ERNet - Segmentation of Endoplasmic Reticulum microscopy images

Meng Lu[1], Francesca W. van Tartwijk[1], Julie Qiaojin Lin[1], Wilco Nijenhuis, Pierre Parutto, Marcus Fantham[1], Charles N. Christensen[1,*], Edward Avezov, Christine E. Holt, Alan Tunnacliffe, David Holcman, Lukas C. Kapitein, Gabriele Kaminski Schierle[1], Clemens F. Kaminski[1]

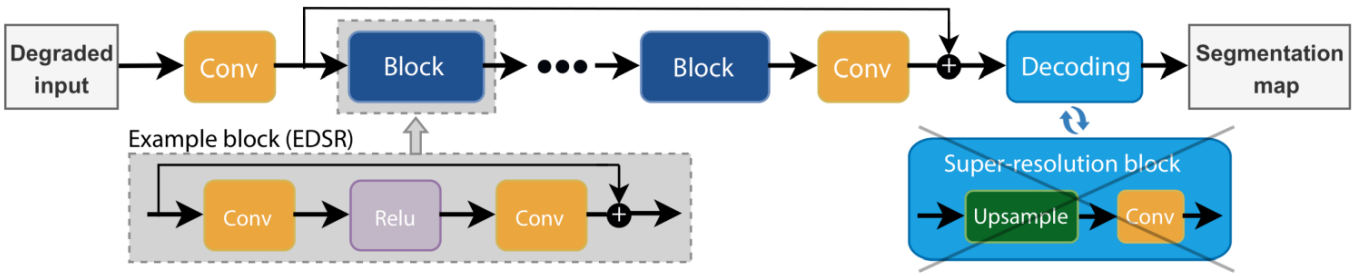
[1]University of Cambridge, Department of Chemical Engineering and Biotechnology

[*]Author of this repository - GitHub username charlesnchr - Email address charles.n.chr@gmail.com

Paper with results based on the code in this repository:
https://www.biorxiv.org/content/10.1101/2020.01.15.907444v2

Introduction

ERNet is an artificial neural network based model for segmentation of endoplasmic reticulum microscopy images. The network architecture of choice is a deep residual network inspired by EDSR and RCAN (Lim et al. 2017, Zhang et al., 2018). The modified architecture is shown in the image below with a block corresponding to EDSR.



The architectures of these models are among several residual learning networks (cite ResNet, SRGan, EDSR) designed for image restoration, specifically single image super-resolution (SR), i.e. image upsampling. The state-of-the-art SR architectures generally do not use downsampling between layers, but instead make training of deep networks feasible by following the structure of residual networks as first introduced with ResNets intended for image classification. The design idea of residual networks was taken one step further in EDSR with the proposal of a modified residual building block called ResBlock, which was found to be superior to the previously proposed and more directly adapted ResNet model called SRResNet

Choosing the first part of our segmentation model to have an architecture built for restoration ensures that it is capable of handling low signal-to-noise ratio as it can learn to perform denoising in these early layers of its network. A neural network model intended for image restoration will by default perform regression in order to output pixel value predictions in the same colour space as the input image. This is achieved during model training by minimising an appropriate loss function, typically the mean squared error.

Installation

This implementation requires Pytorch. We have tested ERNet with Python 3.6 and 3.7 and Pytorch 1.2 and 1.4.

If you start from scratch...

We recommend using miniconda. If you do not yet have a strong opinion, just use it too!

After installing Miniconda, the following lines might are likely the easiest way to get Tensorflow and CuDNN installed on your machine (Note: Macs are not supported, and if you sit on a Windows machine all this might also require some modifications.):