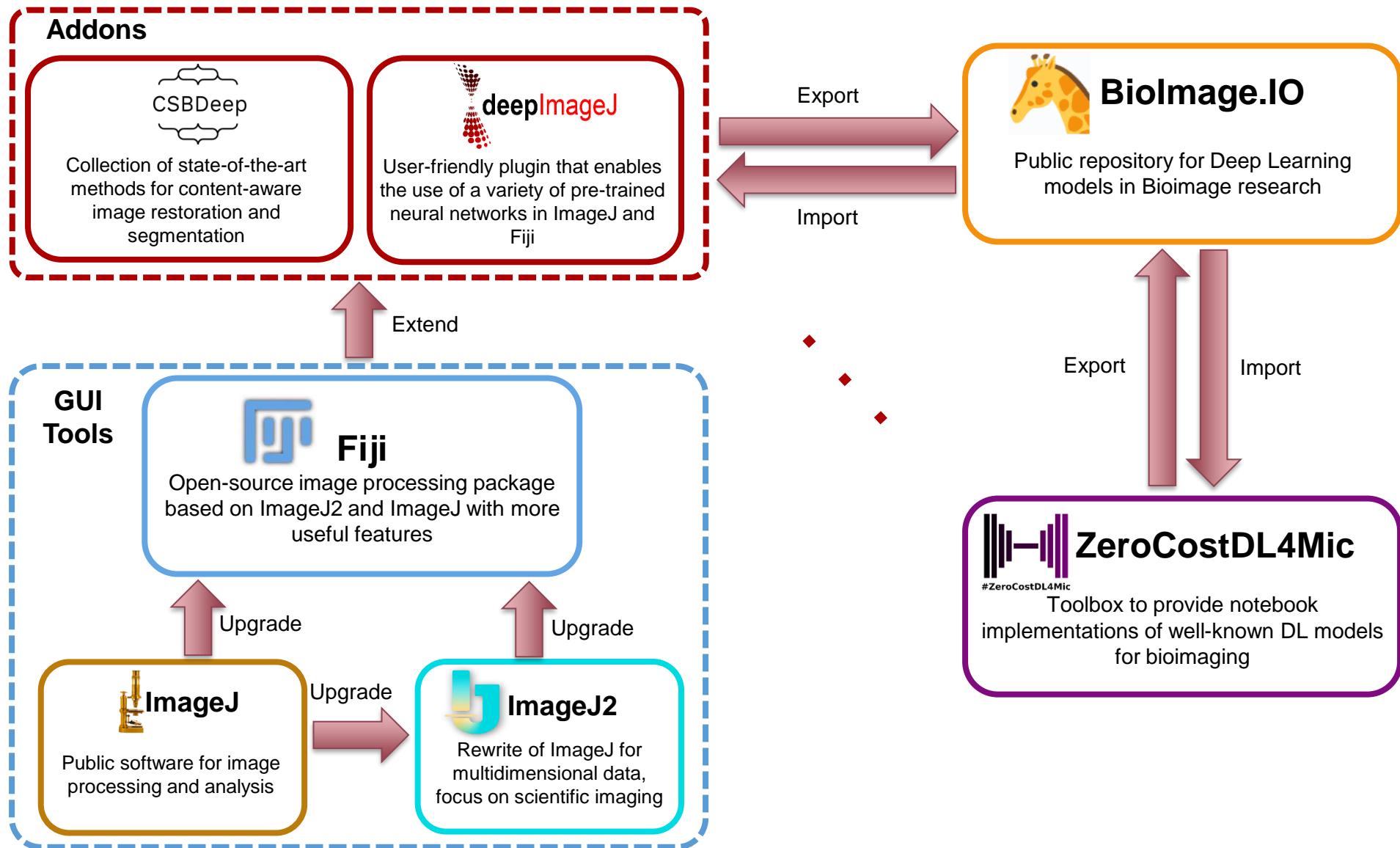
UNIVERSITY OF
ARKANSAS

Exercise 1 – Classification





Software Tools



ImageJ, ImageJ2 and Fiji

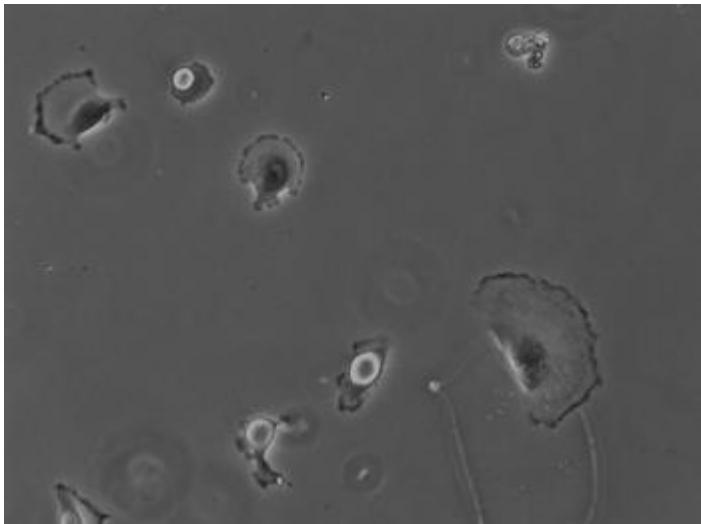
- **ImageJ** is a public software for processing and analyzing images, developed in Java.
 - **ImageJ2** is a new version of ImageJ1 that focuses on multidimensional image data.
 - **Fiji** is another upgrade that includes both ImageJ1, ImageJ2, and more useful features.
-  **Fiji is the best option to use all features from ImageJ1 and ImageJ2.**

ImageJ, ImageJ2 and Fiji

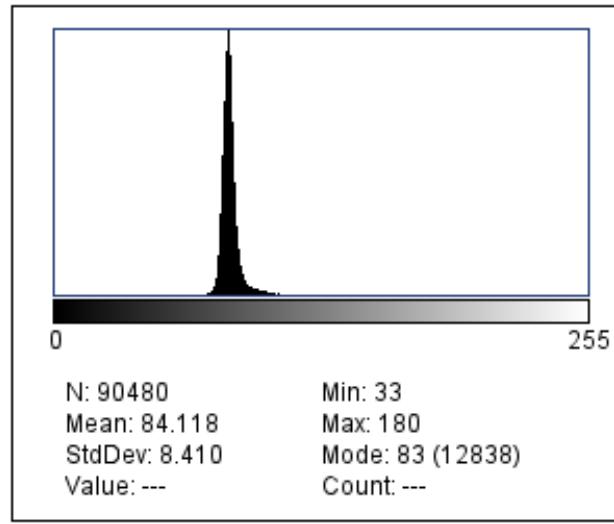
- Fiji/ImageJ provide
 - User interface with useful functions.
 - Tasks: image processing, colocalization, deconvolution, registration, segmentation, tracking, visualization and more.
 - Plugins: extended plugins for more image analyzing tasks, i.e. DeepImageJ is an extension in Fiji/ImageJ.
 - Scripts and macros: reproducible workflow.
 - Community: very active forum for questions and issues.
- Easy-to-install and easy-to-use (we'll see!)
- Good documentation



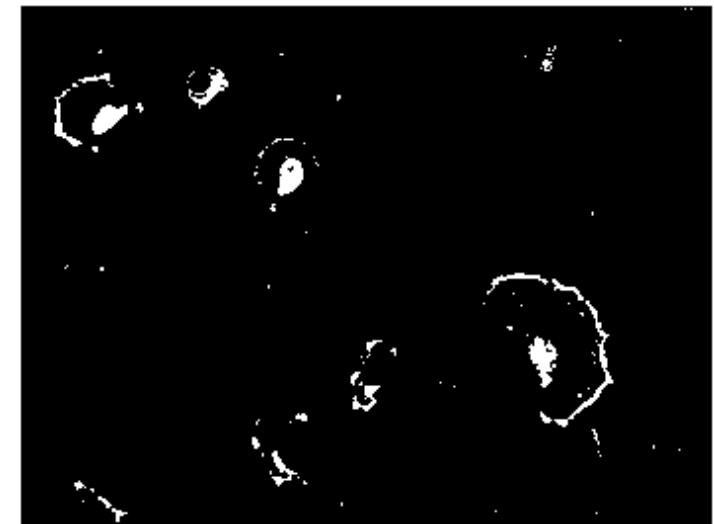
Examples



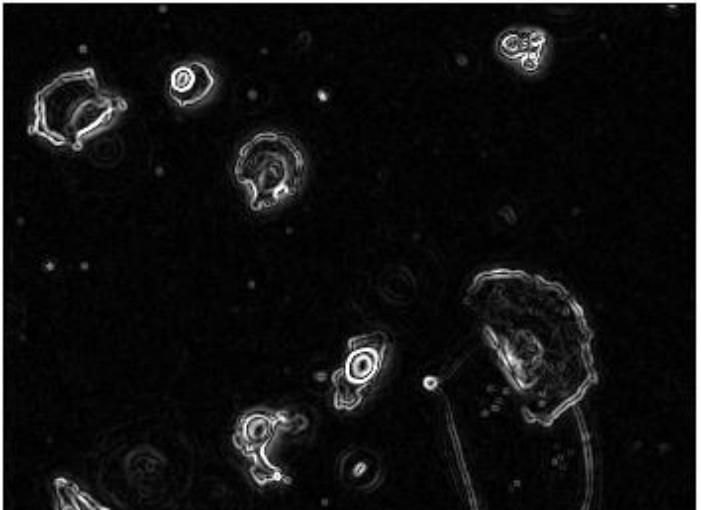
Original image (cell in glioblastoma phase)



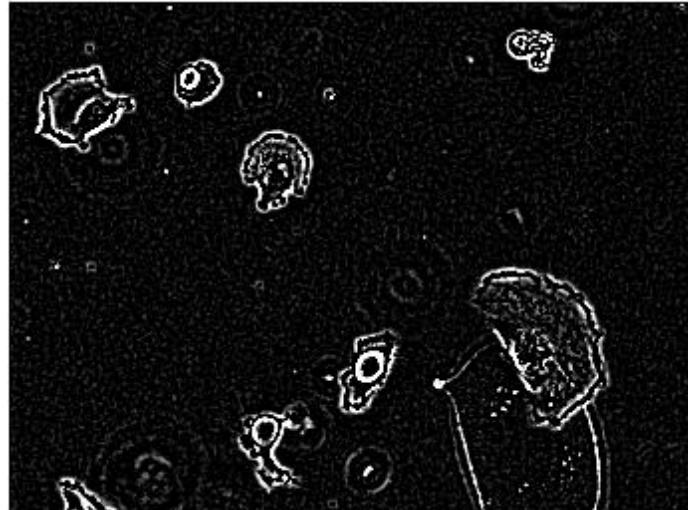
Histogram



Binary image (threshold 73)



Edges detector (Sobel's method)



Convolution operator



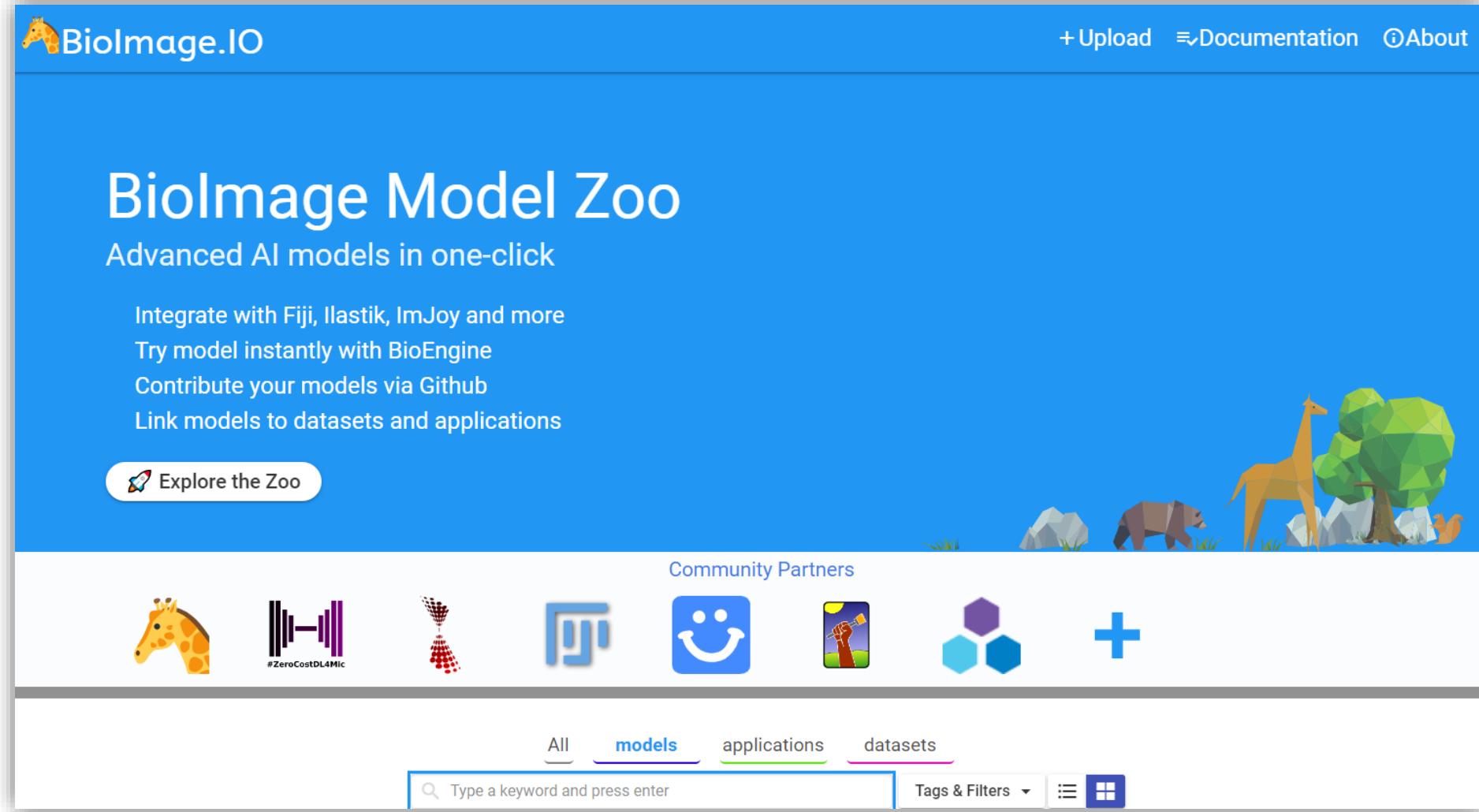
$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 24 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

Kernel for the convolution operator

DeepImageJ

- A plugin in Fiji/ImageJ that supports pre-trained DL models for image data.
- An easy way for biomedical researchers to try and explore DL models without any prerequisites for programming skills or DL knowledge.
- Currently supports
 - TensorFlow 1 & 2, PyTorch
 - GPU support (for Linux machine only)
- Pros
 - Deep Learning models with GUI interactions
 - Can incorporate models from BioImage Model Zoo or trained by users.
- Cons
 - ~~One test image at a time~~

Biolimage Model Zoo



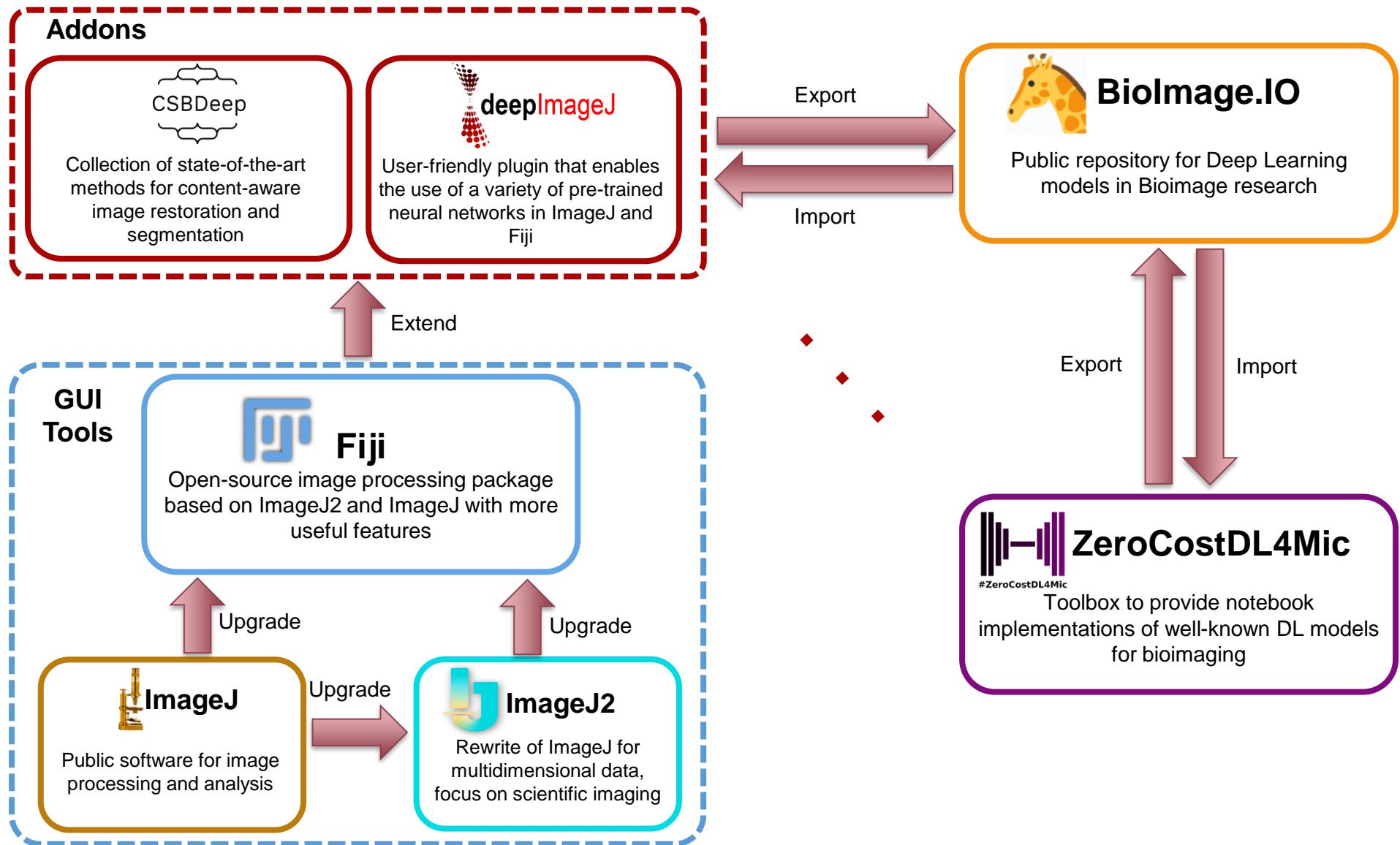
The screenshot shows the homepage of the Biolimage Model Zoo. At the top left is the logo "Biolimage.IO" with a giraffe icon. At the top right are links for "+ Upload", "Documentation", and "About". The main title "Biolimage Model Zoo" is prominently displayed in large white font. Below it is the subtitle "Advanced AI models in one-click". A list of features follows: "Integrate with Fiji, Ilastik, ImJoy and more", "Try model instantly with BioEngine", "Contribute your models via Github", and "Link models to datasets and applications". A "Explore the Zoo" button with a rocket icon is visible. The background features a stylized illustration of a giraffe, a tree, and other animals in a savanna-like setting. Below the main banner is a "Community Partners" section with icons for various organizations. At the bottom is a navigation bar with tabs for "All", "models" (which is underlined in blue), "applications" (underlined in green), and "datasets". A search bar and a "Tags & Filters" dropdown are also present.

BioImage Model Zoo

- A public repository for DL models in Bioimage research
- Contribution from community
 - ImageJ/Fiji
 - DeepImageJ
 - ZeroCostDL4Mic
 - ...
- BioImage Model Zoo provides
 - Pre-trained model
 - Training data
 - Description for each model: related paper, model's architecture, ...

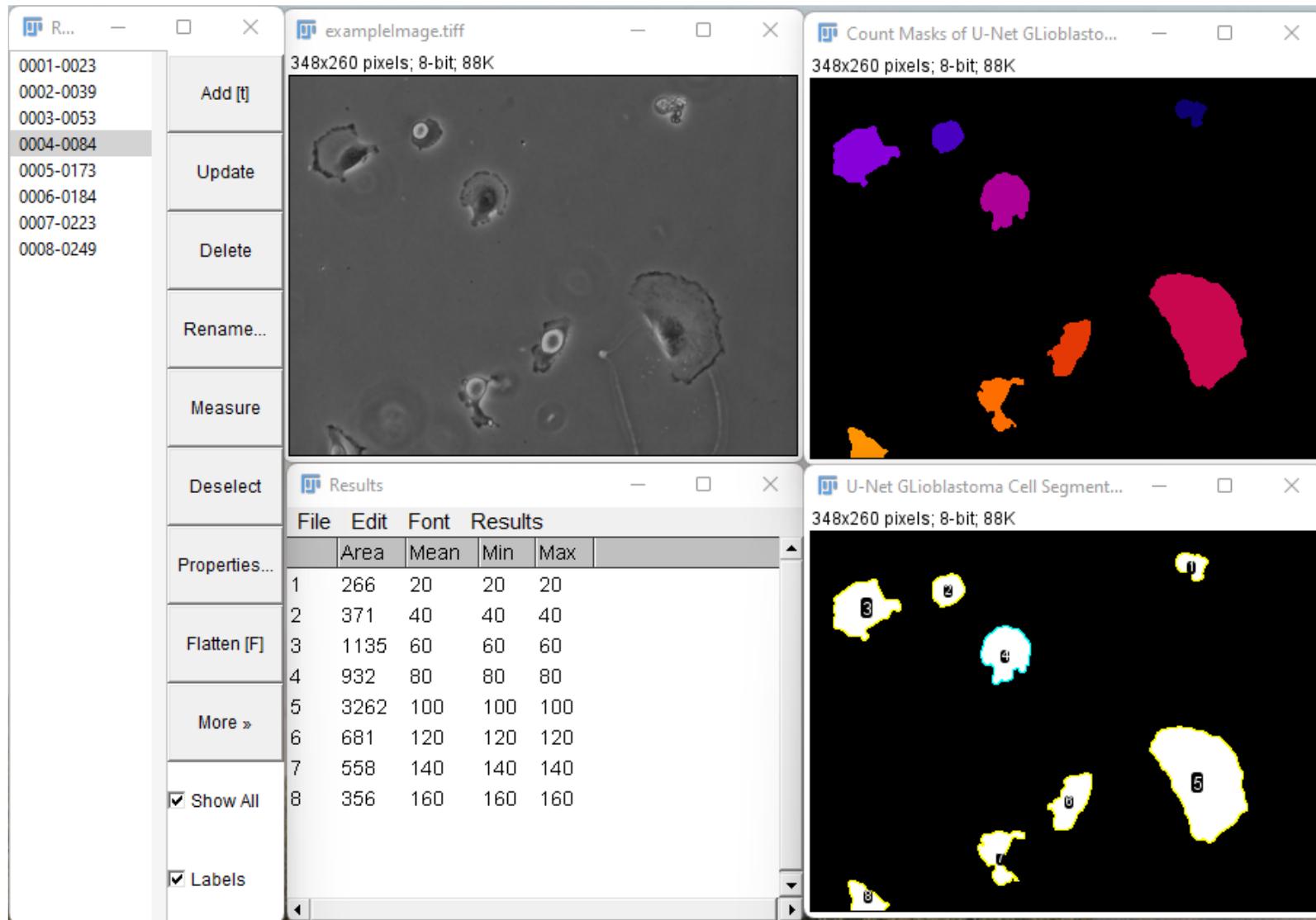


Software Tools





DeepImageJ



Model: Glioblastoma Phase Contrast Cell Segmentation (2D U-NET)

Pre-trained from BioImage Model Zoo Dataset

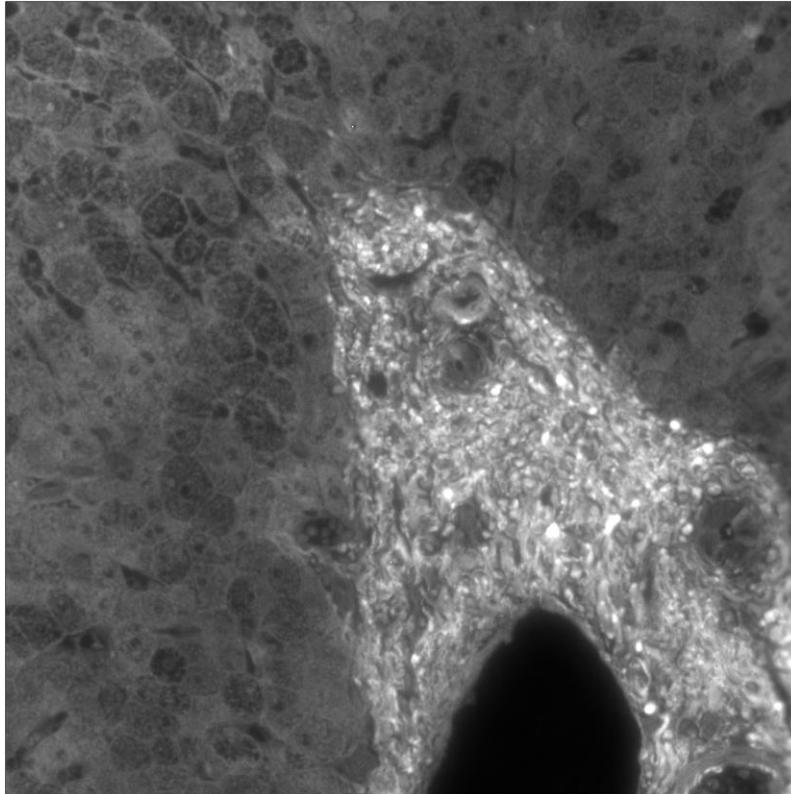
- Microscopy modality: 2D Phase contrast
- 24 images for training, 10 images for testing
- Cell type: Glioblastoma-astrocytoma (U373)
- Source: Cell tracking challenge

Training procedure

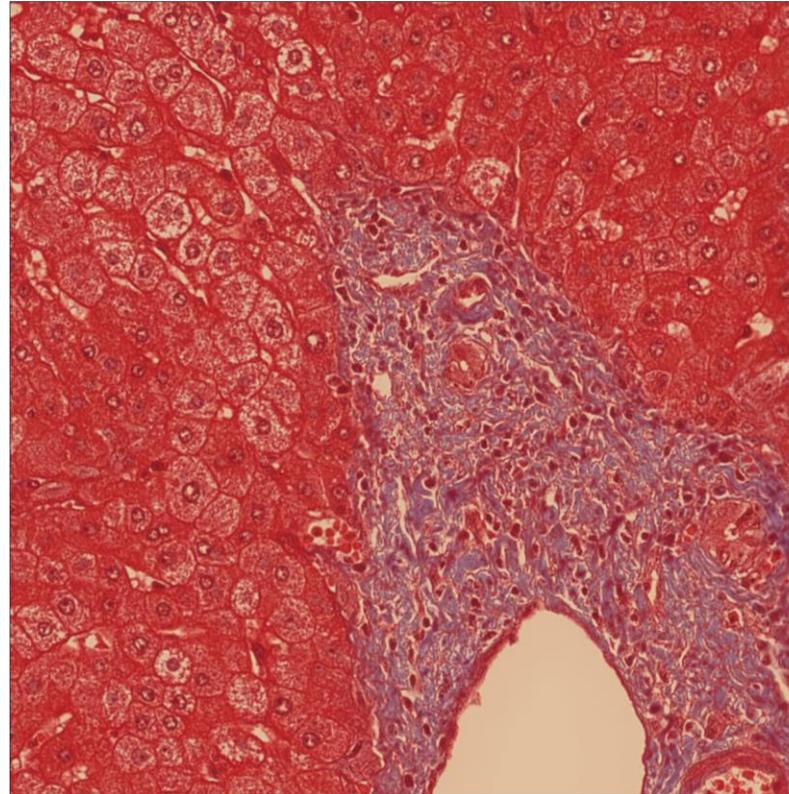
- 10 epochs, 500 steps per epoch
- Learning rate: 0.0001



DeepImageJ



Original Image



Output Image

Model: [Masson's Trichrome Virtual Staining \(GAN\)](#)

Pre-trained from BioImage Model Zoo

Training dataset: N/A.
Dataset is collected by authors.

Training procedure

- 3 epochs
- Learning rate: 0.0001 (generator) and 0.00001 (discriminator)

Exercise 2 - FIJI

1. Download Fiji here:

<https://imagej.net/software/fiji/downloads>



Fiji Downloads

Fiji is a distribution of ImageJ which includes many useful plugins contributed by the community.

~ Download Fiji for your OS ~

Windows 64-bit [imagej.net \(USA\)](#), [micron.ox.ac.uk \(European mirror\)](#)

Windows 32-bit [imagej.net \(USA\)](#), [micron.ox.ac.uk \(European mirror\)](#)

macOS (x86_64) [imagej.net \(USA\)](#), [micron.ox.ac.uk \(European mirror\)](#)

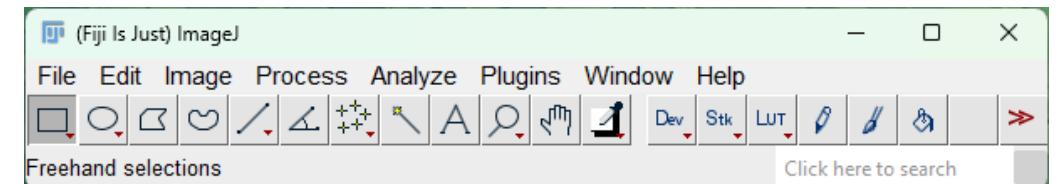
Linux (64-bit) [imagej.net \(USA\)](#), [micron.ox.ac.uk \(European mirror\)](#)

No JRE [imagej.net \(USA\)](#), [micron.ox.ac.uk \(European mirror\)](#)

2. Extract Fiji

Name	Date modified
Contents	8/10/2024 7:55 AM
engines	8/10/2024 8:00 AM
images	8/10/2024 7:56 AM
jars	8/11/2024 10:53 AM
java	8/10/2024 7:56 AM
lib	8/10/2024 7:56 AM
licenses	8/10/2024 7:56 AM
luts	8/10/2024 7:56 AM
macros	8/10/2024 7:56 AM
models	8/11/2024 10:48 AM
plugins	8/10/2024 7:58 AM
retro	8/10/2024 7:56 AM
scripts	8/10/2024 7:58 AM
.checksums	8/11/2024 10:52 AM
db.xml.gz	8/11/2024 10:53 AM
ImageJ-win64.exe	8/10/2024 7:55 AM
README.md	8/10/2024 7:55 AM
WELCOME.md	8/10/2024 7:55 AM

3. Run Fiji

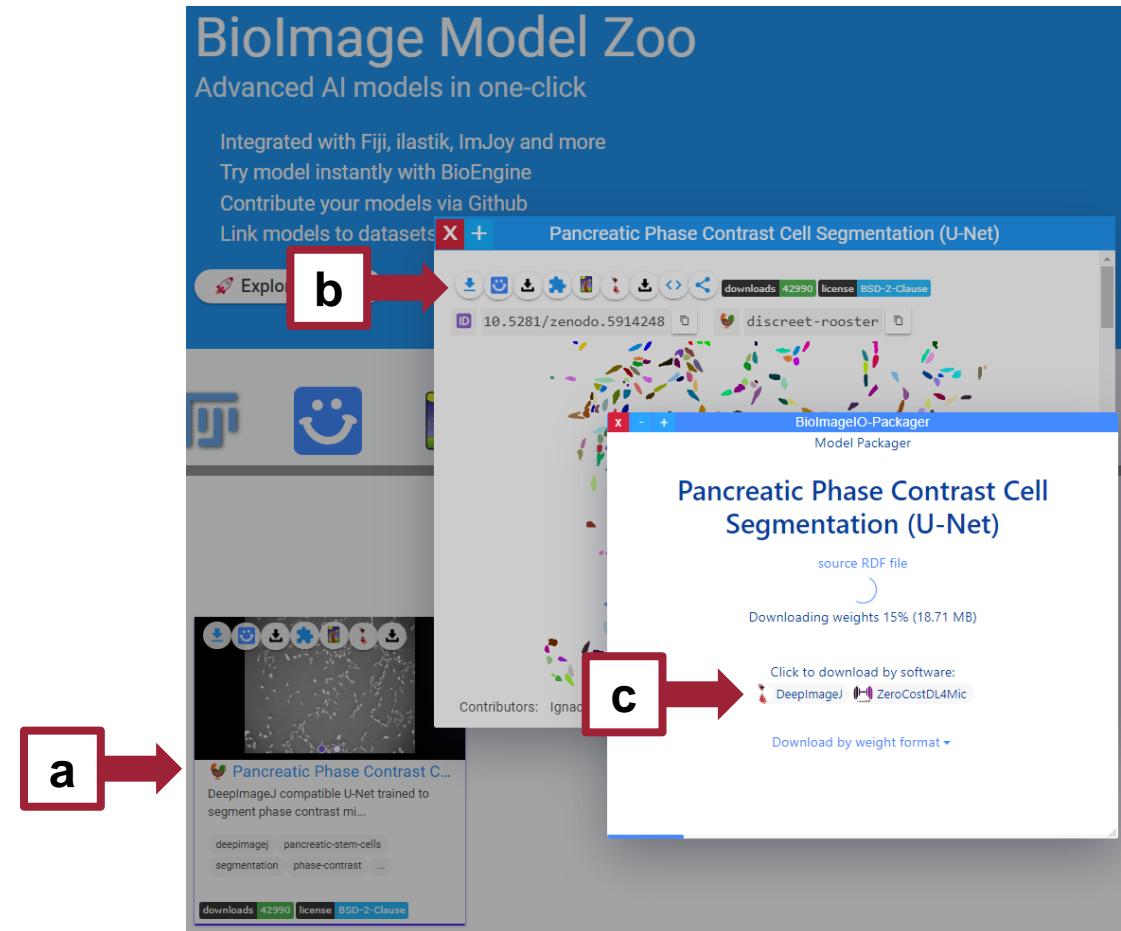




Exercise 2 - FIJI

4. Download a pretrained model from Bioimage.io

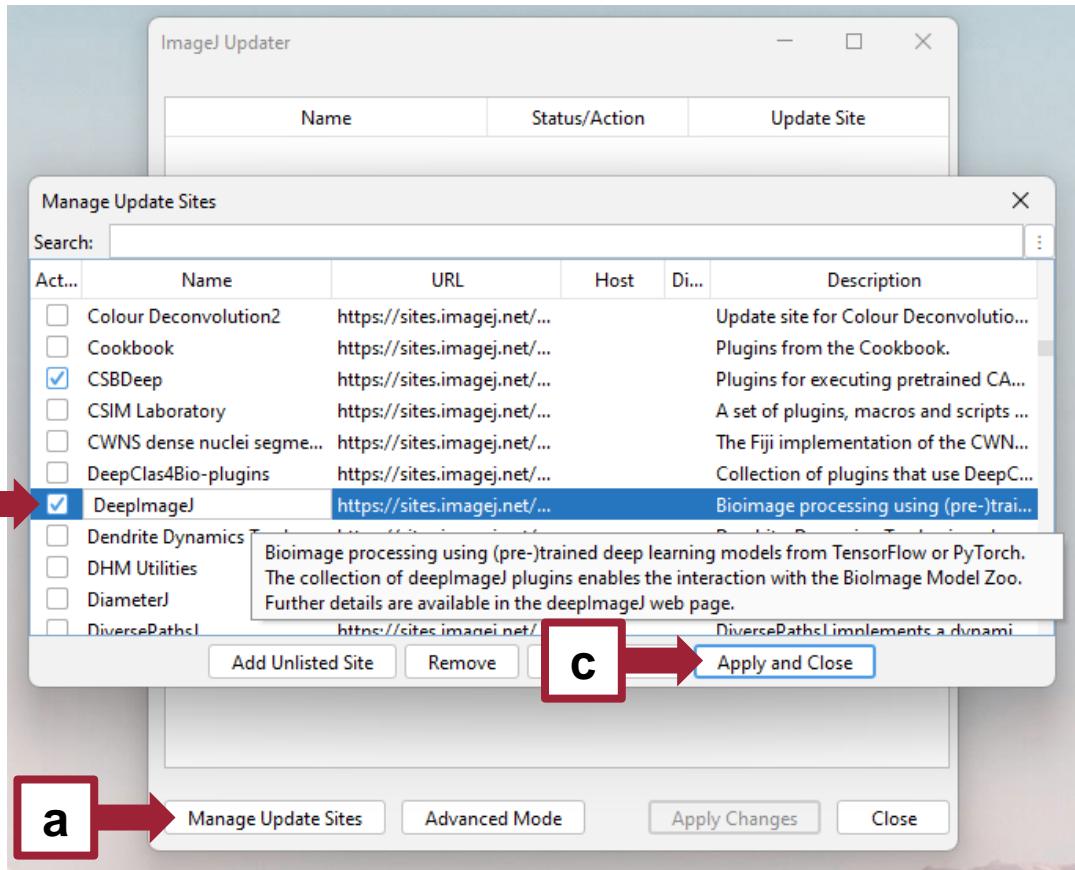
Download the “Pancreatic Phase...” model zip file for DeepImageJ.



Exercise 2 - FIJI

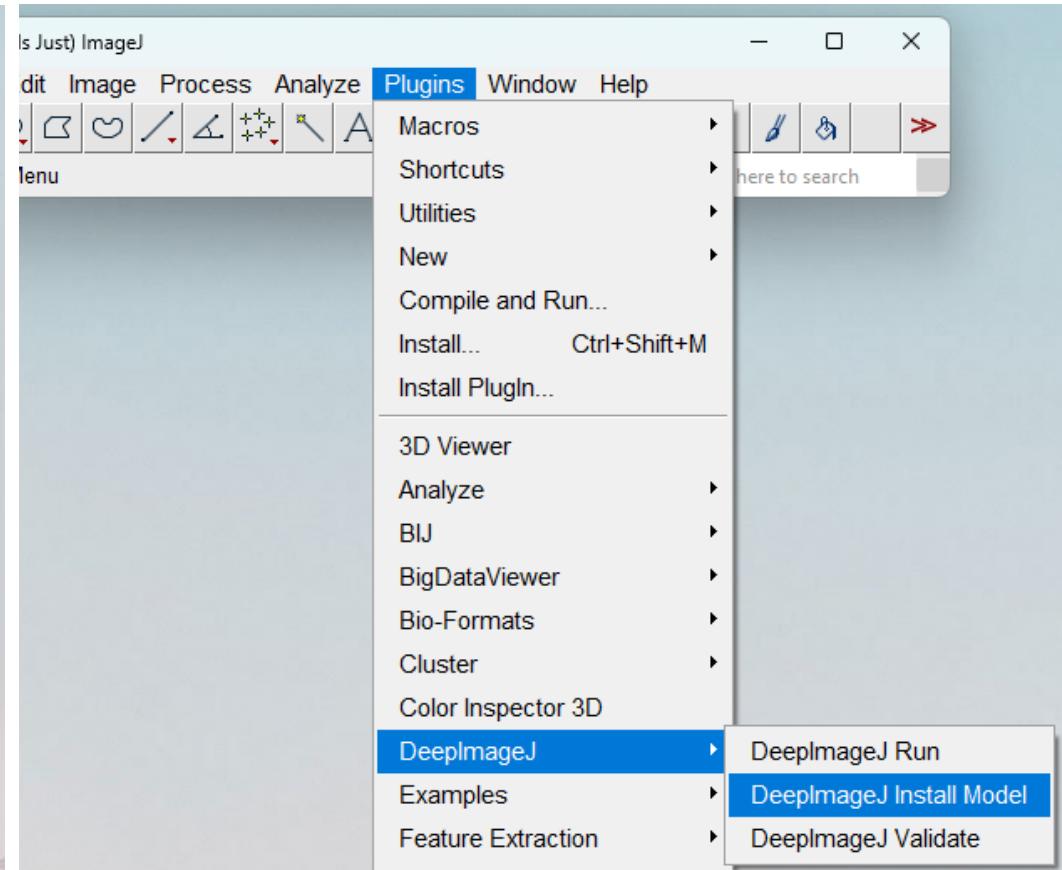
5. Install DeepImageJ Plugin

- Help > Update. Click on Manage Update Sites.
- Check DeepImageJ and click on Apply and Close.
- Restart Fiji.



6. Install a model in DeepImageJ

- Plugins > DeepImageJ > Install Model

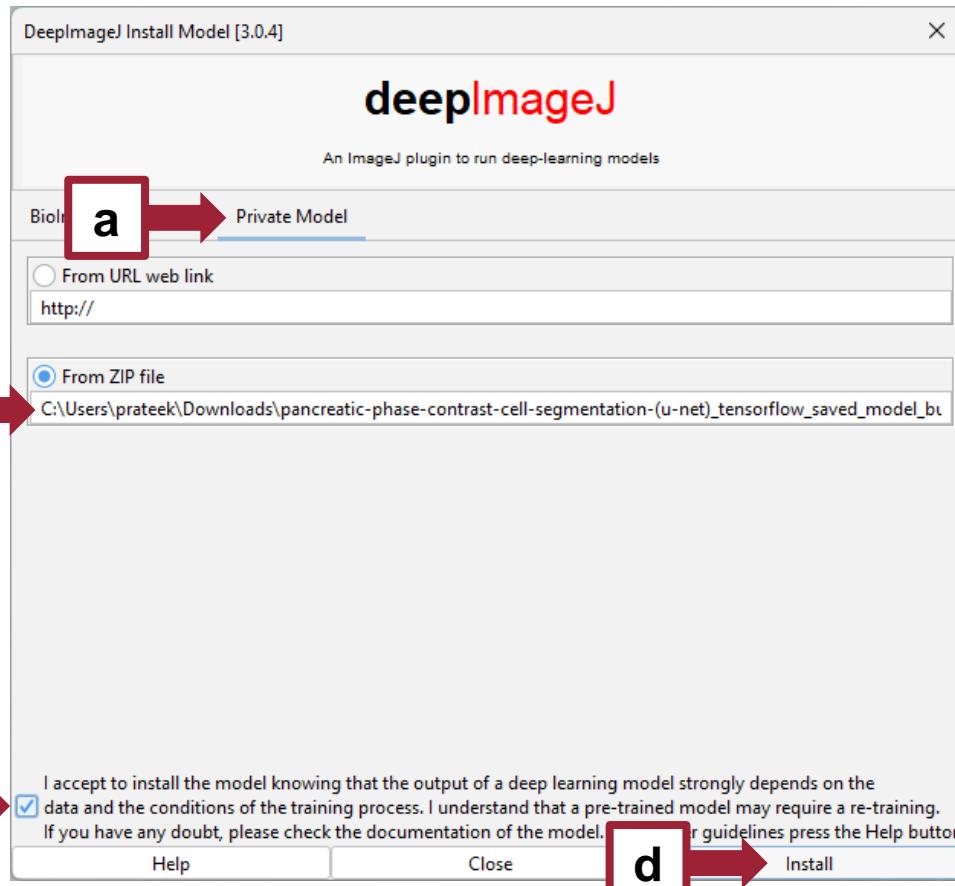




Exercise 2 - FIJI

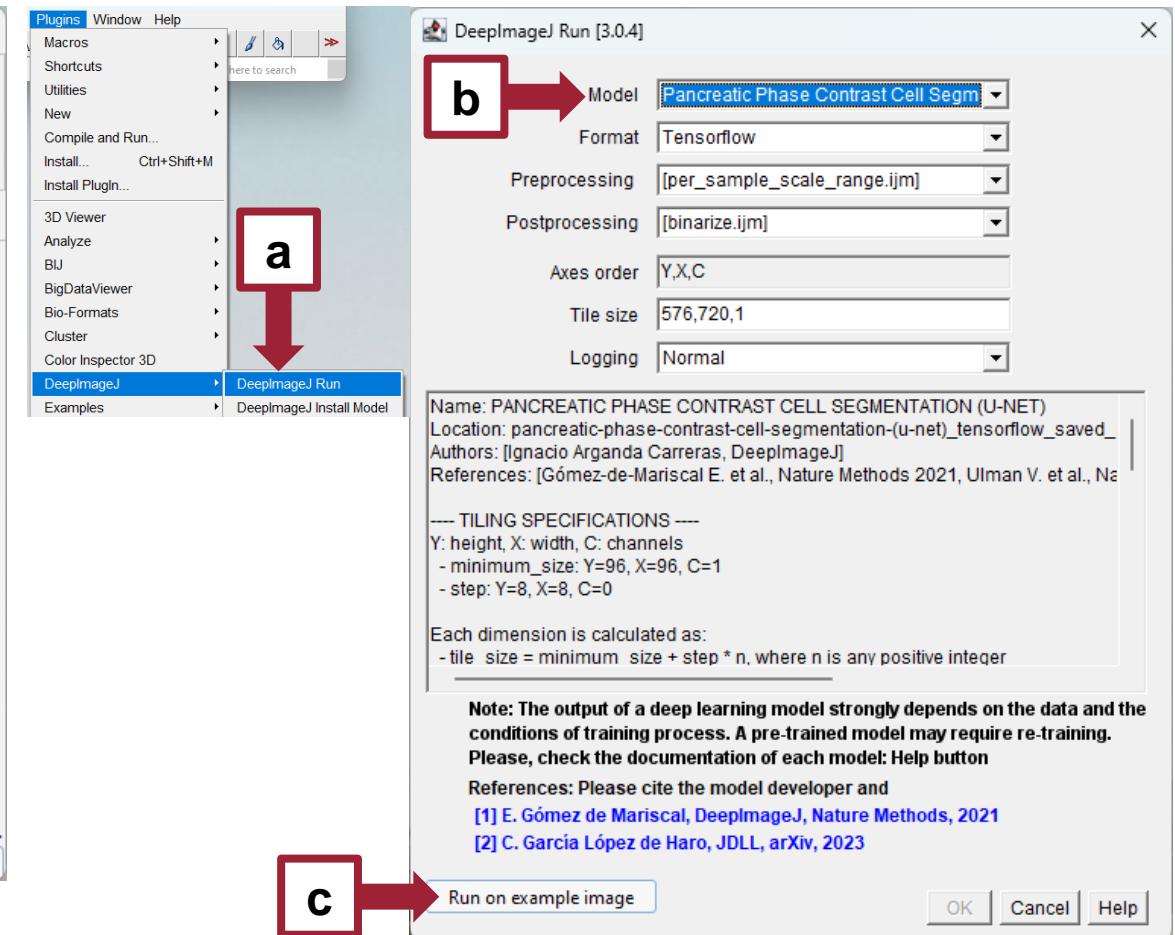
6. Install a model in DeepImageJ (contd.)

- Select “Private Model” tab, “From ZIP file”, enter path
- Check “I accept”, click Install. Restart Fiji.



7. Run model on example image

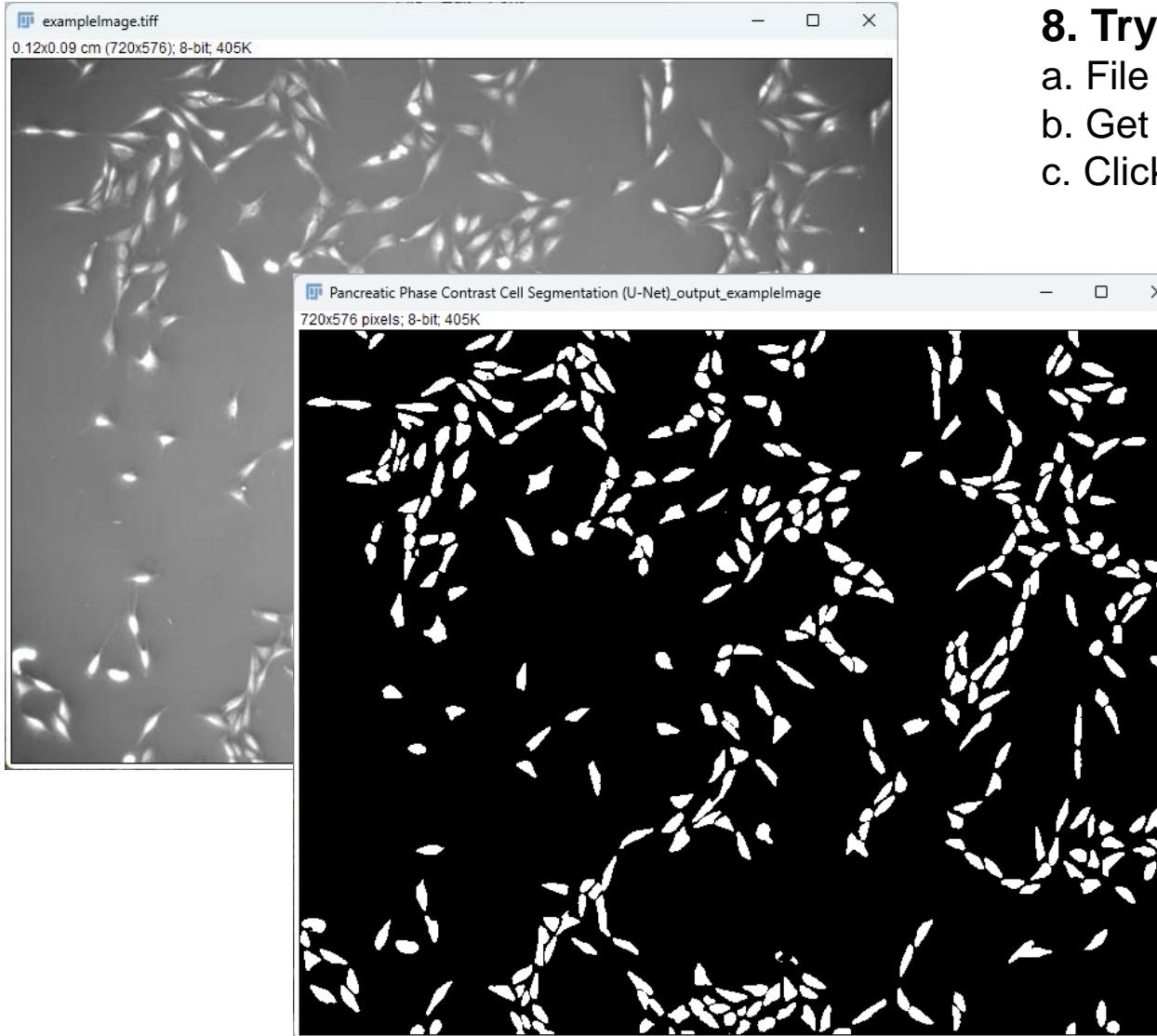
- Plugins > DeepImageJ > “...Run”. Wait for loading.
- Select your model. Click “Run on example image”.





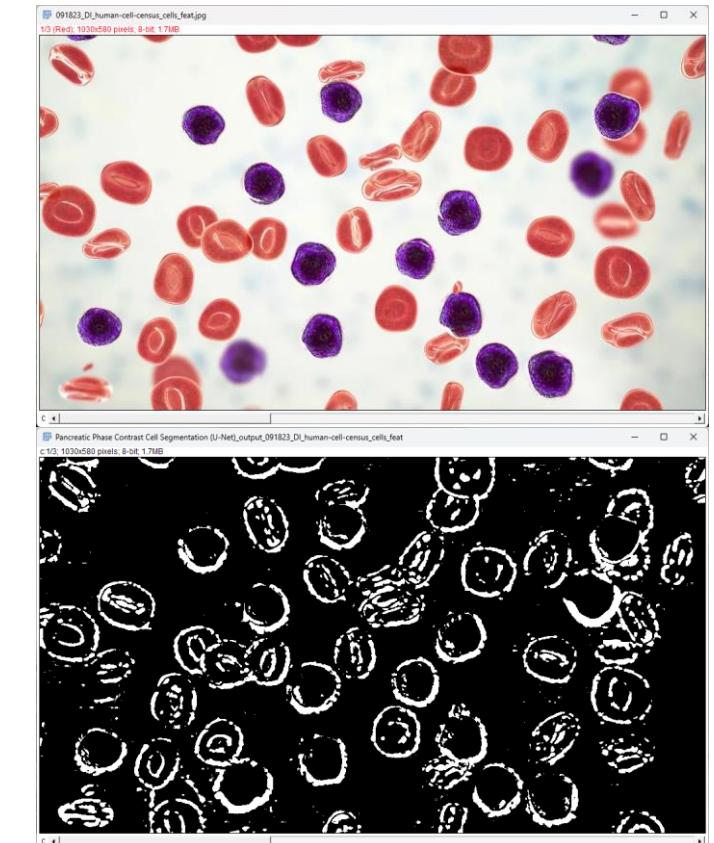
UNIVERSITY OF
ARKANSAS

Exercise 2 - FIJI



8. Try on your own image (or images/stack)

- File > Open. Select image(s)
- Get to the run model dialogue box as described in 7.
- Click on OK, instead of “Run on example image”

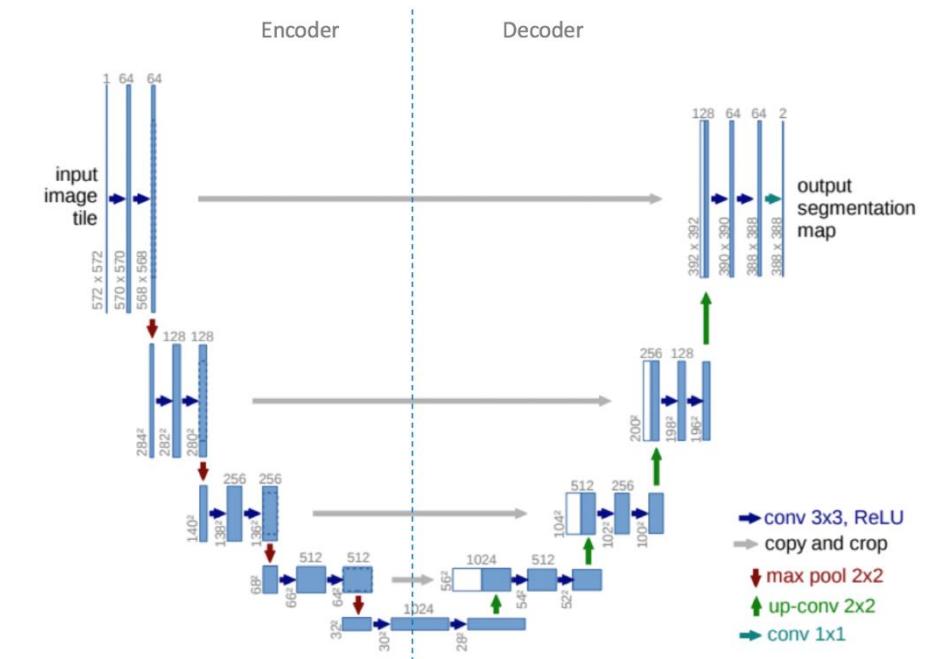
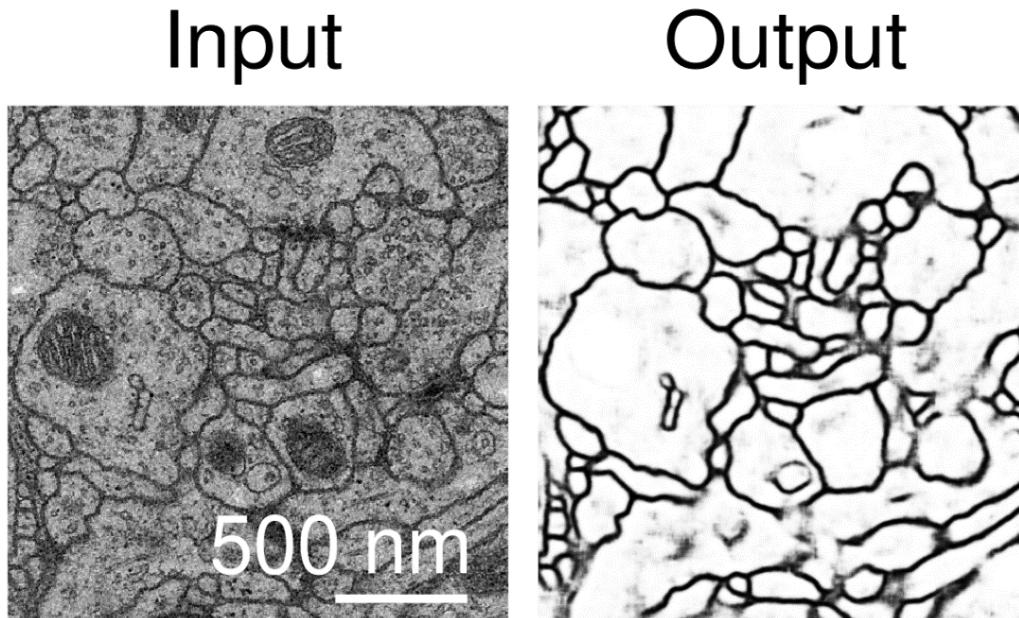




Popular Deep Learning based Bioimaging Models

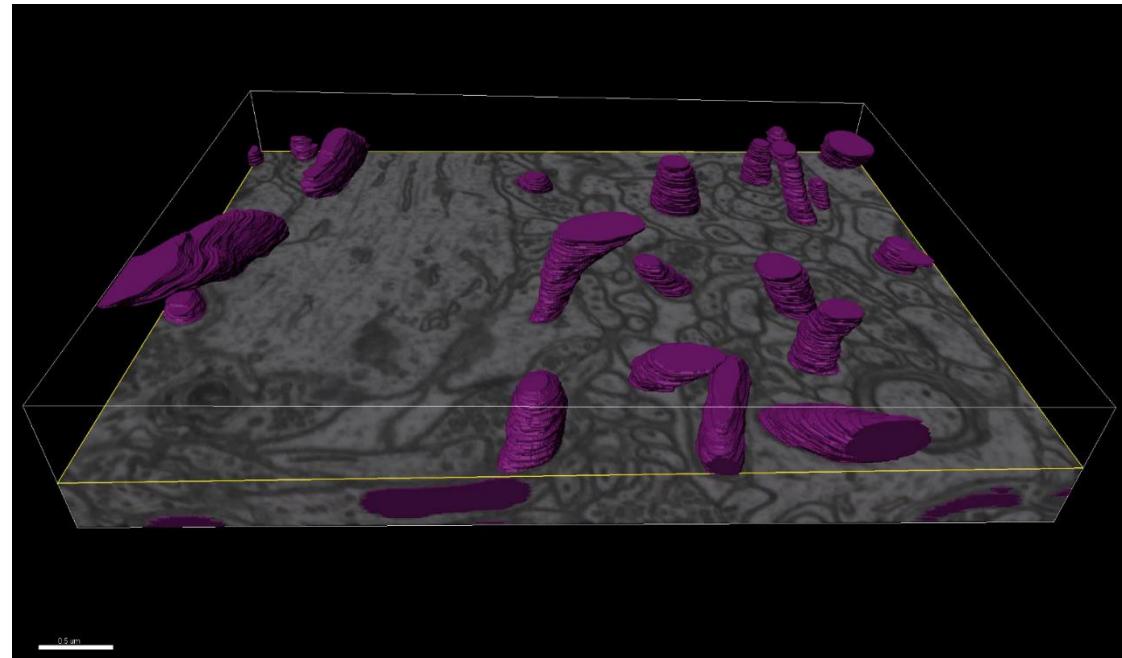
U-Net (2D)

- An encoder-decoder network architecture originally used for image segmentation.
- The first half of the U-Net architecture applies convolution blocks followed by a maxpool downsampling to encode the input image into feature representations at different levels.
- The other half consists of upsampling and concatenation followed by regular convolution operations and project lower resolution features onto the higher resolution to get a dense classification.



U-Net (3D)

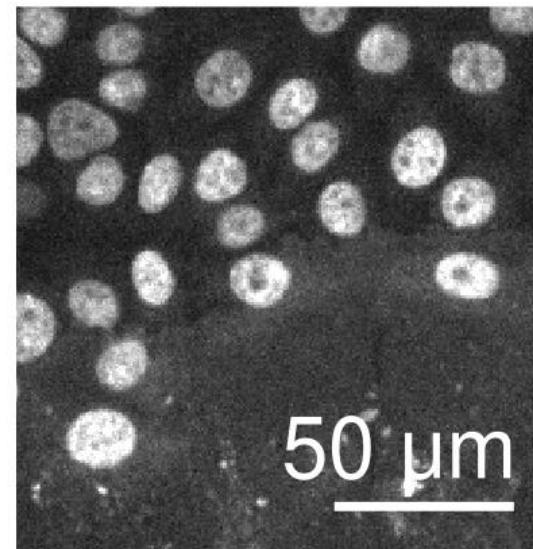
- A network for volumetric segmentation that learns from sparsely annotated volumetric images.
- Extends the U-Net architecture by replacing all 2D operations with 3D counterparts.



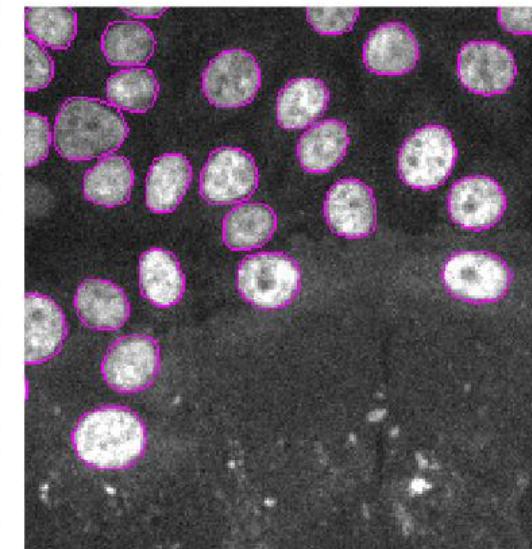
StarDist (2D)

- A DL method that can be used to segment cell nuclei from bioimages.
- Uses a shape representation based on star-convex polygons for nuclei in an image to predict the presence and the shape of these nuclei.
- Improves the approaches of using bounding boxes and does not need shape refinement.

Input



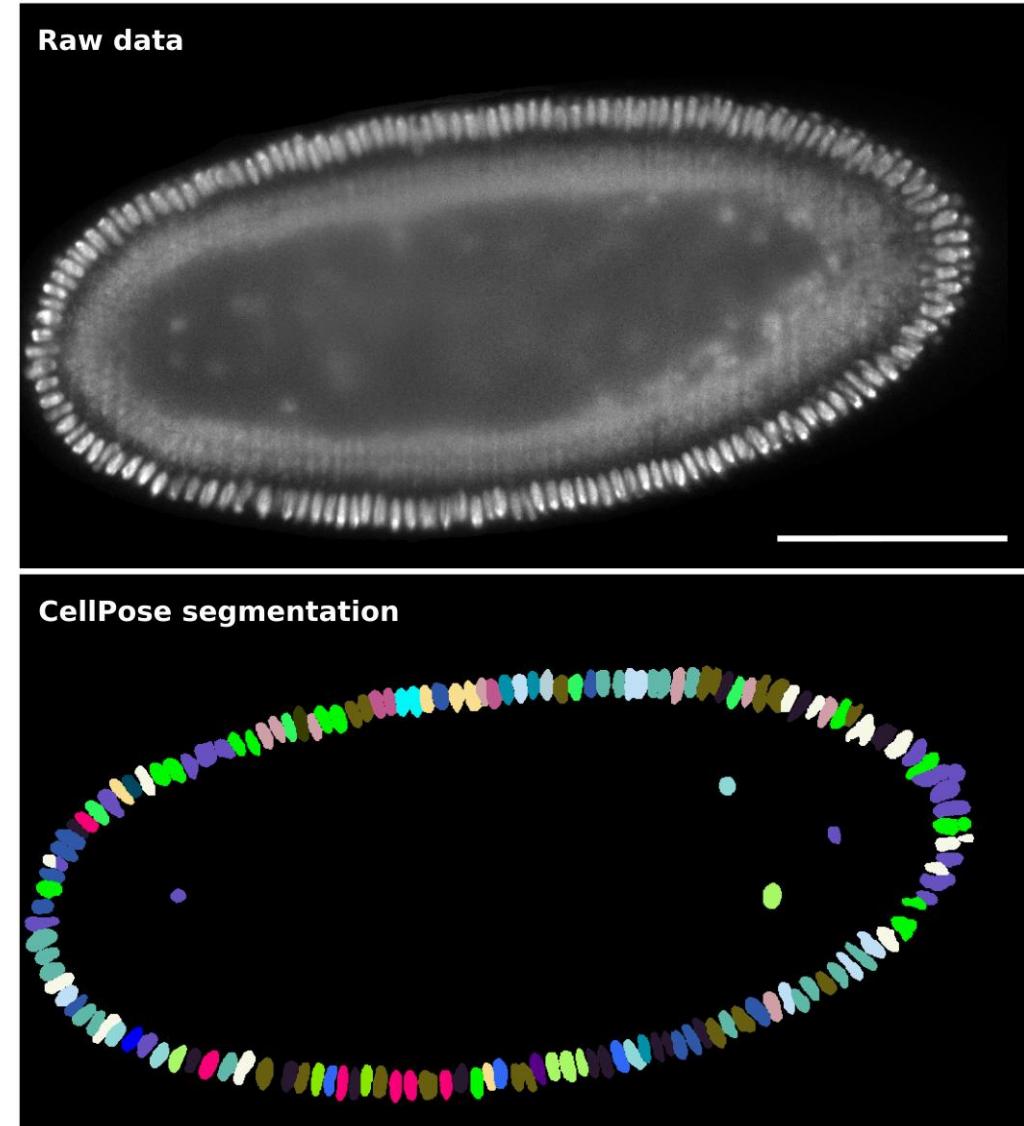
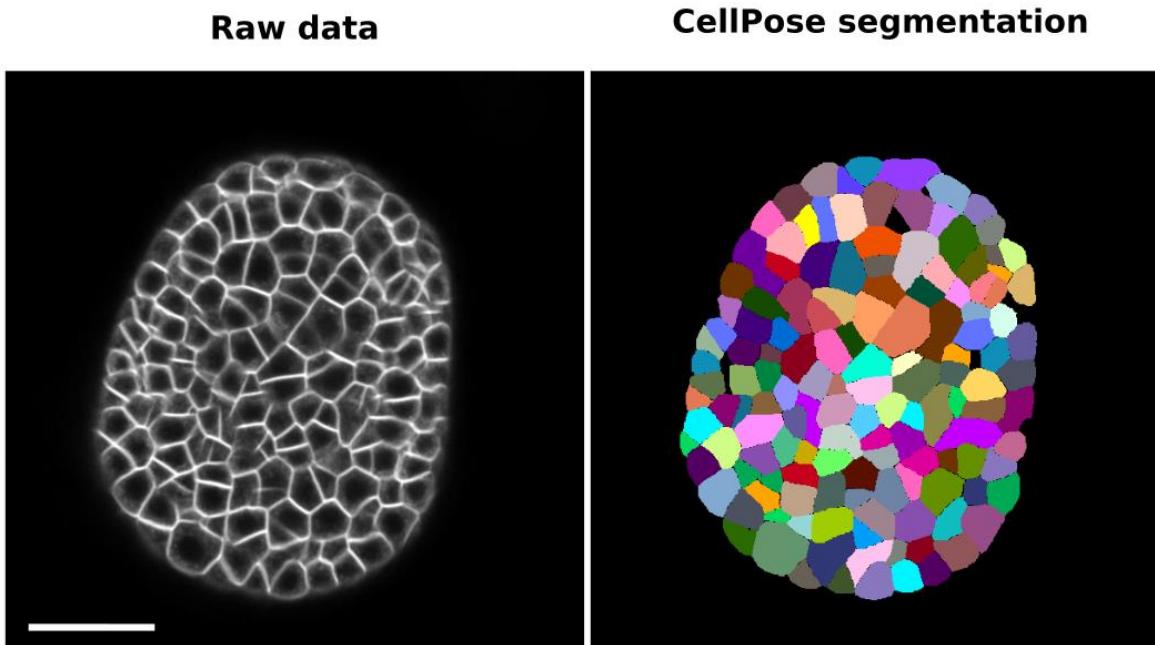
Output





Cellpose

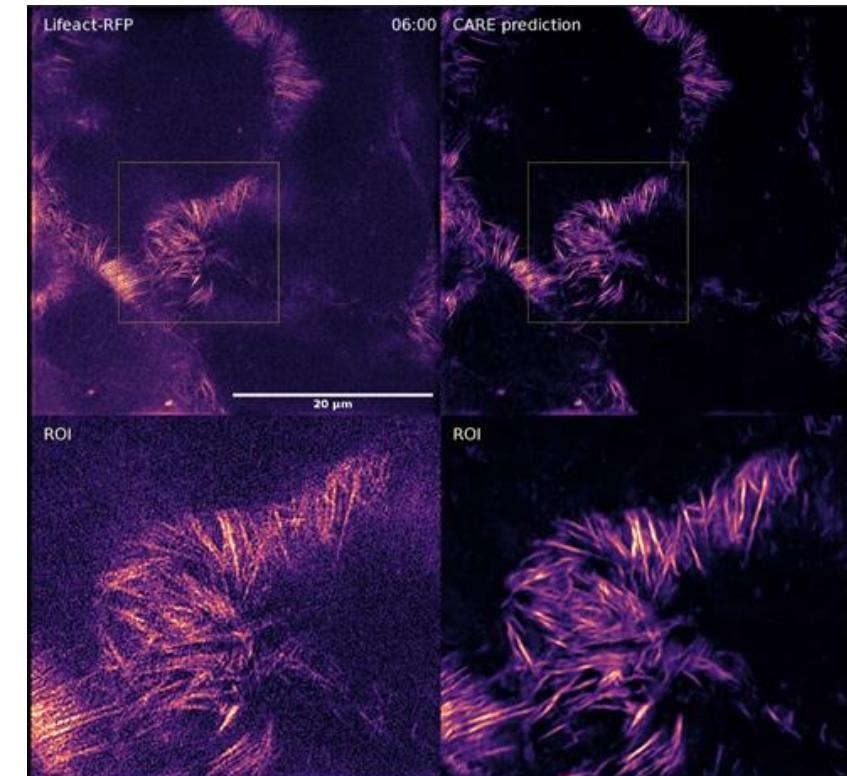
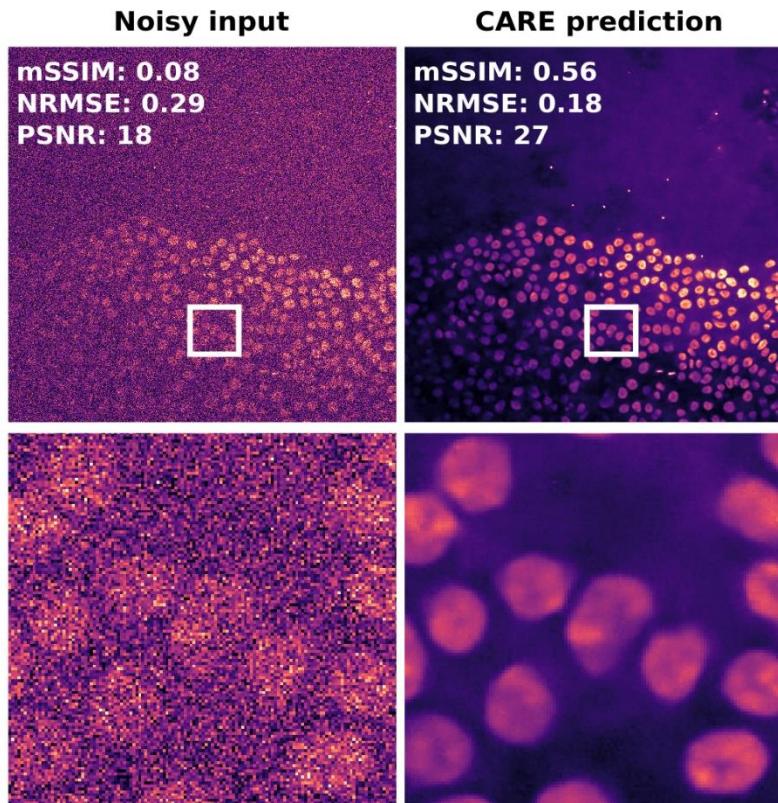
- Segment cell and/or nuclei from a wide range of image types and does not require model pretraining or parameter adjustments.
- Cellpose: a generalist algorithm for cellular segmentation, Nature Methods, 2020
<https://www.nature.com/articles/s41592-020-01018-x>





CARE

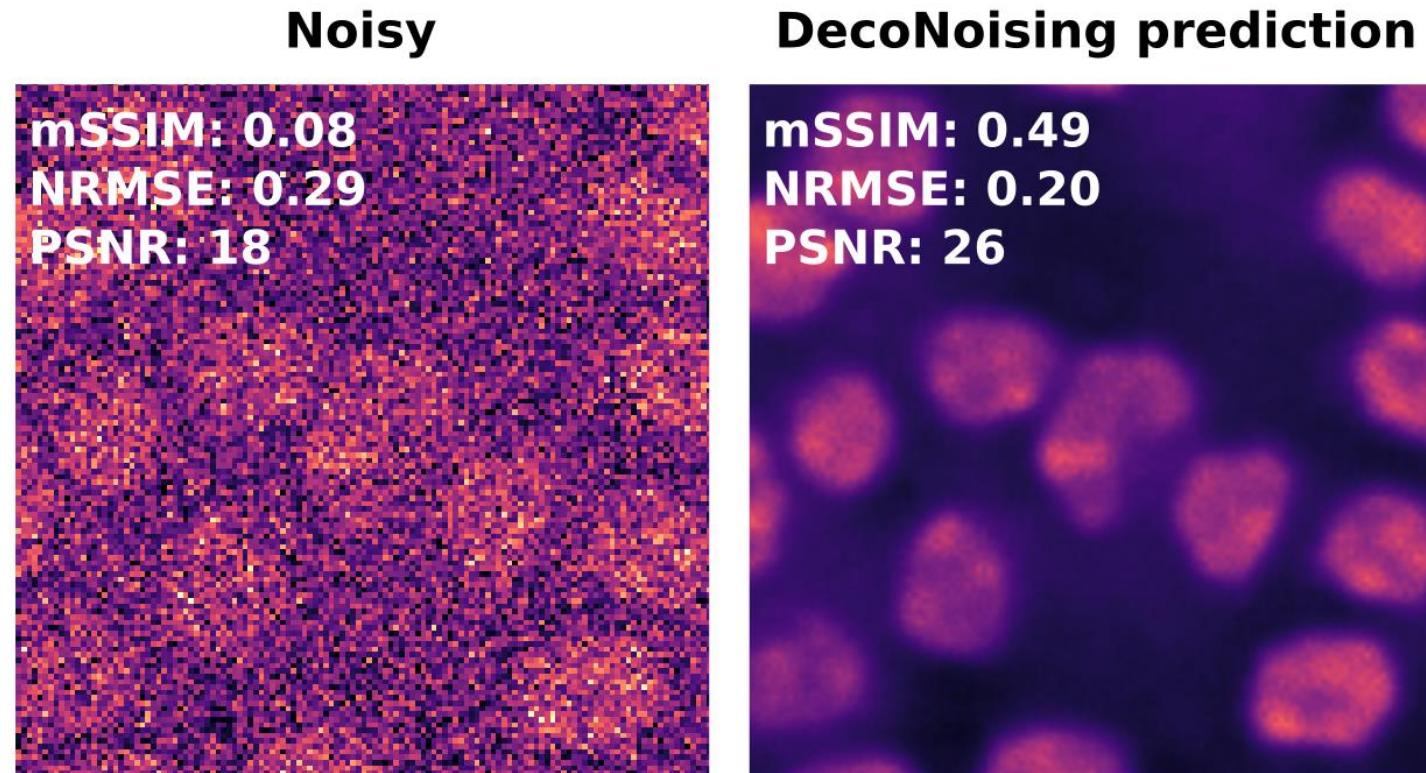
- Content-aware image restoration with matching pairs.
- Uses a U-Net network architecture.



Content-aware image restoration: pushing the limits of fluorescence microscopy, Nature Methods 2018
<https://www.nature.com/articles/s41592-018-0216-7>

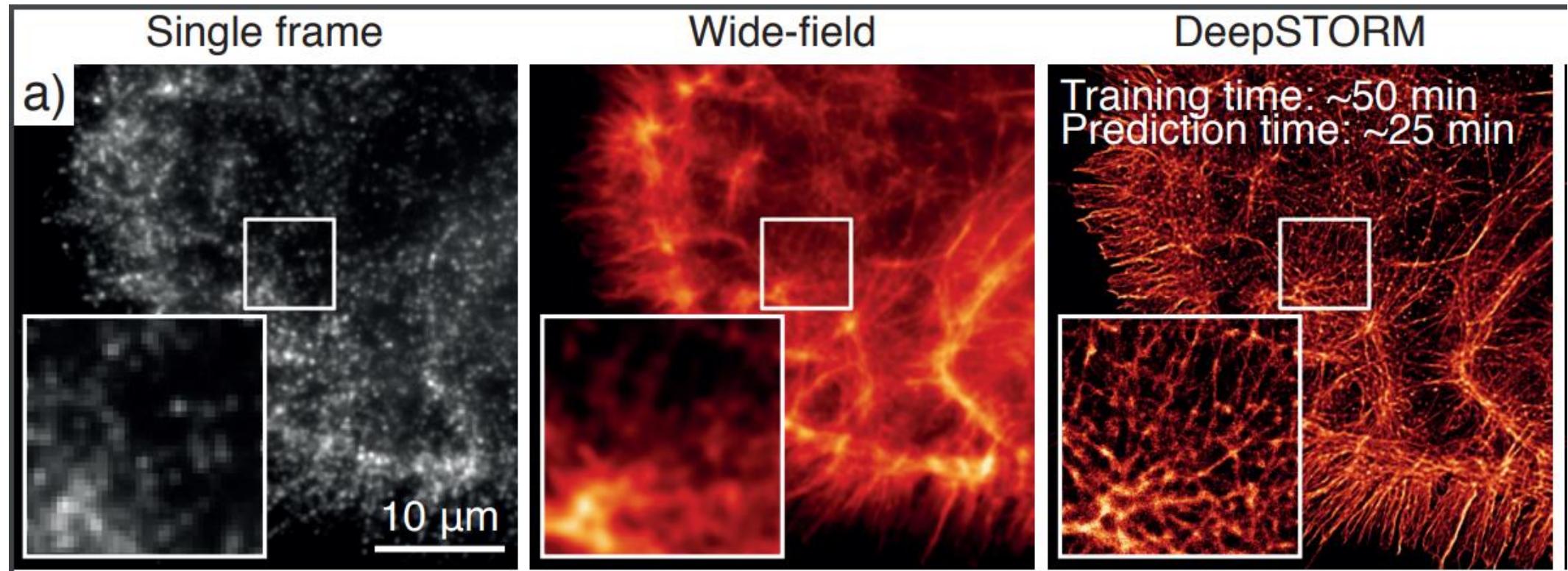
DecoNoising (2D)

- A self-supervised denoising method that does not need paired training data.
- Applies the point spread function to denoise and improve the predictions.



Deep-STORM (2D)

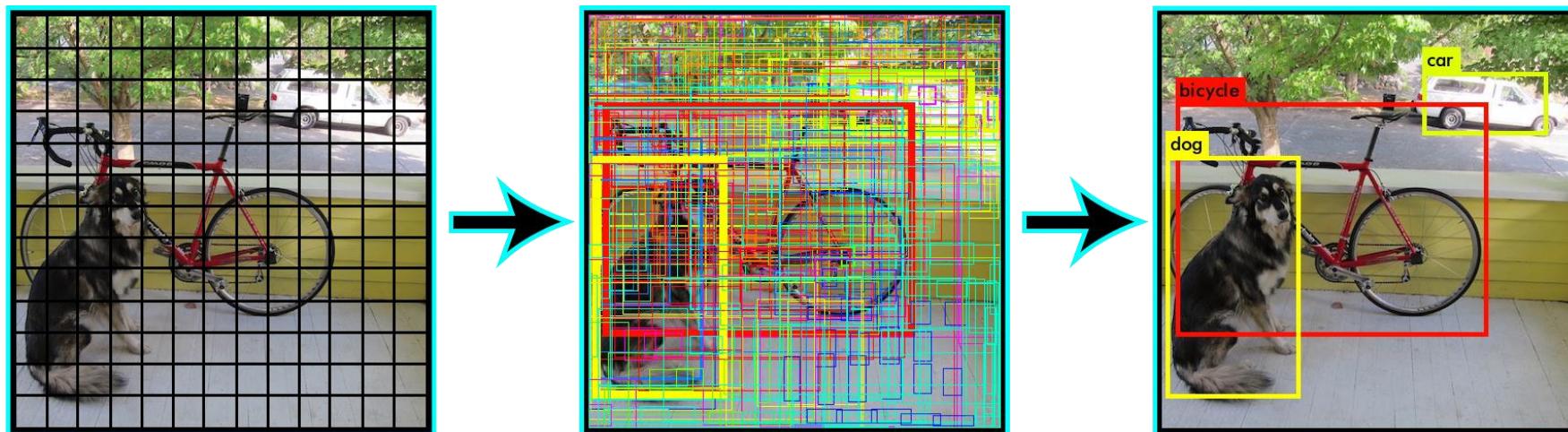
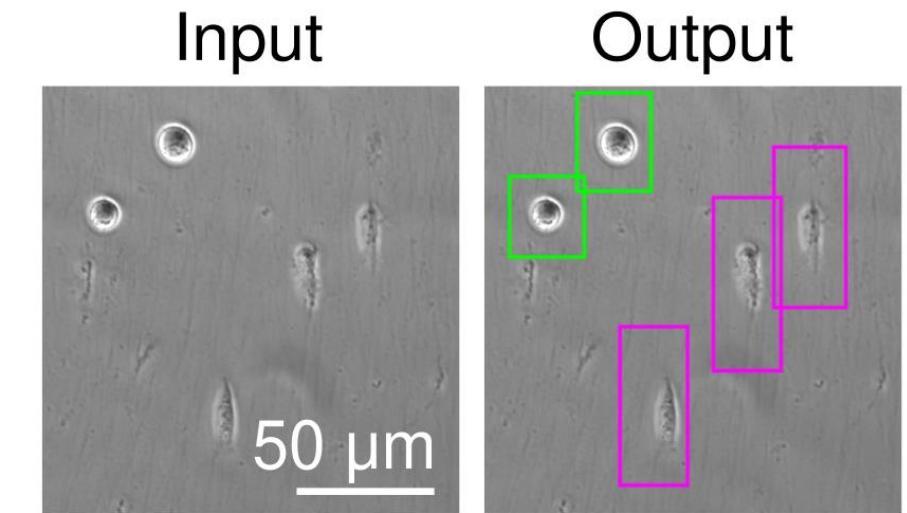
- A deep neural network capable of obtaining super-resolution images from fluorescent molecules used for localization microscopy.





YOLO

- A real time object detection system that can detect over 9000 object categories.
- The network divides the image into regions and predicts bounding boxes and prob. for each region.



Pix2Pix

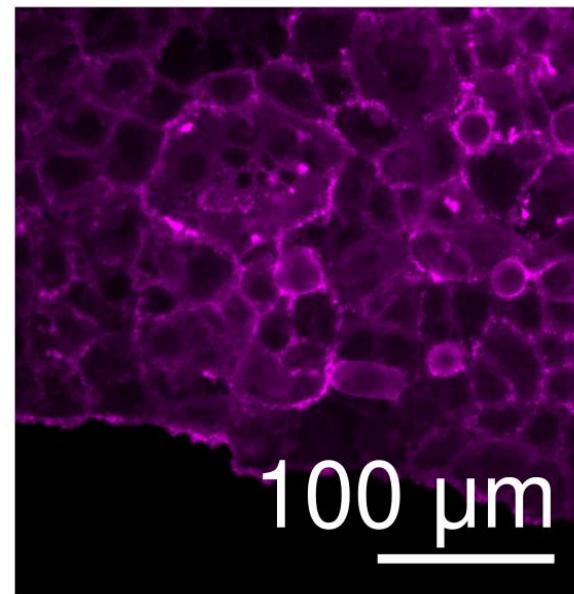
- A DL method allowing image-to-image translation from one image domain type to another image domain type.
 - Requires paired images in training.

Edges to Photo

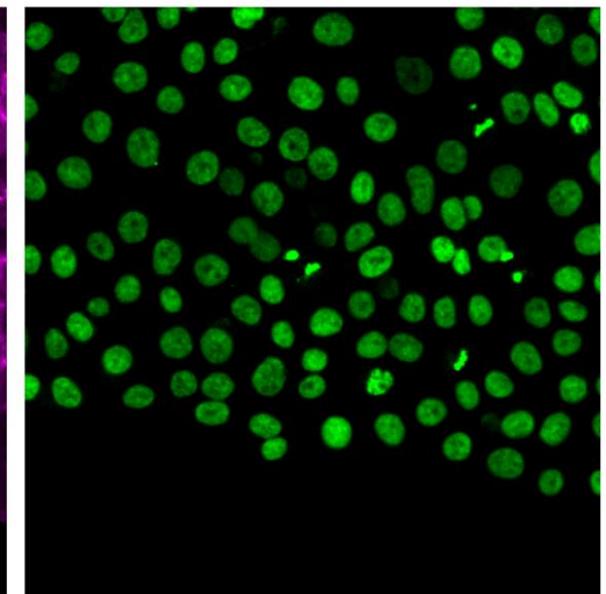
input

output

Input



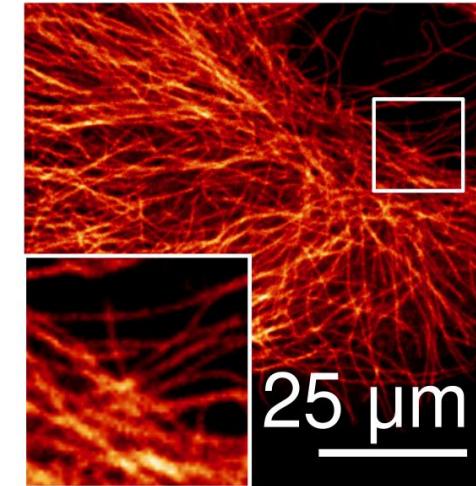
Output



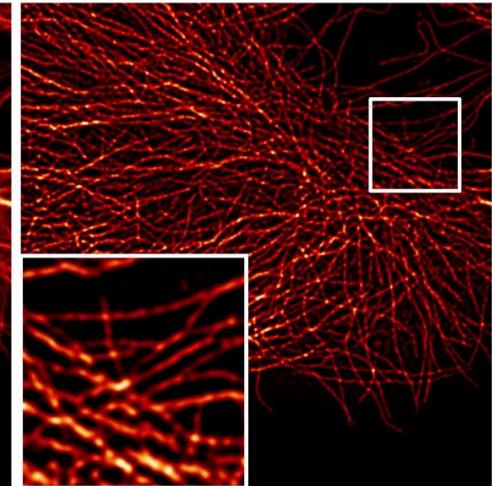
CycleGAN

- Introduces the cycle consistency loss.
- Does not require paired examples in the training.
- Achieves great success in collection style transfer, object transfiguration, season transfer, photo enhancement, etc.

Input



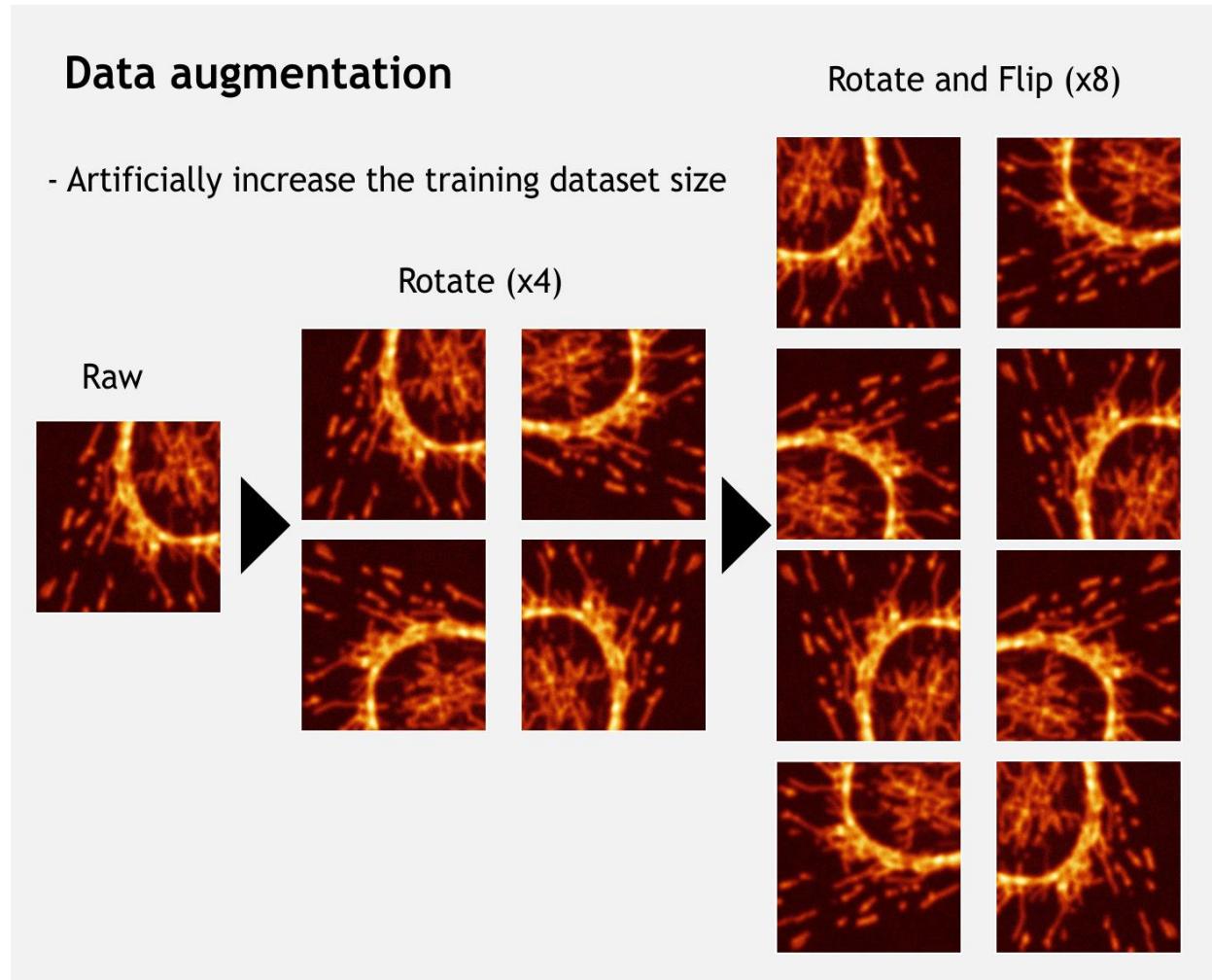
Output



winter Yosemite → summer Yosemite

Augmentor

- Data augmentation can improve training progress by amplifying differences in the dataset.
- Useful if the available dataset is small.
- Supports z-stack augmentation and randomized elastic distortions.





Exercise 3 – U-Net

Please follow the instructions in the Jupyter Notebook.



UNIVERSITY OF
ARKANSAS

Open source
Jupyter Notebooks
ZeroCostDL4Mic

ZeroCostDL4Mic

- A toolbox to provide notebook implementations of well-known models and deploy them in Google Colab.
- Two use cases
 - Online: No need to handle Python environments or GPU setup when using [Google Colab](#). However, the resources are limited with free subscription.
 - Offline: Using local machines/servers to run the notebook.
- Tasks: image segmentation, denoising, restoration, object detection
- Pros
 - Users can run it [online](#) with Colab or [offline](#) with local machines
 - Users have [more control](#) in defining the model, train/test procedure, dataset
 - ZeroCostDL4Mic can help [export to BioImage Model Zoo](#) format
- Cons
 - Users [need to have programming skills](#)
 - This is a code-based toolbox, not an interactive software



ZeroCostDL4Mic

Segmentation networks

Network	Paper(s)	Tasks	Status	Last test	Link to example training and test dataset	Direct link to the notebook in Colab
U-Net (2D)	here and here	Binary segmentation	Fully supported	⚠️ broken (no GPU) (GJ)	here	Open in Colab
U-Net (3D)	here	Binary segmentation	Fully supported	⚠️ broken (no GPU) (GJ)	EPFL dataset	Open in Colab
U-Net (2D) multilabel	here and here	Semantic segmentation	Under beta-testing	⚠️ broken (no GPU) (GJ)	here	Open in Colab
DenoiSeg	here	Joint denoising and binary segmentation	Fully supported	⚠️ broken (no GPU) (GJ)	Available soon	Open in Colab
StarDist (2D)	here and here	Instance segmentation	Fully supported	08/10/22 ✅ working (GJ)	here	Open in Colab
StarDist (3D)	here and here	Instance segmentation	Fully supported	07/10/22 ✅ working (GJ)	from Stardist github	Open in Colab
Cellpose (2D and 3D)	here	Instance segmentation (Cells or Nuclei)	Fully supported	08/10/22 ✅ working (GJ)	Coming soon!	Open in Colab
SplineDist (2D)	here	Instance segmentation	Fully supported	07/10/22 ✅ working (GJ)	here	Open in Colab
EmbedSeg (2D)	here	Instance segmentation	Under beta-testing	⚠️ broken by recent updates	here	Open in Colab
MaskRCNN (2D)	here	Instance segmentation	Under beta-testing		Coming soon!	Open in Colab
Interactive Segmentation - Kaibu (2D)	here	Interactive instance segmentation	Under beta-testing		Coming soon!	Open in Colab

Denoising and image restoration networks

Network	Paper(s)	Tasks	Status	Last test	Link to example training and test dataset	Direct link to the notebook in Colab
Noise2Void (2D)	here	Self-supervised denoising	Fully supported	06/10/22 ✅ working (EGM)	here or here	Open in Colab
Noise2Void (3D)	here	Self-supervised denoising	Fully supported	07/10/22 ✅ working (GJ)	here	Open in Colab
CARE (2D)	here	Supervised denoising	Fully supported	07/10/22 ✅ working (GJ)	here or here	Open in Colab
CARE (3D)	here	Supervised denoising	Fully supported	07/10/22 ✅ working (GJ)	here	Open in Colab
3D-RCAN	here	Supervised denoising	Under beta-testing	⚠️ broken (no GPU)	here	Open in Colab
DecoNoising (2D)	here	Self-supervised denoising	Under beta-testing	07/10/22 ✅ working (GJ)	here or here	Open in Colab

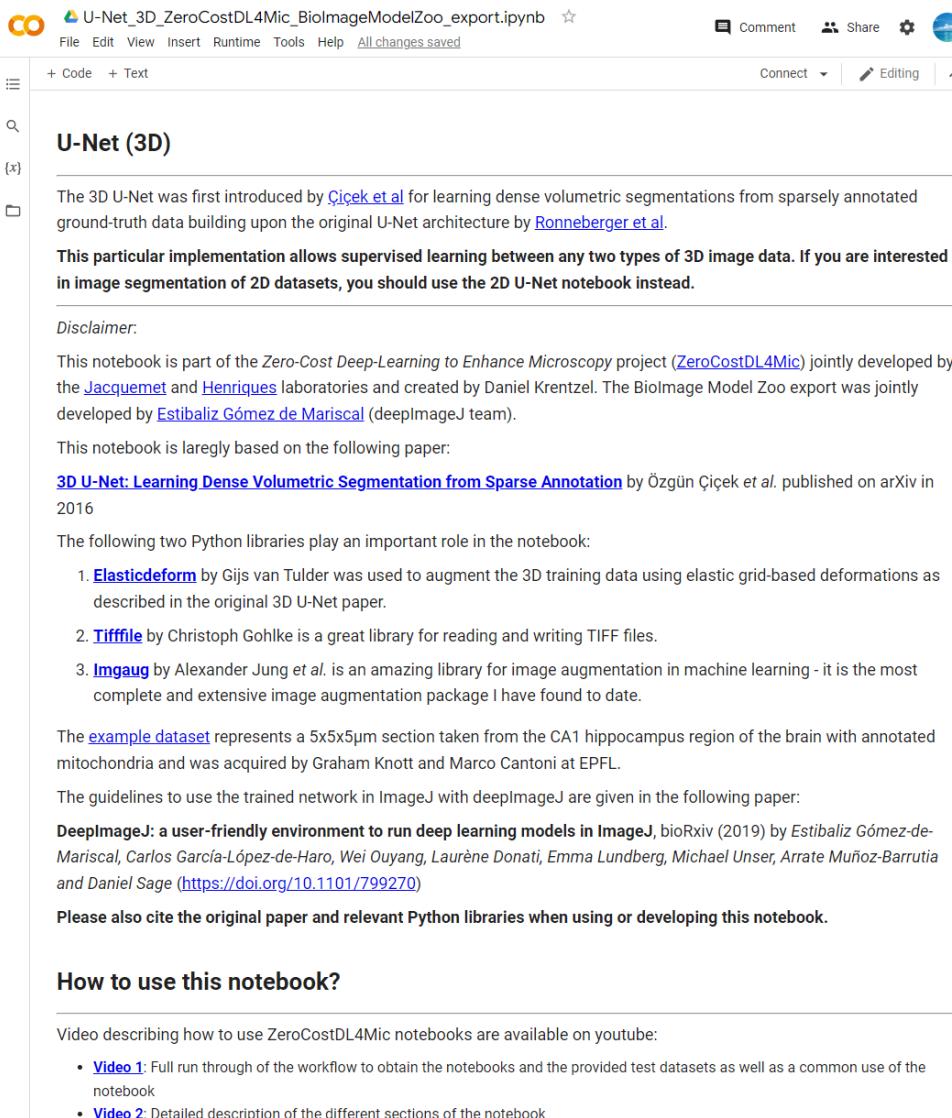
Start notebook with Google Colab

Details about model and data

Super-resolution microscopy networks

Network	Paper(s)	Tasks	Status	Last test	Link to example training and test dataset	Direct link to the notebook in Colab
Deep-STORM	here	Single Molecule Localization Microscopy (SMLM) image reconstruction from high-density emitter data	Fully supported	08/10/22 ✅ working (GJ)	Training data simulated in the notebook or available from here	Open in Colab
DFCAN	here	image upsampling	Under beta-testing	08/10/22 ✅ working (GJ)	here	Open in Colab
WGAN	here	image upsampling	Under beta-testing	22/09/22 ✅ working (IvanHidalgo & EGM)	here	Open in Colab

ZeroCostDL4Mic



U-Net (3D)

The 3D U-Net was first introduced by [Çiçek et al.](#) for learning dense volumetric segmentations from sparsely annotated ground-truth data building upon the original U-Net architecture by [Ronneberger et al.](#).

This particular implementation allows supervised learning between any two types of 3D image data. If you are interested in image segmentation of 2D datasets, you should use the 2D U-Net notebook instead.

Disclaimer:

This notebook is part of the Zero-Cost Deep-Learning to Enhance Microscopy project ([ZeroCostDL4Mic](#)) jointly developed by the [Jacquemet](#) and [Henriques](#) laboratories and created by Daniel Krentzel. The BiolImage Model Zoo export was jointly developed by [Estibaliz Gómez de Mariscal](#) (deeplImageJ team).

This notebook is largely based on the following paper:

[3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation](#) by Özgün Çiçek et al. published on arXiv in 2016

The following two Python libraries play an important role in the notebook:

1. [Elasticdeform](#) by Gijs van Tulder was used to augment the 3D training data using elastic grid-based deformations as described in the original 3D U-Net paper.
2. [Tifffile](#) by Christoph Gohlke is a great library for reading and writing TIFF files.
3. [Imgaug](#) by Alexander Jung et al. is an amazing library for image augmentation in machine learning - it is the most complete and extensive image augmentation package I have found to date.

The [example dataset](#) represents a 5x5x5µm section taken from the CA1 hippocampus region of the brain with annotated mitochondria and was acquired by Graham Knott and Marco Cantoni at EPFL.

The guidelines to use the trained network in ImageJ with deeplImageJ are given in the following paper:

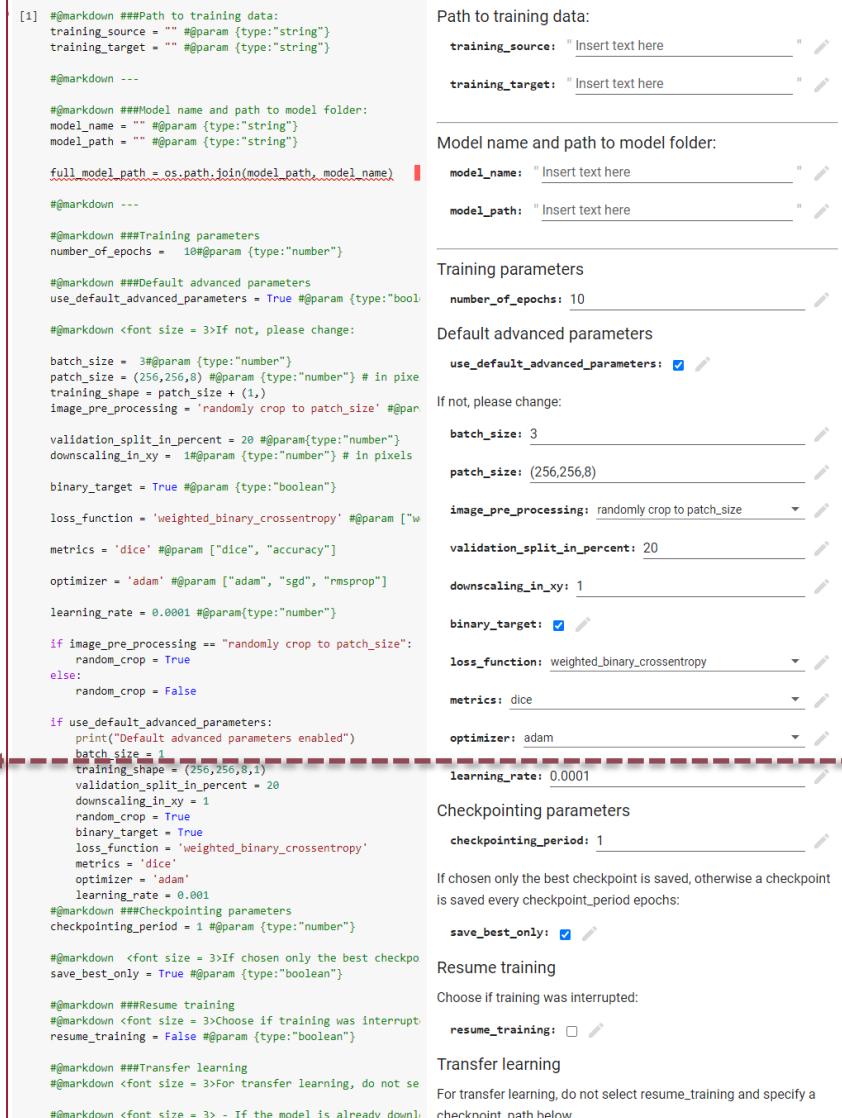
[DeeplImageJ: a user-friendly environment to run deep learning models in ImageJ](#), bioRxiv (2019) by Estibaliz Gómez-de-Mariscal, Carlos García-López-de-Haro, Wei Ouyang, Laurène Donati, Emma Lundberg, Michael Unser, Arrate Muñoz-Barrutia and Daniel Sage (<https://doi.org/10.1101/799270>)

Please also cite the original paper and relevant Python libraries when using or developing this notebook.

How to use this notebook?

Video describing how to use ZeroCostDL4Mic notebooks are available on youtube:

- [Video 1](#): Full run through of the workflow to obtain the notebooks and the provided test datasets as well as a common use of the notebook
- [Video 2](#): Detailed description of the different sections of the notebook



[1] `##Path to training data:
training_source = "" #@param {type:"string"}
training_target = "" #@param {type:"string"}

##Model name and path to model folder:
model_name = "" #@param {type:"string"}
model_path = "" #@param {type:"string"}
full_model_path = os.path.join(model_path, model_name)

##Training parameters
number_of_epochs = 10#@param {type:"number"}

##Default advanced parameters
use_default_advanced_parameters = True #@param {type:"bool"}

##If not, please change:
batch_size = 3#@param {type:"number"}
patch_size = (256,256,8) #@param {type:"number"} # in pixels
training_shape = patch_size + (1,)
image_pre_processing = 'randomly crop to patch_size' #@param {type:"string"}
validation_split_in_percent = 20 #@param{type:"number"}
downscaling_in_xy = 1#@param {type:"number"} # in pixels
binary_target = True #@param {type:"boolean"}
loss_function = 'weighted_binary_crossentropy' #@param ["
dice", "accuracy"]
metrics = 'dice' #@param ["dice", "accuracy"]
optimizer = 'adam' #@param ["adam", "sgd", "rmsprop"]
learning_rate = 0.0001 #@param{type:"number"}

if image_pre_processing == "randomly crop to patch_size":
 random_crop = True
else:
 random_crop = False

if use_default_advanced_parameters:
 print("Default advanced parameters enabled")
 batch_size = 1
 training_shape = (256,256,8,1)
 validation_split_in_percent = 20
 downscaling_in_xy = 1
 random_crop = True
 binary_target = True
 loss_function = 'weighted_binary_crossentropy'
 metrics = 'dice'
 optimizer = 'adam'
 learning_rate = 0.001
 #Markdown: Checkpointing parameters
 checkpointing_period = 1 #@param {type:"number"}

 ##If chosen only the best checkpoint is saved, otherwise a checkpoint is saved every checkpoint_period epochs:
 save_best_only:

Resume training
Choose if training was interrupted:
resume_training:

Transfer learning
For transfer learning, do not select resume_training and specify a checkpoint_path below.`

Path to training data:
 training_source: "Insert text here"

Model name and path to model folder:
 model_name: "Insert text here"
 model_path: "Insert text here"

Training parameters
 number_of_epochs: 10

Default advanced parameters
 use_default_advanced_parameters:

If not, please change:
 batch_size: 3
 patch_size: (256,256,8)
 image_pre_processing: randomly crop to patch_size
 validation_split_in_percent: 20
 downscaling_in_xy: 1
 binary_target:
 loss_function: weighted_binary_crossentropy
 metrics: dice
 optimizer: adam
 learning_rate: 0.0001

Checkpointing parameters
 checkpointing_period: 1

If chosen only the best checkpoint is saved, otherwise a checkpoint is saved every checkpoint_period epochs:
 save_best_only:

Resume training
Choose if training was interrupted:
 resume_training:

Transfer learning
For transfer learning, do not select resume_training and specify a checkpoint_path below.

Either input parameters to input form or modify the code

ZeroCostDL4Mic provides very detailed instruction



Segmentation

Network	Paper(s)	Tasks	Status	Last test	Link to example training and test dataset	Direct link to the notebook in Colab
U-Net (2D)	here and here	Binary segmentation	Fully supported	broken (no GPU) (GJ)	here	Open in Colab
U-Net (3D)	here	Binary segmentation	Fully supported	broken (no GPU) (GJ)	EPFL dataset	Open in Colab
U-Net (2D) multilabel	here and here	Semantic segmentation	Under beta-testing	broken (no GPU) (GJ)	here	Open in Colab
DenoiSeg	here	Joint denoising and binary segmentation	Fully supported	broken (no GPU) (GJ)	Available soon	Open in Colab
StarDist (2D)	here and here	Instance segmentation	Fully supported	08/10/22 <input checked="" type="checkbox"/> working (GJ)	here	Open in Colab
StarDist (3D)	here and here	Instance segmentation	Fully supported	07/10/22 <input checked="" type="checkbox"/> working (GJ)	from Stardist github	Open in Colab



Segmentation (Cont'd)

Cellpose (2D and 3D)	here	Instance segmentation (Cells or Nuclei)	Fully supported	08/10/22 working (GJ)	Coming soon!	Open in Colab
SplineDist (2D)	here	Instance segmentation	Fully supported	07/10/22 working (GJ)	here	Open in Colab
EmbedSeg (2D)	here	Instance segmentation	Under beta-testing	broken by recent updates	here	Open in Colab
MaskRCNN (2D)	here	Instance segmentation	Under beta-testing		Coming soon!	Open in Colab
Interactive Segmentation - Kaibu (2D)	here	Interactive instance segmentation	Under beta-testing		Coming soon!	Open in Colab

Denoising and Image Restoration

Network	Paper(s)	Tasks	Status	Last test	Link to example training and test dataset	Direct link to the notebook in Colab
Noise2Void (2D)	here	Self-supervised denoising	Fully supported	06/10/22 working (EGM)	here or here	Open in Colab
Noise2Void (3D)	here	Self-supervised denoising	Fully supported	07/10/22 working (GJ)	here	Open in Colab
CARE (2D)	here	Supervised denoising	Fully supported	07/10/22 working (GJ)	here or here	Open in Colab
CARE (3D)	here	Supervised denoising	Fully supported	07/10/22 working (GJ)	here	Open in Colab
3D-RCAN	here	Supervised denoising	Under beta-testing	broken (no GPU)	here	Open in Colab
DecoNoising (2D)	here	Self-supervised denoising	Under beta-testing	07/10/22 working (GJ)	here or here	Open in Colab



Super-resolution Microscopy

Network	Paper(s)	Tasks	Status	Last test	Link to example training and test dataset	Direct link to the notebook in Colab
Deep-STORM	here	Single Molecule Localization Microscopy (SMLM) image reconstruction from high-density emitter data	Fully supported	08/10/22 working (GJ)	Training data simulated in the notebook or available from here	Open in Colab
DFCAN	here	image upsampling	Under beta-testing	08/10/22 working (GJ)	here	Open in Colab
WGAN	here	image upsampling	Under beta-testing	22/09/22 working (IvanHidalgo & EGM)	here	Open in Colab



Object Detection

Network	Paper(s)	Tasks	Status	Last test	Link to example training and test dataset	Direct link to the notebook in Colab
YOLOv2	here	Object detection (bounding boxes)	Fully supported		here	Open in Colab
Detectron2	here	Object detection (bounding boxes)	Under beta-testing		here	Open in Colab
RetinaNet	here	Object detection (bounding boxes)	Under beta-testing		here	Open in Colab

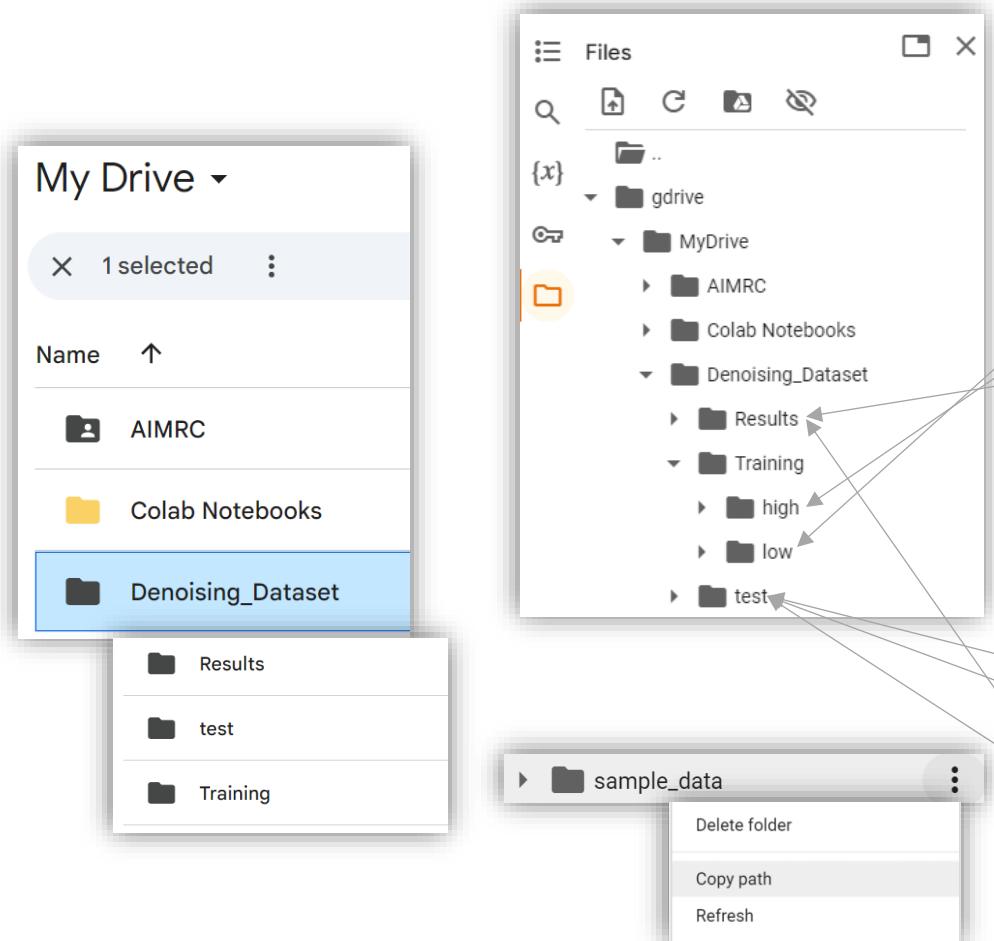


Image-to-image Translation

Network	Paper(s)	Tasks	Status	Last test	Link to example training and test dataset	Direct link to the notebook in Colab
Label-free prediction (fnet) 2D	here	Artificial labelling	Under beta-testing	11/08/22 working (EGM)	Coming soon	Open in Colab
Label-free prediction (fnet) 3D	here	Artificial labelling	Fully supported		here	Open in Colab
CycleGAN	here	Unpaired Image-to-Image Translation	Fully supported		here	Open in Colab
pix2pix	here	Paired Image-to-Image Translation	Fully supported	04/08/22 working (Sujan Ghimire)	here	Open in Colab

Exercise 4 – ZeroCostDL4Mic

Please follow the instructions in the Jupyter Notebook.



1. Upload denoising dataset to your Google Drive
2. Open ZeroCostDLC4Mic Notebook in a new tab.
Save it as a copy to your Drive.
3. Connect to a T4 GPU session.
4. Edit following in the notebook file

Section 3.1

Training_source: /content/gdrive/MyDrive/Denoising_Dataset/Training/low

Training_target: /content/gdrive/MyDrive/Denoising_Dataset/Training/high

model_name: denoising_model

model_path: /content/gdrive/MyDrive/Denoising_Dataset/Results

number of epochs: 5 you can increase/decrease this number depending on whether or not you are connected to a GPU kernel and/or if you have time

Section 4.2 (Important)

Change lr to learning_rate in this line

```
writer.writerow([history.history['loss'][i], history.history['val_loss'][i],  
history.history['learning_rate'][i]])
```

Section 5.2

Source QC folder: /content/gdrive/MyDrive/Denoising_Dataset/test/Low

Target QC folder: /content/gdrive/MyDrive/Denoising_Dataset/test/High

Section 6.1

Data folder: /content/gdrive/MyDrive/Denoising_Dataset/test/Low

Result folder: /content/gdrive/MyDrive/Denoising_Dataset/Results



Even more resources
PyDeepImageJ
CSBDeep

PydeeplImageJ

- Q: How can we use DeepImageJ with our own model?
- PydeeplImageJ is a Python package and can help convert Python models to BioImage Model Zoo format.
- Steps:
 - Training the proposed model in Python.
 - Exporting the trained model to DeepImageJ's format using PydeeplImageJ.
 - Loading exported model to DeepImageJ.
 - Continuing exploration with DeepImageJ.

CSBDeep

- A DL-based toolbox for microscopy image restoration and segmentation.
- Two use cases
 - CSBDeep for Python: users can create and train model with their own data from scratch.
 - CSBDeep for Fiji: pre-trained model with CSBDeep can be loaded to Fiji for training or testing.
- Pros
 - Provides users an ability to train model from scratch.
 - Allows users to train model with their own data directly on Fiji.
 - Allows users to export their models to Fiji's format.
- Cons
 - Supports Tensorflow 1 and 2, but **not Pytorch**



Guide

- I want to learn current DL-based methods, datasets or tools. How could I start?
 - [Start with Biolimage Model Zoo](#)
- I want to try some traditional image processing first. What is the best way?
 - [Fiji/ImageJ](#)
- Without programming skills or technical details. What is the best way to try DL methods?
 - [DeepImageJ and CSBDeep in Fiji](#)
- Without programming skills. How to train some existing models with my own data?
 - [CSBDeep in Fiji and ZeroCostDL4Mic](#)
- I am okay with Python, PyTorch or TensorFlow. What is the best way to try DL methods?
 - [CSBDeep in Python and ZeroCostDL4Mic](#)