**UPSIDE SOFTWARE USER GUIDE**

**System Requirements**

* We recommend Linux Ubuntu 16.04.
* > 10 GB of RAM NVIDIA graphics card. We used NVIDIA Titan X Pascal.

**Installation**

* Python version 3.8.6 with conda version 4.8.2
* CUDA version 9.1 or higher.
* Pytorch version 1.4.0 or higher.
* Docker version 19.03.5 or higher.
* Tensorflow version 1.12.0 or higher for Tensorboard integration in Pytorch. (We used the customized version by <https://github.com/yunjey/pytorch-tutorial/tree/master/tutorials/04-utils/tensorboard>).
* MATLAB v9.0 or higher with ictrack software (download and installation instructions: <https://github.com/KuehLabUW/ictrack/>)

**Description**

This is the user guide on how to run the UPSIDE software that accompanies the manuscript *‘Unsupervised discovery of dynamic cell phenotypic states from transmitted light movies’ (Nguyen et al., 2021).* *Image data is available for download separately at* [*https://idr.openmicroscopy.org/about/experiments.html*](https://idr.openmicroscopy.org/about/experiments.html) *(work in progress)*

Once the data file is downloaded from IDR, place it in the UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/data/ directory.

Below is a tutorial of how to apply the UPSIDE software to generate latent encoding and decoding on the Hematopoietic Cell Types Dataset

**Part I. Label-free Cell Recognition Module (Adapted from Ounkomol et al., 2018)**

* Refer to <https://github.com/AllenCellModeling/pytorch_fnet/tree/release_1> for the original label-free imaging software.
* We modified the software’s scripts to accept 2D images and tiff files format, see ‘UPSIDE\_Patchnote.txt’ for a detailed list of modification.
* To perform model training of fluorescent images:
  + Place folders with acquired training and labeled images to:

UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/data/

* + Add a csv file detailing a list of Brightfield image names in the left column labeled ‘signal’ and a list of corresponding fluorescent image names in the right column labeled ‘target’ to:  
    UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/data/csvs/CellTypeCellTraceTrain112319v2.csv
  + Edit the training bash script with the name of the csv file and place into:

UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/scripts/train\_CellTypeCellTraceTrain112319v2.sh

* + Open docker: UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/docker/$ sudo ./start\_dockerSN.sh
  + Run the training bash script: ~/projects/pytorch\_fnet# ./scripts/ train\_CellTypeCellTraceTrain112319v2.sh
  + The trained model will be saved in: UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/saved\_models/
* To perform fluorescent image prediction of trained model:
  + Place folders with acquired training and labeled images to:

UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/data/

* + Add a csv file detailing a list of Brightfield image names in the left column labeled ‘signal’ and a list of corresponding ground truth image names in the right column labeled ‘target’ to the data/csvs/ folder containing the name of the trained model such as:  
    UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/data/csvs/ CellTypeCellTraceTrain112319/
  + Edit the training bash script with the name of the csv file. Specify the model’s name and the testing csv file’s name and place into:

UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/scripts/predict\_CellTypeCellTraceTrain112319v2.sh

* + Open docker: UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/docker/$ sudo ./start\_dockerSN.sh
  + Run the training bash script: ~/projects/pytorch\_fnet# ./scripts/ predict\_CellTypeCellTraceTrain112319v2.sh
  + Predicted images will be stored in folder with the model name in:  
    UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/results/3d/

**Part II. Cell Segmentation Module (Adapted from Ng et al., 2018)**

* This module uses the ictrack software in MATLAB introduced in Ng et al., 2018 that provides segmentation for MetaMorph generated tiff images ( see [https://www.moleculardevices.com/products/cellular-imaging-systems/acquisition-and-analysis-software/metamorph-microscopy#gref](https://www.moleculardevices.com/products/cellular-imaging-systems/acquisition-and-analysis-software/metamorph-microscopy" \l "gref) for MetaMorph software)
* Convert predicted images to MetaMorph (MM) image format for segmentation:
  + In directory UPSIDEv1/code/ Run the NNFile\_convert.m functio in MATLAB. Enter values the following arguments:
    1. %in\_directory %the directory with the predicted images
    2. %no\_tps %the number of time points in your movie
    3. %no\_pos %the number of positions you have
    4. %image\_name %the name of your predicted images. Can be found in R Results/3d/{model\_name} directory
    5. %MMname % name of the converted image, adding a channel for the predicted images
  + For example, in the MATLAB console:

>> NNFile\_convert('/UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/results/3d/CellTypeCellTraceTrain112319v2/testCellTypes020420',11, 240, 'prediction\_CellTypeCellTraceTrain112319v2.tiff','EXP2\_w2Camera CellTrace')

* + Transfer the converted image files into the data folder

>> find UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/results/3d/CellTypeCellTraceTrain112319v2/testCellTypes020420Converted/ -type f -print0 | xargs -0 mv -t /UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/data/CellTypes020420

* Segment the cells with ictrack using the predicted fluorescent images
  + Start the MATLAB software and set path with all subfolders UPSIDEv1/code/ictrack/
  + Open script main2.m in UPSIDEv1/code/segmentation/AllCellTypes020420/ and edit the indir, basenames, and outdir variables with the appropriate directories with the imaging data:

indir = {'UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/data/CellTypes020420'};

basenames = {'EXP2'};

outdir = {‘UPSIDEv1/data/CellTypes020420/CellTypes020420 - segmented '};

* + Open script makeparams.m in UPSIDEv1/code/segmentation/AllCellTypes020420/ and edit the following variables according to your dataset (if your dataset has more than 2 channels, including the synthetic fluorescent one, add the channel setting as appropriately):

S = 240; % total number of stage positions

X = 1080; % the number of row pixels

Y = 1080; % the number of column pixels

Z = 1; % the number of z-sections

M = 1; % the number of row tiles

N = 1; % the number of column tiles

segchannel = 2; % fluorescence channel used for segmentation

datachannel = 2; % fluorescence channel used to set 'skipping' channel for image viewing

* + Choose the segmentation script’s name:

segfun = 'cellseg120518MLCellTrace'; % user-written function for cell segmentation

cellprops = 'cellprops\_aml';

* + Run the main2.m script to start segmentation:

>> main2.m

**Part III. Cell Crops Generation Module**

* Transfer segmented file info for cell crop image preprocessing and create learned data folders
  + Start the MATLAB software
  + Open script TransferSegmentFiles.m in UPSIDEv1/code/crop\_extraction/AllCellTypes020420/ and edit the segment\_dir, pos\_num, and data\_dir variables with the appropriate directories of the segmented data:

% enter directory of segment file

segment\_dir = 'UPSIDEv1/data/CellTypes020420/CellTypes020420 - segmented/';

% enter the number of positions

pos\_num = 240;

% enter the directory to store all analyzed data

data\_dir = 'UPSIDEv1/data/CellTypes020420/';

* + Run the TransferSegmentFiles script to start segmentation:

>> TransferSegmentFiles

* Generate crops of segmented cells
  + Start MATLAB
  + Open script ExtractTexture.m in directory UPSIDEv1/code/crop\_extraction/CellTypes020420/ and edit the following variables according to your dataset. We are assuming tiff images’ name are in MetaMorph format:

% enter the number of well positions

pos\_num = 240;

%enter the directory of the image crop generation script

script\_dir ='UPSIDEv1/code/crop\_extraction/CellTypes020420/';

%enter the directory of the data files

data\_dir = 'UPSIDEv1/data/CellTypes020420/';

%enter the directory of the raw brightfield images

im\_dir = 'UPSIDEv1/pytorch\_fnet-master/pytorch\_fnet/data/CellTypes020420/';

%enter base name of MM image

basename = 'EXP2';

%enter channel name for the synthetic image

synth\_name = 'w2Camera CellTrace';

%enter segmentation function's name

seg\_name = 'cellseg120518MLCellTrace';

* + Run the image crop extraction script. Cropped images are stored in UPSIDEv1/data/CellTypes020420/SubImTexture . Each cropped cell has a brightfield, a mask, and a texture image:

>> ExtractTexture

* + Combine all the information of extracted crops into one csv file using the python script CombineTextureDirCSV.py in directory UPSIDEv1/code/crop\_extraction/CellTypes020420. Pass the following arguments to the script:  
    -d directory of the extracted crop csvs file  
    -p number of positions  
    -c name of the combined csvs

UPSIDEv1/code/crop\_extraction/CellTypes020420$ python -d CombineTextureDirCSV.py UPSIDEv1/data/CellTypes020420/csvs/ -p 240 -c CombinedTextureDir.csv

* + Annotate the dataset if necessary. Here for this cell type dataset, we will add the trial information and the cell type information to the csv file with the python script assign\_condition.py:  
    UPSIDEv1/code/crop\_extraction/CellTypes020420$ python assign\_condition.py -d UPSIDEv1/data/ CellTypes020420/csvs/CombinedTextureDir.csv -o UPSIDEv1/data/ CellTypes020420/csvs/CombinedDirType.csv

**Part IV. Cell Crop Selection Module**

* Generated cell crops often contain undesired objects such as dead cell, cell debris, and out-of-focus cells. These should be removed as much as possible from the dataset. UPSIDE allows you to do this by two ways: 1) Manually selecting desirable cell crops from the dataset, or 2) Eliminating undesired crops automatically using a trained convolutional classifier. Training dataset needs to be generated manually.
* Method 1: Manually selecting desirable cell crops (recommended for small size (25000 cells) dataset)
  + Open MATLAB
  + Open script selectcelltypes.m in directory UPSIDEv1/code/crop\_selection/CellTypes020420/ and edit the following variables according to your dataset. Refer to the script’s documentation for more detail on how to use it.

% enter directory of the script

code\_dir = 'UPSIDEv1/code/crop\_selection/AllCellTypes020420/';

% enter directory of the data folder

root\_dir = ‘UPSIDEv1/data/CellTypes020420/';

% enter name of the summary csv file

csvfilename = 'CombinedDirType.csv';

% choose specific condition if desirable, enter 0 if no data parsing is

% needed

parse\_data = true;

if parse\_data == true

type = 1;

trial = 1;

end

% choose whether this is the first run

first\_run = true;

* + Here, using the above script, we selected ~ 2000 cells from each cell type and trial and save their indexes in UPSIDEv1/data/CellTypes020420/mat/
  + To extract all cells indexed into a csv file, open script makecsvfromselectcells.m in directory UPSIDEv1/code/crop\_selection/CellTypes020420/ and edit the following variables according to your dataset:

% enter directory of the script

code\_dir = 'UPSIDEv1/code/crop\_selection/AllCellTypes020420/';

% enter directory of the data folder

root\_dir = 'UPSIDEv1/data/CellTypes020420/';

% enter name of the summary csv file

csvfilename = 'CombinedDirType.csv';

% parse data if needed

parse\_data = true;

* + Run the script

>>> makecsvfromselectcells

* Method 2: Automated cell crops using a trained convolutional classifier (recommended for > 25000 collected cells in dataset)
  + The following scripts in UPSDIEv1/data/code/crop\_selection/AML211/ are built for using a convolutional classifier to filter out unwanted cell crops from the dataset. Here, we won’t use this method for the demo dataset. To use the classifier, follow these steps:
    1. Make a csv file with the first column detailing the directory of the brightfield tiff file of the cell crops and the second column detailing 1 for ‘live’ cells and 0 for ‘dead’ cells. You can modify the selectcelltype.m and makecsvfromselectcells.m scripts from the previous part to help with generating this data.
    2. Run the script train\_classifer.py in the command line and pass the following arguments to the script:
       - path of csv file with training data
       - path where learned weights will be saved
       - learning rate
       - number of iteration step
    3. Make a csv file with the first column detailing the directory of the brightfield tiff file to be classified. The following columns should be: position, timepoint, obj, trial information for each cell.
    4. Run the script predict\_classifer.py in the command line and pass the following arguments to the script:
* path of csv file containing to-be-predicted cells
* main path of the dataset
* path to the trained weights and biases
* path of the output file

**Part IV. Cell Crop Self Learning Module**

* To train the perform unsupervised learning of the demo cell types data using UPSIDE’s variational autoencoder, run the following scripts mask\_learning.py and texture\_learning.py in directory UPSIDEv1/code/unsupervised\_learning/CellTypes020420/ and the pass the following arguments to the script:
  + -d Path of the unsupervised learning data
  + -w Directory where weights will be saved
  + -r Contribution of the reconstruction loss
  + -k Contribution of the KLD loss
  + -l Learning rate
  + -s Number of iterations

UPSIDEv1/code/unsupervised\_learning/AllCellTypes020420$ python mask\_learning.py -d /media/phnguyen/Data2/Imaging/UPSIDEv1/data/CellTypes020420/csvs/CombinedDirTypeChosen.csv -w /media/phnguyen/Data2/Imaging/UPSIDEv1/data/CellTypes020420/mask\_weights/ -r 4096 -k 0.001 -l 0.0001 -s 100000

UPSIDEv1/code/unsupervised\_learning/AllCellTypes020420$ python texture\_learning.py -d /media/phnguyen/Data2/Imaging/UPSIDEv1/data/CellTypes020420/csvs/CombinedDirTypeChosen.csv -w /media/phnguyen/Data2/Imaging/UPSIDEv1/data/CellTypes020420/texture\_weights/ -r 40960 -k 0.000005 -l 0.000005 -s 200000

* Note: To run this script on your own data, modify the scripts CellTypesDataset\_Mask.py and CellTypesDataset\_Texture.py so that they accept the format of your summary data csv file. Additionally, modify the file ToTensorLive.py according to accept same file format. Currently, they are set up to accept arguments with the csv set in the following orders:
  + 1st column: directory name of the cell crop file
  + 2nd column: imaging position of the cell
  + 3rd column: imaging timepoint of the cell
  + 4th column: cell number ID of the cell
  + 7th column: trial information of the cell

Then import these functions into the learning scripts above as well as the encoding script below, your data format is recognized.

* Encode the image crops into mask and texture barcodes using the learned encoders. Run the following scripts mask\_encoding.py and texture\_encoding.py and pass the following arguments to the scripts:
  + -d Directory to the csv file of your cell crop data
  + -r Directory where all the raw data would be saved
  + -w Directory to the learned weights to be used
  + -o Output csv file directory

UPSIDEv1/code/unsupervised\_learning/AllCellTypes020420$ python texture\_encoding.py -d UPSIDEv1/data/CellTypes020420/csvs/CombinedDirTypeChosen.csv -r UPSIDEv1/data/CellTypes020420/NNDataTexture/ -w UPSIDEv1/data/CellTypes020420/texture\_weights/Q\_VAE\_CellTypes\_\_large\_LIVE\_100z\_nodropout\_Texture\_200000.pt -o UPSIDEv1/data/CellTypes020420/csvs/

UPSIDEv1/code/unsupervised\_learning/AllCellTypes020420$ python mask\_encoding.py -d UPSIDEv1/data/CellTypes020420/csvs/CombinedDirTypeChosen.csv -r UPSIDEv1/data/CellTypes020420/NNDataMask/ -w UPSIDEv1/data/CellTypes020420/mask\_weights/Q\_VAE\_CellTypes\_\_large\_LIVE\_100z\_nodropout\_Mask\_60000.pt - UPSIDEv1/data/CellTypes020420/csvs/

* The scripts generate two csv files, one containing the latent dimensions and one containing cell crop’s information (position, timepoints, ect). Run the clusterCells.py script to cluster cells and generate umap representation from the learned dimensions. Pass the following arguments to the script:
  + -d Directory to the csv files
  + -m csv file name carrying identifying information for the mask crops
  + -t csv file name carrying identifying information for the texture crops
  + -a csv file name carrying latent information for the mask crops
  + -x csv file name carrying latent information for the texture crops
  + -o csv output file name

UPSIDEv1/code/unsupervised\_learning/AllCellTypes020420$ python clusterCells.py -d UPSIDEv1/data/CellTypes020420/csvs/ -m root\_CellTypes020420\_VAE\_MaskChosen.csv -t root\_CellTypes020420\_VAE\_TextureChosen.csv -a style\_CellTypes020420\_VAE\_MaskChosen.csv -x style\_CellTypes020420\_VAE\_TextureChosen.csv -o combined\_UMAP\_cluster\_z.csv

* To generate synthetic barcodes, open script generateSyntheticIm.m in the directory /media/phnguyen/Data2/Imaging/UPSIDEv1/code/unsupervised\_learning/AllCellTypes020420 and edit the following variables according to your dataset:

% enter the code directory

code\_dir = 'UPSIDEv1/code/CellTypes020420/unsupervised\_learning/';

% enter the directory for the csv file containing barcode information

root\_dir = 'UPSIDEv1/data/CellTypes020420/csvs/';

% enter the name of the summary csv file

datadirfile = 'combined\_UMAP\_cluster\_z.csv';

% enter the number of properties (features+latent dims) in csvfile

datacolumn = 217;

% enter the mask feature list in brackets

idxM = [0:99];

% enter the texture feature list in brackets

idxT = [0:99];

% enter the output csv file name

outfile = 'decoded\_features.csv';

* Run the script. It will generates two csv files, one containing variation for each of the mask latent dimension and on containing variation of each of the texture latent dimension

>>> generateSyntheticIm

* To generate synthetic decoded images from the barcodes, run the scripts mask\_decoding.py and texture\_decoding.py and edit the following variables according to your dataset:
  + -d The directory of the csv file with barcodes for decoding
  + -o The directory and file name where decoded images will be saved (.png)
  + -v The number of features being changed (number of rows)
  + -s The number of step variation in each changed feature (number of columns)
  + -w The weights file being used for the predictor - MUST be the in conjunction with the weights file used in the encoder

UPSIDEv1/code/unsupervised\_learning/AllCellTypes020420$ python mask\_decoding.py -d UPSIDEv1/data/CellTypes020420/csvs/Mask\_decoded\_features.csv -o UPSIDEv1/data/CellTypes020420/png/Mask\_decoded\_features.png -v 100 -s 3 -w UPSIDEv1/data/CellTypes020420/mask\_weights/P\_VAE\_CellTypes\_\_large\_LIVE\_100z\_nodropout\_Mask\_60000.pt

UPSIDEv1/code/unsupervised\_learning/AllCellTypes020420$ python texture\_decoding.py -d -o UPSIDEv1/data/CellTypes020420/csvs/Texture\_decoded\_features.csv UPSIDEv1/data/CellTypes020420/png/Texture\_decoded\_features.png -v 100 -s 3 -w UPSIDEv1/data/CellTypes020420/texture\_weights/P\_VAE\_CellTypes\_\_large\_LIVE\_100z\_nodropout\_Texture\_200000.pt

**Part V. Cell linking module**

* To generate synthetic barcodes, open script celltrackingondataset.m in the directory UPSIDEv1/code/pairwise\_linking/ and edit the following variables according to your dataset:

% enter the code directory

code\_dir = '/media/phnguyen/Data2/Imaging/UPSIDEv1/code/pairwise\_linking';

% enter the directory of the csv file containing cell information

root\_dir = '/media/phnguyen/Data2/Imaging/UPSIDEv1/data/AML211/csvs';

% enter the csv file name

csvfilename = 'LIVE\_position\_subgate.csv';

cd(code\_dir)

% enter the number of data fields

datacolumn = 211;

% enter the parameters for tracking algorithm:

%D\_now this sets the minimum distance the cell at time t must be away from

%other cells to be considered

%Dlower\_next this sets the maximum boundary away from cell(t) for cell(t+1)

%to be considered candidate partner

%Dupper\_next this sets the minimum boundary the 2nd best partner to cell(t)

%has to be away for the match to be considered valid

D\_now = 40;

Dlower\_next = 60;

Dupper\_next = 70;

%output file name

outfile = 'Live\_tracked.csv';

The algorithm pairs cells together by finding cells from the same position and connecting it to another cell in the same position at the next time point. If it finds a match, it will put the ‘cell’ data of that partner cell into a new field called ‘pcell’. If it cannot find a suitable cell partner for the current cell, the value 0 will be entered into ‘pcell’. Therefore the input csv file should have the following data fields ‘pos’, ‘t’, and ‘cell’