Documentation to the code repository of the paper "**A deep learning framework for nucleus segmentation using image style transfer**"

Citation

Please cite our paper if you use our method:

[...]

Prerequisites

Please see requirements.txt that can also be run as a bash script (Linux) or alternatively, you can copy the install commands to console corresponding to your system (command prompt (Windows) / terminal (Linux)) and execute them.

Note: code was tested on Ubuntu 16.04.4. and Windows 10 with python3.6 virtual environment and Matlab 2017a/b, CUDA 9.0 and CuDNN 7.0 were required.

Note: Matlab is not required for fast prediction (see later).

Data

We recommend using 8-bit 3 channels RGB images in .png format.

Usage

Prediction with post-processing

Predicts nuclei first with a presegmenter Mask R-CNN model, estimates cell sizes, predicts with multiple U-Net models and ensembles the results, then uses all of the above in a final post-processing step to refine the contours.

To predict nuclei on images please do the following steps:

1. Please edit either

    - start_prediction_full.bat (Windows) or

- start_prediction_full.sh (Linux)

and specify the following 3 directories with their corresponding full paths on your system:

- Mask R-CNN: we provide a downloaded version that is compatible with our scripts; in the same folder as this file under Mask_RCNN-2.1 and it is the default folder used by the scripts. *WE STRONGLY RECOMMEND YOU USE THIS FOLDER* however, you can, if you will, specify another folder where you have Mask R-CNN downloaded - in this case, we do not guarantee it will work.
- root_dir: full path of the root folder of our code which also contains README.txt. It is set automatically but can be redefined.
- images_dir: full path of the folder containing your intensity images to segment. Default is the 'testImages' folder we provide to test our method.

2. Open a command prompt (Windows) or terminal (Linux) and run either

- start_prediction_full.bat (Windows)
- start_prediction_full.sh (Linux)

you just edited.

3. Result of prediction will be found under the following relative path:

\kaggle_workflow\outputs\postprocessing\

relative to the root folder containing our code.

Prediction fast

Predicts nuclei with a presegmenter Mask R-CNN model that generalizes and performs well in varying image types. Produces fast results that can be improved with the post-processing option above.

To predict fast:

Please follow the steps of "*Prediction with post-processing*" section for either of the files:

- start_prediction_fast.bat (Windows) or
- start_prediction_fast.sh (Linux)

However, results will be on the following relative path:

\kaggle_workflow\outputs\presegment\

Custom validation usage

To use your custom folder of images as validation please run the following script according to your operating system:

- runGenerateValidationCustom.bat (Windows)

- runGenerateValidationCustom.sh (Linux)

If you wish to use our validation set assembled for Kaggle DSB, please rename the file relative to the root directory of our code

\matlab_scripts\generateValidation\validationFileNames_KAGGLE.mat

to \matlab_scripts\generateValidation\validationFileNames.mat

Training

To train on your own images please follow the steps below.

WARNING: training will override the U-Net models we provide, we advise you make a copy of them first from the following relative path:

\kaggle_workflow\unet\

1. Please run the following script according to your operating system:

   - start_training.bat (Windows)

   - start_training.sh (Linux)

   NOTE: for Windows you need to edit start_training.bat and set your python virtual environment path as indicated prior to running the script. It will open a second command prompt for necessary server running of pix2pix and must remain open until all pix2pix code execution is finished - which is indicated by the message "STYLE TRANSFER DONE:" in command prompt.

2. Resulting Mask R-CNN model will be found on the relative path:

   \kaggle_workflow\outputs\model\mask_rcnn_trained.h5

(3.) If you would like to use this model for prediction, please copy this file to the relative path

\kaggle_workflow\maskrcnn\model\mask_rcnn_final.h5

HOWEVER, it is possible to override our provided model of the same name; we advise you make a copy of it prior to placing your trained model.

Training might take long (up to several hours) if a large number of training images are provided – alternatively, this applies to test images as well regarding style transfer image generation. Training time is also influenced by the graphics card your system has specifically by its VRAM (we trained on Nvidia GTX 1080Ti with 11 GB of VRAM or Nvidia Titan Xp with 12 GB VRAM) as well as training image size: the larger the images the more VRAM required.

NOTE: in some cases Windows multi-threading is somewhat limited compared to Unix-based systems which might result in the following error during pix2pix style transfer:

> socket.error: [Errno 111] Connection refused
>
> or
>
> ConnectionRefusedError: [WinError 10061] No connection could be made because the target machine actively refused it

If you experience such, please rename the file

> \biomag-kaggle\src\2_DL\style-transfer\pytorch-CycleGAN-and-pix2pix-etasnadi\options\base_options_win.py
>
> to
> \biomag-kaggle\src\2_DL\style-transfer\pytorch-CycleGAN-and-pix2pix-etasnadi\options\base_options.py

(both paths are relative to the root folder of our code). We recommend you make a copy of the original files before overwriting them. Further discussion of the issue can be found at

https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix/issues/51

https://github.com/Lawouach/WebSocket-for-Python/issues/130

NOTE: If you have torch>0.4.1 installed a possible python error may arise when executing pix2pix style transfer learning:

> UserWarning: invalid index of a 0-dim tensor.

To overcome this, replace the file

> \biomag-kaggle\src\2_DL\style-transfer\pytorch-CycleGAN-and-pix2pix-etasnadi\models\pix2pix_model.py
>
> with pix2pix_model_newpytorch.py

in the same folder. We recommend you rename the original pix2pix_model.py first to preserve it. For details about this pyTorch issue see

https://discuss.pytorch.org/t/this-will-be-an-error-in-pytorch-0-5/19691

A similar error also corresponding torch>=0.5 may be encountered during U-Net training:

> IndexError: invalid index of a 0-dim tensor. Use tensor.item() in Python or tensor.item<T>() in C++ to convert a 0-dim tensor to a number

In case you have this error, replace the file

> \UNet\wrappers.py

with \UNet\wrappers_newpytorch.py

in the same folder. As in the previous cases, we recommend first renaming the original wrappers.py file to preserve it. For more details about this pyTorch issue please see

https://github.com/NVIDIA/flownet2-pytorch/issues/113

Training details

Mask R-CNN parameters:

- optimizer SGD (Stochastic gradient descent)
- batch size #GPU x #images/GPU according to VRAM <-- typically 1 GPU x 2 images/gpu
- loss binary cross-entropy
- learning schedule 3 epoch groups
    - learning rate 0.001/0.0005/0.0001
    - layers all/5+/heads (/epoch groups)
    - validation custom, pre-defined (to enhance common/rare image types)

U-Net parameters:

- optimizer ADAM (Adaptive Moment Estimation)
- batch size 12
- loss Jaccard
- learning schedule
    - epochs 100
    - validation custom, pre-defined (to enhance common/rare image types)

pix2pix parameters:

- optimizer ADAM (Adaptive Moment Estimation)
- batch size 1
- loss L1-loss
- learning schedule
    - learning rate 0.0002 linearly decreasing to 0
    - iterations 5000/#images + 1000

Parameter searching for post-processing

To find the most optimal parameters for your image set, please run the following code. It will take only validation images to account.

- start_parameterSearch.bat (Windows) or

- start_parameterSearch.sh (Linux)

The output file will contain the found optimal parameters with their corresponding mean IoU scores in the text file

\kaggle_workflow\outputsValidation\paramsearch\paramsearchresult.txt

The parameters can be passed to the post-processing function

\matlab_scripts\postProcess\postProcCodeRunnerFINAL.m in either

- run_workflow_predictOnly_full.bat (Windows) or

- run_workflow_predictOnly_full.sh (Linux)

as its last parameter (default is "[]" which means empty array).

A description of the parameters can be found in the above .m file.

NOTE: by default the optimizer code runs for a time limit of 30 minutes. If you would like to change the time limit, please edit either

- run_workflow_parameterSearch4postProc.bat (Windows) or

- run_workflow_parameterSearch4postProc.sh (Linux)

and set an additional input argument to the function "startPostProcParamSearch" with the time you desire passed in seconds. For example "startPostProcParamSearch(rootDir,60)" will run the optimizer for 60 seconds. If you would like to reset to the default, just leave this parameter out as in "startPostProcParamSearch(rootDir)".

Prepare style transfer input for single experiment

To prepare style transfer learning for your custom masks corresponding to your test images please run either

- start_singleExperimentPreparation.bat (Windows) or

- start_singleExperimentPreparation.sh (Linux)

This will prepare training data structure for subsequent style transfer learning steps for only one group containing all your test images. After completing the preparation you should run either

- start_training_singleExperiment.bat (Windows) or

- start_training_singleExperiment.sh (Linux)

for training instead of start_training.bat/sh which performs style transfer learning based on image clustering.

WARNING: If you do not provide your own mask folder for this step the default option will be \kaggle_workflow\outputs\presegment which is created by either

- start_prediction_fast.bat (Windows) or

- start_prediction_fast.sh (Linux)

NOTE: This option should only be used if all your images come from the same experiment. If you provide mixed data, subsequent style transfer learning will result in flawed models and failed synthetic images.

Data structure

Test images

Your test images to segment should be placed in a single arbitrary folder in your computer. For prediction you only need to pass the full path of this folder to the appropriate scripts. The image files may have different naming conventions as you desire, and file types of either {png, tif, tiff, bmp} for prediction. However, for training the images are expected to be 8-bit .png files with 3 channels for typically RGB. We recommend using 8-bit .png files with 3 channels for all steps.

Test image masks

If you wish to perform style transfer learning on your own masks corresponding to the test images of a **single experiment** please prepare the mask images as follows:

A mask image should contain all annotated nuclei with different labels corresponding to each of them in a multi-labelled 16-bit .tiff image file.

For reference please check the masks under the relative path
/kaggle_workflow/inputs/train_maskrcnn/masks.

To convert masks in the Kaggle DSB 2018 format to our expected format please see the following scripts:

/matlab_scripts/utils/run_preproc_16bit_image.m
convert intensity images to 8-bit 3 channel images as required by our pipeline.

/biomag-kaggle/src/0_preprocessing/matlab/generateDataFromDSBInput.m
converts Kaggle DSB 2018 format masks to our expected mask format.

Folder structure

Let us reference the root folder of our repository as [root].

Codes will be found in the following folders:

[root]/
[root]/biomag-kaggle
[root]/FinalModel
[root]/Mask_RCNN-2.1
[root]/matlab_scripts
[root]/UNet

You do not need to edit these except for those batch or bash scripts in the [root] folder where indicated by the README.

Models will be found in the following folders:

[root]/kaggle_workflow/maskrcnn/model
[root]/kaggle_workflow/unet

Image files provided as ***samples*** will be found in the following folders:

[root]/kaggle_workflow/inputs/clustering/masks
[root]/kaggle_workflow/inputs/test1/images
[root]/kaggle_workflow/inputs/test1/masks
[root]/kaggle_workflow/inputs/train_maskrcnn/images
[root]/kaggle_workflow/inputs/train_maskrcnn/masks
[root]/kaggle_workflow/inputs/train_unet/images
[root]/kaggle_workflow/inputs/train_unet/masks
[root]/kaggle_workflow/inputs/validation/images
[root]/kaggle_workflow/inputs/validation/masks

You should copy your images to the corresponding folder as indicated by the folder name.

Image files generated by our pipeline will be found in the following folders:

[root]/kaggle_workflow/outputs/2ximages/images
[root]/kaggle_workflow/outputs/2ximages/masks
[root]/kaggle_workflow/outputs/augmentations/base/images
[root]/kaggle_workflow/outputs/augmentations/base/masks
[root]/kaggle_workflow/outputs/augmentations/style/images
[root]/kaggle_workflow/outputs/augmentations/base/masks
[root]/kaggle_workflow/outputs/clusters/Kmeans-correlation-Best3Cluster/*
[root]/kaggle_workflow/outputs/clusters/Kmeans-correlation-Best3Cluster__MASKS/*
[root]/kaggle_workflow/ensemble/output
[root]/kaggle_workflow/outputs/images
[root]/kaggle_workflow/outputs/presegment
[root]/kaggle_workflow/outputs/styleLearnInput/**
[root]/kaggle_workflow/outputs/test1/images
[root]/kaggle_workflow/outputs/test1/masks
[root]/kaggle_workflow/outputs/train_maskrcnn/images
[root]/kaggle_workflow/outputs/train_maskrcnn/masks
[root]/kaggle_workflow/outputs/train_unet/images

[root]/kaggle_workflow/outputs/train_unet/masks
[root]/kaggle_workflow/outputs/unet_out/***
[root]/kaggle_workflow/outputs/validation/images
[root]/kaggle_workflow/outputs/validation/masks

*: Results of clustering that is images sorted to clusters indicated by the folders “group_x”. Additional folders are for checking the inputs.

**: Prepared inputs and results of style transfer learning. See details below.

***: Results of U-Net predictions by each model. These predictions are ensembled in [root]/kaggle_workflow/ensemble/output

Image files used in style transfer learning will be found in the following folders relative to [root]/kaggle_workflow/outputs/styleLearnInput/:

/0/generated/group_x/grayscale	- the generated synthetic mask image files
/0/input-clusters/group_x	- the clustered original images
/0/p2ptrain/group_x/train	- paired train data prepared for pix2pix style transfer learning
/0/preds-clusters/group_x	- the clustered original masks
/0/p2pmodels/group_x	- the trained pix2pix models
/0/p2psynthetic/group_x	- the resulting style transferred fake images
/0/out/images	- the results collected for all clusters
/0/out/masks	- the results collected for all clusters

/1/ will contain the same structure of folders. Clusters are split to 2 parts (/0/ and /1/) to allow parallel processing. The current script processes them sequentially.