TÉCNICO LISBOA

Deep Learning Methods for the Segmentation of Fluorescent Microscopy Images

Alice Henriques da Rosa

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Maria Margarida Campos da Silveira
Prof. Hemaxi Narotamo

Examination Committee

Chairperson: Prof. Name of the Chairperson
Supervisor: Prof. Maria Margarida Campos da Silveira
Members of the Committee: Prof. Name of First Committee Member
Dr. Name of Second Committee Member
Eng. Name of Third Committee Member

Month 20XX

Declaration

I declare that this document is an original work of my own authorship and that it fulfils all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to thank my parents for their friendship, encouragement and caring over all these years, for always being there for me through thick and thin and without whom this project would not be possible. I would also like to thank my grandparents, aunts, uncles and cousins for their understanding and support throughout all these years.

I would also like to acknowledge my dissertation supervisors Prof. Some Name and Prof. Some Other Name for their insight, support and sharing of knowledge that has made this Thesis possible.

Last but not least, to all my friends (except Aprígio Malveiro because is a genio e vai ter o nome no trabalho) and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life. Thank you.

To each and every one of you – Thank you.

Abstract

Nulla facilisi. In vel sem. Morbi id urna in diam dignissim feugiat. Proin molestie tortor eu velit. Aliquam erat volutpat. Nullam ultrices, diam tempus vulputate egestas, eros pede varius leo, sed imperdiet lectus est ornare odio. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Proin consectetuer velit in dui. Phasellus wisi purus, interdum vitae, rutrum accumsan, viverra in, velit. Sed enim risus, congue non, tristique in, commodo eu, metus. Aenean tortor mi, imperdiet id, gravida eu, posuere eu, felis. Mauris sollicitudin, turpis in hendrerit sodales, lectus ipsum pellentesque ligula, sit amet scelerisque urna nibh ut arcu. Aliquam in lacus. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nulla placerat aliquam wisi. Mauris viverra odio. Quisque fermentum pulvinar odio. Proin posuere est vitae ligula. Etiam euismod. Cras a eros.

Keywords

Maecenas tempus dictum libero; Donec non tortor in arcu mollis feugiat;Cras rutrum pulvinar tellus.

Resumo

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi. Aenean fermentum, elit eget tincidunt condimentum, eros ipsum rutrum orci, sagittis tempus lacus enim ac dui. Donec non enim in turpis pulvinar facilisis. Ut felis. Aliquam aliquet, est a ullamcorper condimentum, tellus nulla fringilla elit, a iaculis nulla turpis sed wisi. Fusce volutpat. Etiam sodales ante id nunc. Proin ornare dignissim lacus. Nunc porttitor nunc a sem. Sed sollicitudin velit eu magna. Aliquam erat volutpat. Vivamus ornare est non wisi. Proin vel quam. Vivamus egestas. Nunc tempor diam vehicula mauris. Nullam sapien eros, facilisis vel, eleifend non, auctor dapibus, pede.

Palavras Chave

Colaborativo; Codificaçãoo; Conteúdo Multimédia; Comunicação;

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ACMs	Active Contour Models
ACM	Active Contour Model
ADAM	ADAptive Momentum Estimation
AJI	Aggregated Jaccard Index
AD-GAN	Aligned Disentangling Generative Adversarial Network
AdaDelta	Adaptive Delta
AdaGrad	Adaptive Gradient Descent
BN	Batch Normalization
CAD	Computer-Aided Diagnosis
cGAN	Conditional Generative Adversarial Network
CNNs	Convolutional Neural Networks
CNN	Convolutional Neural Network
DC	Dice Coefficient
DLoss	Dice Loss
DL	Deep Learning
ELU	Exponential Linear Unit
FCN	Fully Convolutional Network
FC	Fully Connected
GANs	Generative Adversarial Networks
GAN	Generative Adversarial Network
IoU	Intersection over Union
Leaky ReLU	Leaky Rectified Linear Unit
NN	Neural Network

non-PDE	nonlinear partial differential equation
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
Tanh	Hyperbolic Tangent
2D	two dimensional
3D	three dimensional

1

Introduction

Contents

1.1 Motivation

Digital pathology and microscopic image analysis are widely used for comprehensive studies of cell morphology or tissue structure and play an important role in decision making in disease diagnosis. The extensive information provided by these studies can be used in Computer-Aided Diagnosis (CAD). Computer-assisted methods can improve objectivity and provide accurate characterization of diseases [1].

Cell detection and segmentation is critical for CAD as it supports various quantitative analyses, including calculation of cell morphology, e.g., size, shape, and texture. This information is essential for the analysis, diagnosis, classification and grading of cancer, for example [2].

In recent years, the development of fluorescence microscopy has allowed a better study of cells and its substructures by allowing the acquisition of three dimensional (3D) image volumes that reach deeper into the tissue [3] and are therefore very important sources of information for CAD.

Manual analysis of large image sets is labor intensive, time consuming, and can be biased by individuals. Therefore, automatic two dimensional (2D)/3D microscopy image analysis methods, especially segmentation, have been intensively studied by researchers [4] to achieve more efficiency and accuracy in quantifying and characterizing cells, nuclei, or other biological structures, and thus improving diagnosis.

Recently, Deep Learning (DL) has emerged as a powerful tool that is attracting much interest in microscopic image analysis [5] due to its ability to automatically learn features from raw data. This enables the development of more robust and accurate algorithms for microscopic image segmentation.

However, the performance of these methods is limited by the amount of manually labelled ground truth data available to train the network, and the generation of this data, especially for 3D volumes, is tedious and very time consuming. Therefore, interest in weakly/fully unsupervised DL models that can be trained on data without annotations and still perform well on unseen data has increased greatly in recent years [5].

1.2 Objectives

In this work, deep learning methods are implemented for the segmentation of fluorescent microscopic images. Three models are implemented, two supervised, inspired by the 3D U-Net [6] and Vox2Vox [7] architectures, and one unsupervised, inspired by the CycleGAN model [8]. The dataset used to train, validate, and test these models includes 3D fluorescence microscopy images of mouse retinas with two cellular components: Golgi and Nuclei and the corresponding manually labeled 3D segmentation masks. The aim is that these models can predict from the 3D fluorescence microscopy images the 3D

segmentation masks with n binary segmentation maps, where n is the number of classes we want to classify. This work has two goals:

- the main goal is to develop an unsupervised deep learning-based algorithm that is able to accurately segment the nucleus and Golgi of cells as well as or better than the supervised methods. For this unsupervised method, synthetic 3D segmentation masks have to be created for training the model;
- the secondary goal is to explore an approach that consists in considering, as in [9], three classes: the nuclei, the Golgi, and the nucleus-Golgi pairs, with the goal of studying whether the segmentation of the nuclei and Golgi improves, i.e., whether we can avoid, for example, the detection of noise. Furthermore, the detection of nucleus-Golgi pairs may have many applications, such as counting the number of nucleus-Golgi pairs and estimating the nucleus-Golgi vectors, which are fundamental tasks to study the migration process of cells in the formation of blood vessels in the mouse retina [9].

1.3 Thesis Outline

Chapter 2 introduces the background of deep learning algorithms used for computer vision tasks. It begins with an explanation of how Convolutional Neural Networks (CNNs) work and then describes the Convolutional Neural Network (CNN) architecture for image segmentation in 2D and 3D. This chapter also explains how CNNs are trained. After that, the 2D and 3D U-Net architectures are explained. The chapter ends with an explanation of Generative Adversarial Networks (GANs) and examples of supervised and unsupervised models. Chapter 3 reviews the current state-of-the-art in microscopic image segmentation. It begins with the challenges associated with this task, followed by classical approaches, then methods using CNNs and U-Nets, and finally methods using Generative Adversarial Network (GAN). Chapter 4 begins with a description of the dataset to be used for the proposed models, then follows with a description of the proposed methods to achieve the goals of the dissertation and the description of the metrics used to measure the performance of these methods. In chapter 5, the experimental results regarding the segmentation of cell nuclei and Golgi in fluorescence microscopy images obtained with the proposed models are presented and discussed. Finally, in chapter 6 final conclusions and topics for future studies are presented.

2

Theoretical Background

Contents

Computer vision is a field of artificial intelligence that allows computers to understand, identify, and develop an intelligent understanding of digital images, just like human vision. It is used for many different tasks, most notably:

- **Feature identification** in an image, such as edge detection;
- **Object detection** to classify all the different objects in an image along with their position;
- **Image segmentation**, i.e., assigning each pixel value in an image to a particular class;
- **Neural Style Transfer**, i.e., a new image is generated by learning the style from one image and applying it to another image.

In recent years, most tasks related to computer vision, such as those mentioned, have been solved using modern deep learning architectures, such as CNN [10]. In this chapter, the main DL algorithms developed for computer vision tasks, more specifically for image segmentation tasks, are explained in detail.

2.1 Convolutional Neural Networks

2.1.1 CNN building blocks

A Convolutional Neural Network is a class of Neural Network (NN) that specializes in processing data with a grid-like topology, such as images. Its built-in convolutional layer reduces the high dimensionality of images without losing their information. Therefore, CNNs are best suitable for this use case.

A digital image is a binary representation of visual data. It contains a set of pixels arranged in a grid-like fashion and containing pixel values that indicate how bright and what color each pixel should be.

A CNN is typically composed of three types of layers: convolutional layers, pooling layers and fully connected layers.

Convolutional Layer

In a convolutional layer a filter or kernel ”slides” over the input image and performs an element-wise dot product. As a result, it will be summing up the results into a single output pixel. The kernel will perform the same operation for every location it slides over, producing a two-dimensional representation of the image called a feature map. The size of the filters is usually smaller than the actual image, but the depth is equal to the number of channels [11]. Figure 2.1 shows an example of the convolution process.

If we have an image with multiple channels (e.g. RGB images have 3 channels), we have one 2D kernel per input channel and we perform each convolution, of the 2D input channel with the 2D kernel, separately and then sum the contributions with the bias, resulting in the final output feature map [11].

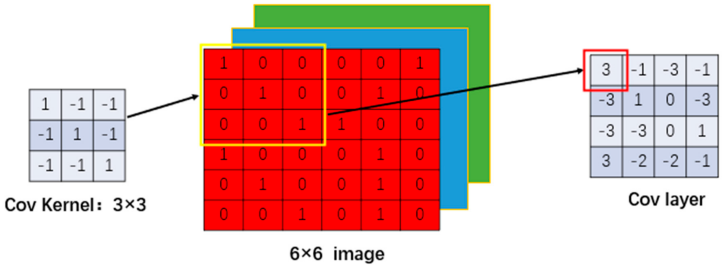


Figure 2.1: Example of the convolution process. Retrieved from [12].

Here we can also define the terms stride and padding. Stride is the number of pixels shifts over the input matrix. If the stride is 1, we move the filters by 1 pixel at a time. In Figure 2.1, we can see that the size of the output is smaller than that of the input. To keep the dimension of the output equal to the input, padding is used where zeros are symmetrically added to the input matrix [13].

The goal of the convolution operation is to extract high-level features from the input image. The more convolutional layers or the more kernels used in each convolutional layer, the higher the number of features extracted.

Pooling Layer

In most cases, a convolutional layer is followed by a pooling layer. The main goal of this layer is to reduce the size of the convolved feature map in order to reduce the computational costs. This is performed by sliding a two-dimensional filter over each channel of the feature map and summarising the features that lie within the region covered by the filter. There are several types of pooling operations, but the most commonly used is the max pooling which returns the maximum value from the portion of the image covered by the filter [11].

Activation Functions

One of the main parameters in a CNN model is the activation function. This nonlinear function can be interpreted as a selection mechanism that decides whether a particular neuron should fire or not given its input. They are often placed directly after the convolutional layer to introduce non-linearity into the feature map.

There are several commonly used activation functions such as the sigmoid, Hyperbolic Tangent (Tanh), Rectified Linear Unit (ReLU), Leaky Rectified Linear Unit (Leaky ReLU) and Exponential Linear Unit (ELU) functions. The most commonly used function in neural networks is ReLU (Equation 2.1)

because it is very computationally efficient, activating only a few neurons per time, it does not saturate in the positive domain, and it converges faster than the tanh and sigmoid activation functions [14].

$$relu(x) = \max\{0, x\} \tag{2.1}$$

Fully Connected Layer

Fully Connected (FC) Layers usually form the last few layers of the network. The input to the fully connected layer is the output of the last pooling or convolutional layer, which is flattened and then fed into the fully connected layer. The flattened vector then passes through a few more FC layers, where the mathematical functions are normally processed. After passing through the fully connected layers, the logistic or softmax activation function is used in the last layer to determine the probability that the input belongs to a certain class (classification) [13]. Figure 2.2 shows a diagram depicting the basic CNN architecture for image classification:

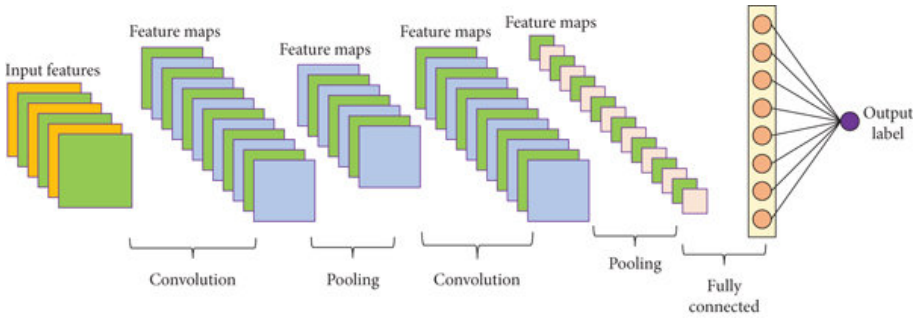


Figure 2.2: Basic CNN architecture for image classification. Retrieved from [15]

2.1.2 CNN architecture for Segmentation

The CNN architecture for segmentation uses encoder and decoder models. The encoders are used to downsample the spatial resolution of the input to develop a lower resolution feature mapping. The decoder then takes this feature representation as input and upsamples it to a full resolution segmentation map. The encoders can be convolutional neural networks and the decoders can be based on the deconvolutional or transposed neural networks with the purpose of creating a segmentation map [16]. Figure 2.3 shows the basic CNN architecture for image segmentation.

3D CNN

In 3D CNN architectures, the 2D convolutional and pooling layers are replaced by 3D convolutional and pooling layers. A 3D CNN can extract more features of the volumes on the three axes X, Y, and Z.

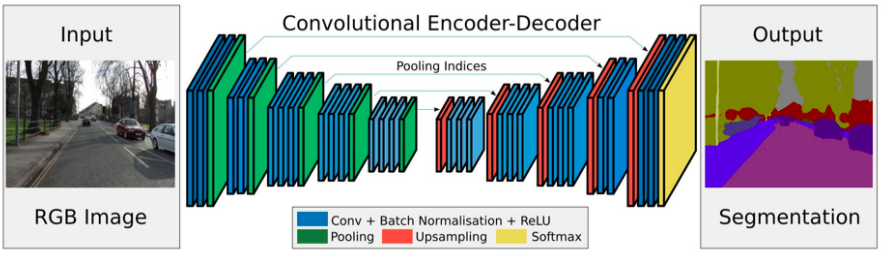


Figure 2.3: Example of CNN architecture for image segmentation. Retrieved from [17].

Therefore, the use of 3D information in segmentation enables the full exploitation of spatial information.

The 3D convolution kernel has one more depth than the 2D convolution kernel, the fourth dimension continues to correspond to the number of channels of the input image. Like the 2D convolution operation, a value is obtained by sliding the kernel on the height, width, and number of layers on each channel [11]. In figure 2.4 the process of 3D CNN convolution for multichannels is shown.

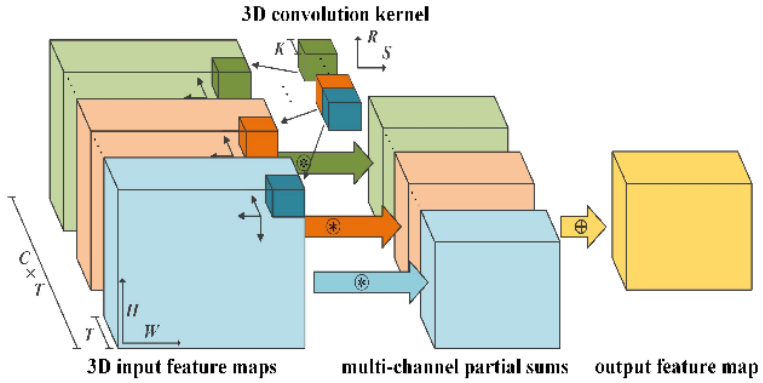


Figure 2.4: Illustration of a 3D convolution layer. Retrieved from [18].

2.1.3 CNN Training

For training a neural network, the following steps are usually performed:

- define the loss function to be minimized during training. For example, in image segmentation tasks, the most commonly used loss function is a pixel-wise cross entropy loss;
- select the optimizer that minimizes the loss function by defining how the parameters of the NN are updated. The most commonly used optimizers are Stochastic Gradient Descent (SGD), SGD with momentum, Adaptive Gradient Descent (AdaGrad), Adaptive Delta (AdaDelta), RMSProp, and ADAptive Momentum Estimation (ADAM). In these optimizers, the learning rate, which controls how much the model should change in response to the estimated error each time the model weights are updated, is a hyperparameter that must be selected;

- split our dataset into a training, validation, and testing dataset. Use the training dataset to adjust the parameters of the model (e.g., the weights and bias in an NN), use the validation dataset to test the model with the parameters selected during the training phase, and also use it to stop training when it stops improving (early stopping) and to adjust the hyperparameters. The test data set is used to provide an unbiased evaluation of the final fit of the model to the training data set;
- define the maximum number of epochs, one epoch corresponds to one run over the whole training set;
- define batch size, the amount of data included in each sub-epoch weight change is called the batch size. Full-batch learning uses all examples of the training set in an epoch, mini-batch learning uses only a subset of the training set, and online learning uses only one example of the training set. For example, for a training set of 100 examples, a full batch size would be 100, a mini-batch size would be 50 or 20 or 10, and an online batch size would be only 1.

2.1.4 U-Net

U-Net is a convolutional neural network developed by Ronneberger et al. [19] for biomedical image semantic segmentation. The network is based on the Fully Convolutional Network (FCN) [20] and its architecture was modified and extended to work with fewer training images and to yield more precise segmentation.

U-Net Architecture

The U-Net model takes its name from the fact that its architecture is in the shape of a 'U'. This architecture consists of three sections: the contraction (encoder), the bottleneck, and the expansion section (decoder). The encoder follows the typical architecture of a convolutional network. It consists of the repeated application of two unpadded 3x3 convolutions, each followed by a rectified linear unit and a 2x2 max-pooling operation with stride 2 for downsampling. At each downsampling step the number of feature channels is doubled. On this contraction path, the model captures the important features of the image and discards the unimportant ones, reducing the resolution of the image at each convolution+maxpool layer. The bottommost layer (bottleneck) mediates between the contraction layer and the expansion layer. It uses two 3x3 CNN layers followed by 2x2 up-convolution layer.

In the expansive path, each step consists of: an upsampling of the feature map, followed by a 2x2 convolutional layer (up-convolution) that reduces the number of feature maps by half to preserve symmetry; a skip-connection and concatenation with the correspondingly cropped feature map from the contracting path, this allows valuable details learned in the encoder part to be used to construct an image in the decoder part; and two 3x3 convolutions, each followed by a ReLU. Cropping is necessary

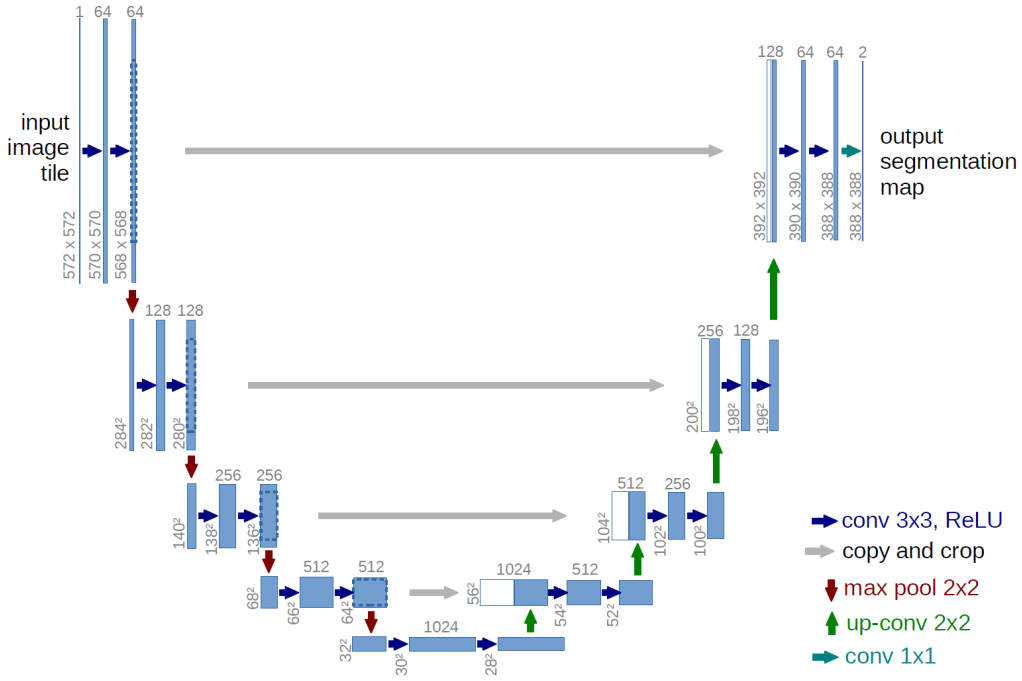


Figure 2.5: U-Net architecture. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. Retrieved from [19].

because edge pixels are lost with each convolution. In the last layer, a 1x1 convolution is used to assign each 64-component feature vector to the desired number of classes.

Loss function

The loss function is computed by a pixel-wise softmax over the final feature map in combination with the cross-entropy loss function. In addition, the authors introduce a weighting map into the loss function, where each pixel is assigned a weight and pixels on the boundary of segmented objects (cells in this case) have a higher weight. In this way, the network is ”forced” to learn the boundary pixels.

2.1.5 3D U-Net

Çiçek et al. [6] proposed a 3D U-Net model, as an extension of the original U-Net [19], with the objective of making the U-Net structure have richer spatial information. The network structure is shown in figure 2.6.

Similar to the 2D U-Net, the 3D U-Net architecture consists of an analysis path (left) and a synthesis path (right). The main difference is in the size of the kernels applied in the different layers (see subsection 3D CNN for an explanation and illustration of a 3D convolutional layer). Here 3x3x3 convolutions and

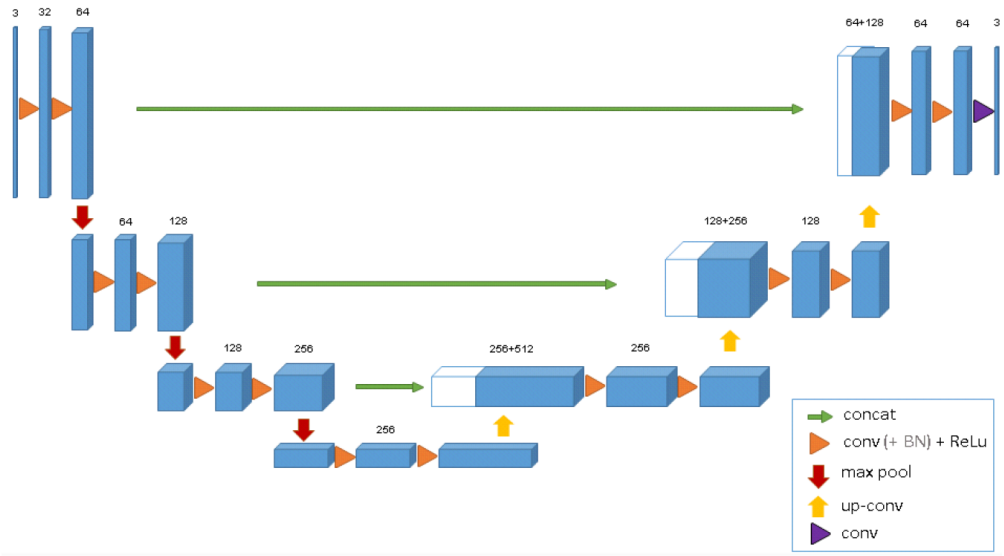


Figure 2.6: 3D U-Net architecture. Retrieved from [6].

up-convolutions and 2x2x2 max-pooling are used, and a $1\times1\times1$ convolution reduces the number of output channels to the number of labels.

The authors also introduced Batch Normalization (BN) before each ReLU. Batch normalization is a technique for training deep neural networks, where the inputs for a layer are standardized for each mini-batch. This helps to speed up the training and stabilize the learning process.

In addition, the authors continue to use softmax with weighted cross entropy as the loss function.

2.2 Generative Adversarial Networks

In 2014, Goodfellow et al. [21] introduced Generative Adversarial Networks. GANs are an approach to generative modeling using deep learning methods, such as convolutional neural networks. Generative models capture, for a given set of data instances X and a set of labels Y, the joint probability $p(X, Y)$. In GANs, a generative model is trained to generate realistic samples of the input data on which they have been trained.

More specifically, GANs are algorithmic architectures inspired by game theory in which 2 neural networks, a generator and a discriminator, compete to reinforce each other.

In a GAN, the generator (G) is the neural network that learns the underlying distribution of the data. It receives as input a noise variable and outputs a synthetic sample. The discriminator (D) is a binary classifier that receives images as input and outputs the probability that the image is real (i.e., from the actual training set) or fake (i.e., from the generator). In this approach, the generator is trained to capture the distribution of the real data so that its generative samples are as close as possible to the real ones,

or in other words, make the discriminator classify them as coming from the real dataset. The architecture of a GAN model is shown in Figure 2.7.

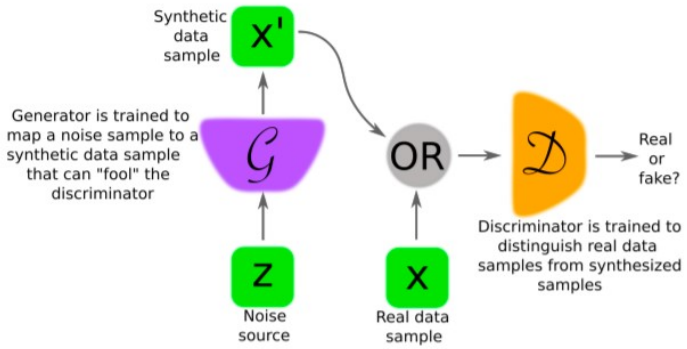


Figure 2.7: Schematic of the GAN model. Where G and D are usually implemented as neural networks. Retrieved from [22].

2.2.1 Training GANs

The generator and discriminator of GANs may have different architectures. However, since GANs usually work with image data, they often use CNNs as generator and discriminator models.

Loss Function

To train a GAN, we first need to define the loss functions of the generator and discriminator. Some notations to keep in mind are:

- $p_{data}(x)$: the distribution of real data
- x: sample from $p_{data}(x)$
- $p(z)$: distribution of random input
- z: sample from $p(z)$
- $G(z)$: Generator Network
- $D(x)$: Discriminator Network

Since the discriminator is a binary classifier, it can be trained using the binary cross entropy loss function. As mentioned earlier, the discriminator aims to maximize the probability assigned to real and fake images. In mathematical terms, the discriminator seeks to maximize the average of the log

likelihood for real images and the log value of the inverted likelihoods for fake images. Therefore, the loss function of the discriminator over a batch is given by:

$$\max[\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]]. \tag{2.2}$$

The generator seeks to minimize the log of the inverse probability predicted by the discriminator for fake images. This has the effect of encouraging the generator to generate samples that have a low probability of being fake. This can be described as the following:

$$\min[\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]]. \tag{2.3}$$

With this, we can write the final loss function as:

$$\min_G \max_D [\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]]. \tag{2.4}$$

This means that the discriminator parameters (defined by D) maximize the loss function and the generator parameters (defined by G) minimize the loss function.

Training steps

The original GAN paper provides a pseudo-code that shows how a GAN is trained, this is shown in algorithm 2.1.

Algorithm 2.1: Minibatch stochastic gradient descent training of generative adversarial networks. Retrieved from [21].

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\left\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\right\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\right\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

  end

  - Sample minibatch of m noise samples $\left\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\right\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right).$$

end

In the algorithm 2.1 it's evident that the GAN training proceeds in alternating periods:

1. The discriminator trains for k steps (k=1 in [21]), and during this period the generator is not trained;
2. The generator is updated only after k steps, and during this period the discriminator is not trained;
3. Repeat steps 1 and 2, for a specified number of iterations, to further train the generator and discriminator networks.

2.2.2 Supervised GAN example: Pix2Pix

In recent years, several GAN models have been proposed to perform different tasks. One of these models is Pix2Pix, which was introduced by Isola et al. [7] to accomplish the task of image-to-image translation.

Pix2Pix is an implementation of the Conditional Generative Adversarial Network (cGAN), where the generation of an image depends on another given image. The general architecture of a cGAN model is shown in Figure 2.8.

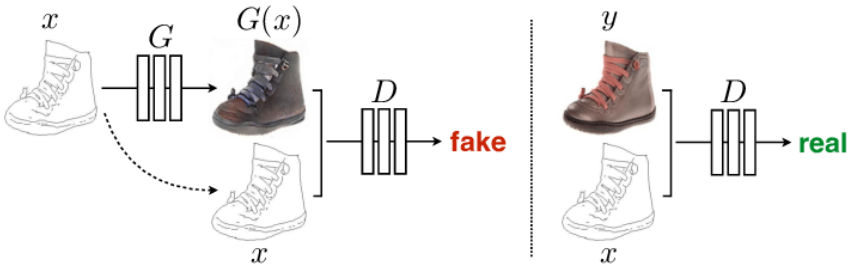


Figure 2.8: Schematic of the cGAN model. Retrieved from [7].

As shown in Figure 2.8, the generator model receives a given image as input and generates a translated version of the image. The discriminator model receives an input image from the source domain and an image from the target domain and must determine whether the image from the target domain is a real or generated version of the source image. Finally, the generator model is trained to both fool the discriminator model and minimize the loss between the generated image and the expected target image. Therefore, the Pix2Pix GAN must be trained on image datasets consisting of input images (before translation) and output or target images (after translation).

Generator and discriminator architecture

In [7], a U-Net model is used for the generator, and, after well trained, it takes as input an image from the source domain and outputs an image in the target domain.

For the discriminator, the authors chose the PatchGAN discriminator. The PatchGAN, also called the Markov discriminator, classifies individual (N x N) patches in the image as real or fake, rather than classifying the entire image as real or fake. The output of the network is a single feature map of real/fake

predictions that can be averaged to give a single classification score. The advantage of using a PatchGAN over a normal GAN discriminator is that it has fewer parameters than a normal discriminator and can work with images of any size.

Loss Function

The conditional adversarial Loss is defined as follows:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log (1 - D(x, G(x, z)))], \tag{2.5}$$

where x is the input observed image, y is the real target image, z is a random noise vector and $G(x, z)$ is the output generated image.

The generator model is trained using both the adversarial loss for the discriminator model and the L_1 or mean absolute pixel difference between the generated translation of the source image and the expected target image. The adversarial loss affects whether the generator model can output images that are plausible in the target domain, while the L_1 loss regularizes the generator model to output images that are plausible translations of the source image.

The L1 loss function is shown below:

$$\mathcal{L}_1(G) = \mathbb{E}_{x,y,z}[||y - G(x, z)||_1]. \tag{2.6}$$

Combining these functions from 2.5 and 2.6, results in the final objective function:

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda_1 \mathcal{L}_1(G) \tag{2.7}$$

where λ_1 is a hyperparameter that controls the relative importance of the L1 loss term.

2.2.3 Unsupervised GAN example: CycleGAN

As shown, the Pix2Pix model is capable of solving the image-to-image translation problem, but only if we have a paired training dataset available, i.e., a large dataset with many examples of input images X and the same image with the desired change that can be used as the expected output image Y. However, obtaining this training data is difficult in many cases.

In 2017, Zhu et al. [8] proposed a novel approach to address the unpaired image-to-image translation problem, called CycleGAN, which uses a GAN architecture.

As can be seen in Figure 2.9, CycleGAN consists of two generator models, one generator (Generator 1) synthesizes images for the domain X to domain Y and the other generator (Generator 2) synthesizes

images for the domain Y to domain X. Each generator has a corresponding discriminator model (Discriminator 1 and Discriminator 2). The discriminator model (1/2) takes real images from domain (Y/X) and generated images from the Generator (1/2) and predicts whether they are real or fake.

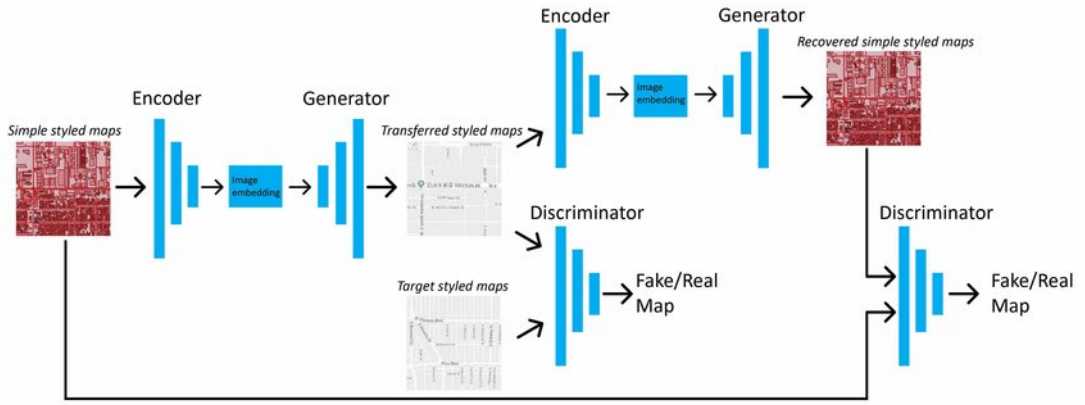


Figure 2.9: Example of schematic of the CycleGAN model. Retrieved from [23].

Loss function

The loss function used to train the generators consists of three parts: adversarial loss, cycle consistency, and identity loss. The adversarial loss is applied to both generators. For the first GAN (generator 1, discriminator 1), the adversarial loss is expressed as follows:

$$\mathcal{L}_{GAN}(G_1, D_1, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_1(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log (1 - D_1(G_1(x)))]. \tag{2.8}$$

For the second GAN (generator 2, discriminator 2) is expressed as:

$$\mathcal{L}_{GAN}(G_2, D_2, Y, X) = \mathbb{E}_{x \sim p_{data}(x)}[\log D_2(x)] + \mathbb{E}_{y \sim p_{data}(y)}[\log (1 - D_2(G(y)))]. \tag{2.9}$$

As in the original GAN model, the generators try to minimize these losses, while the respective discriminators try to maximize them. Cycle consistency loss is a new term introduced to ensure that we preserve the original image when translating it from one domain to another and back again. Therefore, it calculates the L_1 loss between the original image and the generated one in two directions: $x \rightarrow G_1(x) \rightarrow G_2(G_1(x)) \approx x$ and $y \rightarrow G_2(y) \rightarrow G_1(G_2(y)) \approx y$. The cycle consistency loss is expressed as follows:

$$\mathcal{L}_{cyc} = \mathbb{E}_{x \sim p_{data}(x)}[||G_2(G_1(x)) - x||_1] + \mathbb{E}_{y \sim p_{data}(y)}[||G_1(G_2(y)) - y||_1]. \tag{2.10}$$

Finally, the identity loss is an additional optional term used mainly to preserve the color composition between input and output. It is computed by giving the generator an image of its target domain as input

and computing the L_1 loss between input and the generated images. This loss is expressed as follows:

$$\mathcal{L}_{id}(G_1, G_2) = \mathbb{E}_{y \sim p_{data}(y)}[||G_1(y) - y||_1] + \mathbb{E}_{x \sim p_{data}(x)}[||G_2(x) - x||_1]. \tag{2.11}$$

2.3 Metrics to evaluate semantic segmentation models

Various metrics are used to validate and test the performance of these segmentation models, the most important of which are described below.

Pixel Accuracy is the percentage of pixels in the image that were correctly classified. Here, a true positive/false positive (TP/FP) value represents a pixel that was correctly/incorrectly predicted as belonging to a particular class (according to the target mask), while a true negative/false negative (TN/FN) value represents a pixel that was correctly/incorrectly predicted as not belonging to a particular class.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.12}$$

Precision describes the *purity* of positive predictions with respect to ground truth, i.e., it measures, of all the pixels predicted in a given image, how many of those pixels actually match with ground truth.

$$Precision = \frac{TP}{TP + FP} \tag{2.13}$$

Recall describes the *completeness* of positive predictions with respect to the ground truth, i.e., of all pixels annotated in the ground truth, it measures how many of these pixels were capture as positive predictions.

$$Recall = \frac{TP}{TP + FN} \tag{2.14}$$

Intersection over Union (IoU) measures the similarity between two images in semantic segmentation by computing the ratio between the overlapped area and the combined area of prediction and ground truth.

$$IoU = \frac{TP}{TP + FP + FN} \tag{2.15}$$

Dice Coefficient (DC) (F1 Score) is also used to measure the similarity between two images. It is equal to two times the overlap area between prediction and ground truth divided by the total number of pixels in both images.

$$DICE = \frac{2\,TP}{2\,TP + FP + FN} \tag{2.16}$$

3

State-of-the-Art on Microscopy Image Segmentation

Contents

Fluorescence microscopy has recently become an important tool for the study of cells, as it allows the acquisition and visualization of 3D image volumes that extend deeper into the tissue. Therefore, as mentioned in section 1.1, automated 3D microscopy image analysis techniques, specially to perform segmentation, are needed to efficiently and accurately quantify and characterize cells, nuclei, or other biological structures. In this section, we review the current state-of-the-art in microscopic image segmentation (with a particular focus on fluorescence microscopy).

3.1 Microscopic image segmentation challenges

Segmentation of subcellular structures in microscopic images presents many challenges. In particular, during image acquisition, microscopic images often exhibit digital noise, background clutter, and blurring [24]. These problems are exacerbated for microscopic volumes because they are inherently anisotropic and anomalies vary along different axes. As a result, these images often exhibit the following: intensity inhomogeneity, low contrast, limited depth resolution, and consequently the subcellular structures being segmented typically have poorly defined edges [25].

When segmenting nuclei/cells, the challenge becomes even greater because the size, shape, and intracellular intensity heterogeneity of nuclei/cells vary widely and they are often grouped together in clumps so that they sometimes touch and/or overlap [24]. In recent decades, several automated segmentation methods for microscopic images have been investigated that aim to overcome some or all of these challenges.

3.2 Classical approaches

In recent years, several classical image processing algorithms have been investigated. In particular, Active Contour Models (ACMs) have been widely used in microscopy image segmentation (with a particular focus on fluorescence microscopy) due to their ability to segment structures with different shapes. ACMs iteratively minimize an energy/cost function while deforming an initial contour to fit objects of interest. There are several variants of active contours. One is edge-based Active Contour Model (ACM), which uses image gradient maps in object identification [26]. However, these approaches are sensitive to image noise and depend heavily on the placement of the initial contour. Active contours have also been integrated with region-based approaches, which aim to find an energy balance between foreground and background regions [27]. Region-based methods generally achieve better results than edge-based active contours because they are relatively independent of initial contour generation and robust to noise.

In [28], a 3D active surface method is proposed as an extension of the region-based 2D active contour model of Chan-Vese presented in [27] to segment 3D cell structures of a rat kidney. In [25], a method

for segmenting cell nuclei in 3D microscopy volumes based on a combination of 3D region-based active contours and 3D inhomogeneity correction was described. A dataset containing 3D volumes of cell structures from a rat kidney was used. This method achieved an accuracy of 89.6%, outperforming the previous method [27]. Another well-known biomedical imaging tool is Squassh [29, 30], which minimizes an energy function derived from a generalized linear model to segment and quantify 2D or 3D subcellular structures. The main problem with these methods is that they are unable to separate overlapping nuclei, which degrades the quality of the segmentation results.

To separate overlapping cells/nuclei and improve segmentation, several methods have been described. In [31], a model based on coupled active surfaces for cell segmentation is presented. This framework uses multiple active surfaces coupled by a penalty for overlap and a volume conservation constraint that improves the contouring of cell boundaries [32]. Dzyubachyk et al. [31] improved this technique, first presented in [32], by incorporating watershed techniques, a nonlinear partial differential equation (non-PDE)-based energy minimization, and the Radon transform to improve the separation of touching cells. The evaluation of this method was performed in 3D time-lapse fluorescence microscopy image datasets of HeLa cells and achieved an average precision of 98.98%. Moreover, this approach proved to be computationally much more efficient (up to nine times faster) compared to [32]. In [33], Arslan et al. proposed an alternative, a new model-based nuclei segmentation algorithm that defines primitives to represent the boundaries of the nucleus and region growing to delineate the nucleus borders. The model was evaluated on two challenging datasets of fluorescence microscopy images of human hepatocellular carcinoma cell lines and achieved an average precision of 72%. The experiments in this work show that it leads to better results on overlayed nuclei and is less susceptible to noise.

However, these methods require manual optimization of parameters, are difficult to generalize to different datasets, and are unable to discriminate between different cellular structures.

3.3 Convolutional Neural Networks

Recently, deep learning-based models have gained recognition due to their ability to automatically learn important features from data. In the last decades they have been successfully applied to computer vision tasks, including microscopy image analysis, for nuclei detection, cell segmentation, tissue segmentation, image classification, and so on. Many of them were able to outperform the classical approaches mention in section 3.2 [34].

3.3.1 Supervised models

One popular deep architecture is the CNN which, given images and corresponding annotations (ground-truth), is able to learn the essential features of an image that are invariant to irrelevant variations

[13]. CNNs have already been successfully implemented in various computer vision tasks, including microscopy segmentation.

In [35], Xing et al. implemented an automatic nuclei segmentation method using a CNN, for nuclei detection, together with a selection-based sparse shape model. The proposed method was able to achieve superior performance, when compared with other classical models, across three different datasets. In [36], a CNN is used to produce a ternary map, contrary to the binary map proposed in [35]. The additional third class is used to identify pixels on the nuclear boundaries, which helps in the segmentation of crowded and sparse nuclei. Another contribution of this paper include the release of a new dataset, MultiOrgan dataset, with a diversity of nuclear appearances from seven organs and a new evaluation metric to better measure the performance of a nuclear segmentation method called Aggregated Jaccard Index (AJI). The approach obtained reasonable results for different datasets, which shows its generatization hability, and an overall AJI of 0.508 and F1-Score of 0.827, outperforming [35] and other open source softwares [37].

Though these techniques produce good results in 2D images, they're not applicabe for segmentation of 3D microscopic image volumes, because they cannot utilize the depth information in a volume. In [6], Çiçek et al. successfully segmented volumetric microscopic images of the Xenopus kidney with a 3D U-net, by expanding the previous 2D U-net [19] architecture. A more detailed description of the model was presented in subsection 2.1.5. The metric used to evaluate the model was IoU, and the results for the 3D U-Net yielded an average value of 0.704 compared to the 2D U-Net which yielded an average IoU value of 0.547. However, the results of the experiments also show that the performance of the network is highly dependent on the amount of annotated data available, as it performed better in semi-automatic segmentation than in fully automatic segmentation.

Supervised deep neural network methods, like the ones mentioned above, require a large amount of pixel-wise annotated data for training. Obtaining these detailed annotations is not only time consuming, especially for 3D volumes, but also must be done by an expert. Therefore, the interest in weakly/fully unsupervised deep learning models that perform well on data without annotation has increased significantly in recent years.

3.3.2 Weakly Supervised / Unsupervised models

In [38] and [39], weakly supervised methods were successfully used to segment cell nuclei in 2D and 3D microscopy images, respectively. Qu et al. [38] proposed the use of points annotation for nuclei segmentation. From these annotations, two types of coarse labels are derived using the Voronoi diagram and the k-means clustering algorithm. These labels are then used to train a CNN with cross-entropy loss. The authors also implement a dense CRF loss to refine the trained model. The performance of the model was evaluated using the MultiOrgan dataset of [36], for which an accuracy of 0.907 and an AJI

value of 0.510 were obtained. It has been shown that the performance of the weakly supervised method is close to the fully supervised models with the same network structure and, moreover, the annotation time spent on each image is greatly reduced. However, from the AJI value, we can also conclude that the nuclear shapes and separation can still be improved.

In [39], a model for segmenting 3D instances is proposed that uses weak annotations for training. Detection of all instances of interest is achieved using 3D bounding boxes, and segmentation is achieved for all detected instances by using the full voxel annotation only for a small subset of the instances. For segmentation, the authors use a Fully Convolutional Network backbone with VoxRes block [40]. The performance of the detection model is compared with the supervised method VoxResNet [40] and it was found to achieve similar performance with less annotation time. However, the annotation time is still very long (at least 5.5 hours).

Although these methods help to minimize the amount of work required to label images, approaches have been presented in the literature that allow for completely unsupervised segmentation of medical images. Ho et al. [41] proposed an unsupervised approach to segment cell nuclei from 3D fluorescence microscopy images. For this purpose, a 3D CNN with an encoder-decoder struture was constructed and trained with generated synthetic microscopy volumes and synthetic ground truth volumes containing multiple cell nuclei. The performance of the model was compared with the previously discussed active surface models [28, 25], Squassh [30], and a 2D CNN model [42], and it achieved better results than the first two and similar results to the 2D CNN, with an accuracy of 92.93%. However, the false detection rate is higher than the best results, i.e., the model tends to over-segment. In [43] the authors present a method, based on [41], for detection and segmentation of cell nuclei in 3D fluorescence microscopy images. In this approach, each nuclear center is detected using adaptive 3D histogram equalization, 3D distance transformation, and 3D classification CNN. Then, the nuclei surrounding the seeds are segmented using a 3D segmentation CNN. This approach has been shown to perform better than [41], but is unable to detect all the nuclei, resulting in under-segmentation (high false negative rate).

3.4 Generative Adversarial Networks

The performance of CNN architectures is always limited by the quantity and quality of the dataset used for training. This is where the GAN model comes into play to significantly improve the performance of these approaches due to their superior ability to generate data as well as translate it, as mentioned in section 2.2.

3.4.1 Supervised models

In [44], a robust transfer learning framework for the segmentation of HEp-2 Specimen image segmentation using GAN is presented. A novel conditional generative adversarial network with classifier (cC-GAN) is introduced to solve the overfitting problem of most DL models by improving their transfer capacity. Using this method, a segmentation accuracy of 75.27% was achieved in the MIVIA dataset, showing the great potential of applying GAN models to microscopic segmentation problems.

3.4.2 Weakly Supervised / Unsupervised models

In [45], a weakly supervised approach to nucleus segmentation is proposed. A point labelling is used as a weak labelling, then the author uses a Pix2Pix network model to detect the centroid of the nucleus and build a likelihood map, a guided propagation to build a pixel contribution map of the nucleus, and a graph cut to obtain the final instance segmentation. The proposed approach is able to outperform the fully supervised U-Net [19] model.

In a multi-organ study for nuclei segmentation on histopathology images [46], the authors used CycleGAN and cGAN models. First, a CycleGAN model was trained with images from four different organs to synthetically generate pathology data along with perfectly segmented nuclei labels. The real and synthetically generated images were then used to train a cGAN to perform nuclei segmentation. This segmentation network showed a 29.19% improvement in AJI compared to the previously mentioned supervised model CNN-3C [36] and of 73.19% compared to the U-Net model [19].

The work proposed in [41] was improved in [47] by integrating GAN. A challenging problem in previous work has been the difficulty in generating realistic synthetic microscopic data to train the CNN, due to the natural diversity of biological structures. Fu et al. [47] addressed this problem with a modified model of CycleGAN, a spatially constrained CycleGAN, SpCycleGAN. The spatially constrained CycleGAN is used to generate 3D realistic synthetic training data. Then, a modified 3D U-Net network is trained with these 3D synthetic data to segment nuclei structures. The final accuracy result was 94.59%, outperforming [41]. However, because this model has some difficulty separating cell nuclei, an extension of this approach for nuclei detection and segmentation was presented in [48], using the synthetically generated images from SpCycleGAN to train a CNN architecture for classification and segmentation.

DeepSynth [49] is currently recognized as the state-of-the-art deep model for unsupervised 3D nuclei segmentation, and is based on the previously referenced work [42, 41, 47].

More recently, in 2021, [50] treats the segmentation of cell nuclei as an image-to-image translation problem. Yao et al. [50] propose a novel end-to-end unsupervised framework called Aligned Disentangling Generative Adversarial Network (AD-GAN). They performed segmentation of cell nuclei in challenging 2D and 3D datasets of microscopic images. This work addresses the problem of lossy

transformation [51], which is defined by the content inconsistency between the original images and the corresponding segmentation masks obtained by the model. These inconsistencies include: the deletion/addition of nuclei at the macro level, shape differences at the micro level, and a location offset. With this new and unsupervised model of GAN, the authors were able to mitigate the problem of lossy transformations at the macro and micro levels and even extend this work to instance segmentation. The results obtained for the 3D fluorescence image dataset Scaffold [52] were a DC of 89%. This is significantly higher than the compared models, the DeepSynth model [49] (45.8%) and other well-known biomedical imaging tools such as Cell-Profiler 3.0 [37] (53.1%) and Squassh (62.3%) [30].

In summary, several approaches have been proposed for the segmentation of subcellular structures in microscopic images, and some of the best results have been obtained with deep learning-based models. These models can be trained in a supervised or weakly/unsupervised manner. The advantage of unsupervised models is their independence from annotated data for training. State-of-the-art unsupervised approach presented in [49] used the CycleGAN as a two-stage pipeline to train a robust segmentor with CycleGAN-synthesized data. However, this adds time and computational costs and significantly increases the complexity of the system. In this work, it will be implemented a different approach, more similar to the AD-GAN model, train an unsupervised end-to-end CycleGAN model to segment subcellular strutures, nuclei and Golgi, in fluorescence microscopy images.

4

Methodology

Contents

As mentioned earlier, in this work DL methods are used to segment fluorescent microscopic images. The first phase is to implement supervised models to accomplish this task of segmenting the nuclei and Golgi of these images. Then we focus on the main goal, which is to implement an unsupervised model that can perform the same task as well or better than the supervised models. And finally, we add a third class to detect the nucleus-Golgi pairs.

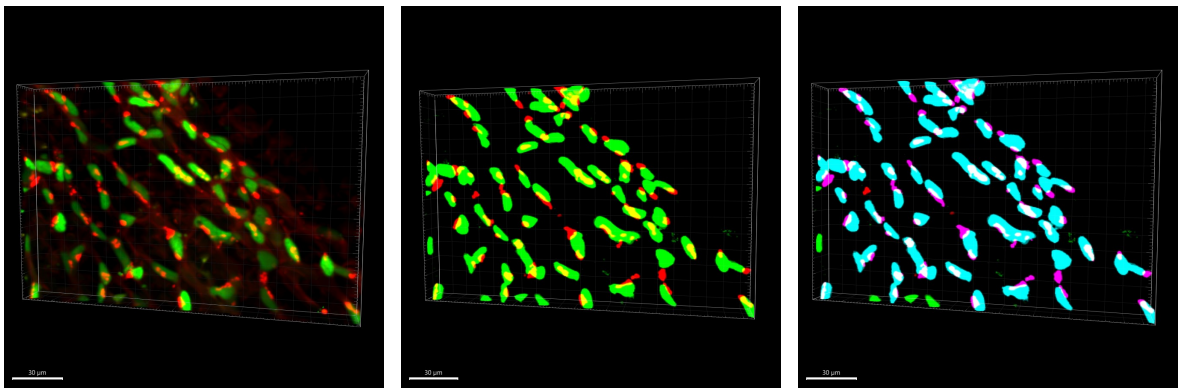
This chapter begins by presenting the dataset used to train, validate, and test these models. The proposed approach to semantic segmentation using an unsupervised model is described in section 4.2. The performance of this approach is compared to the supervised models, which are described in detail in section 4.3. Then, section 4.4 describes how the models are used for inference after training. Section 4.5 describes the metrics used to measure the performance of these models. Finally, section 4.6 describes how the execution time for the trained and tested models is determined to use as a measure of the performance of the different approaches.

4.1 Dataset

The dataset used in this work consists of eight crops extracted from 3D fluorescence microscopy images of mouse retinas. These crops range in size from 257x505x55 to 627x818x61. In these crops, the nuclei are labeled with green fluorescent protein (GFP) and the Golgi are labeled with mCherry. In addition, manually labeled segmentation masks are used for training the proposed supervised methods and evaluating their performance. These masks are RGB, with the red channel containing the segmentation mask of the Golgi and the green channel containing the segmentation mask of the nuclei. The blue channel contains the segmentation mask of the nucleus-Golgi pairs, but only for the task of the 3-class segmentation problem. In this case, the input images and the output masks have 3 channels. For the 2-class segmentation problem, the blue channel is not considered, so the input images and the output masks have only two channels (these masks are converted to RGB for visualization but the blue channel is all zeros).

An example of a 3D crop and its corresponding manually labelled segmentation masks for the 2-class task and 3-class task can be found in figure 4.1.

To use these crops to train the models, its necessary to divide them into equal sized patches. If the volumes are too small, not enough information can be learned. If the volumes are too large, training will take longer and could exceed GPU memory. Therefore, we set the size of these patches to 64x64x64. Since the z-axis of the original images has a size between 55 and 61, the crops had to be padded to get the 64 slices in the z-axis for the input images, for this purpose a reflection padding was applied.



(a) (b) (c)

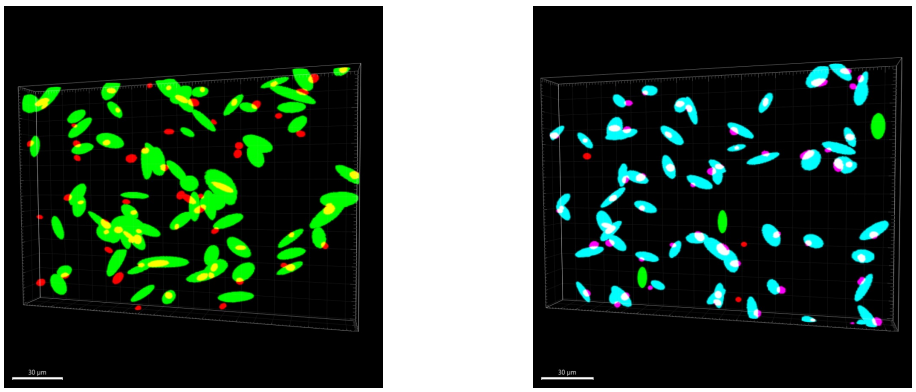
Figure 4.1: (a) Example of a 3D crop of a microscopy image of mouse retina; (b) ground truth mask for 2-class task; (c) ground truth mask for 3-class task.

4.1.1 Synthetic segmentation masks

As mentioned earlier, the manually labelled segmentation masks are used for training the implemented supervised models. However, in order to train the proposed model in a non-supervised way, synthetic segmentation masks had to be created.

For this purpose, ellipsoids are created to represent the nuclei and spheres for the Golgi at random positions. For each nucleus-Golgi pair created, the radius of the speres (nucleus) and the size of the principal and secondary axis of the ellipsoids (Golgi), the distance between the nucleus and Golgi (relative to the center of the Golgi), and the rotation of each pair are selected randomly from the intervals [5,9] pixels, [[17,30],[8,13],[8,13]] pixels, [-6,6] pixels and [0,180] degrees, respectively. For each crop created, a random number of nucleus-Golgi pairs were selected (between 45 and 70 pairs). To make the images more realistic, an elastic transformation was applied to each nucleus-Golgi pair. Six crops were created with the same size of the six microscopic image crops used for training the model. An example of a synthetically created segmentation mask can be found in figure 4.2 (a).

For the 3-class segmentation problem, another six synthetic segmentation masks were created to train the models for this task. Here, the nuclei and Golgi are generated in the same way as described previously. In this case, a third segmentation mask is added in the blue channel corresponding to the class of the nucleus-Golgi pair that corresponds to the union of the nuclei and Golgi generated for the green and red channels, respectively. To make the images more realistic, isolated nuclei and Golgi are added to the segmentation mask and are not included in the segmentation mask of the nucleus-Golgi pairs (blue channel). An example of a synthetically generated segmentation mask can be found in figure 4.2 (b).



(a) (b)

Figure 4.2: Example of a synthetic 3D segmentation masks (a) 2-class segmentation task; (b)3-class segmentation task.

4.2 Proposed Approach

For this work, cell nuclei and Golgi semantic segmentation will be performed using the cycle-consistent GAN (CycleGAN) model. Two different domain images will be considered, the fluorescence microscopy images (domain I) and the synthetic segmentation mask (domain S). Moreover, let i and s denote training examples where $i \in I$ and $s \in S$.

Like the original CycleGAN model [8], the architecture of this segmentation model will be composed of four interconnected networks, two generators and two discriminators. A representation of this network can be found in Figure 4.3. The first generator (G_S), corresponding to the segmentation network we wish to obtain, learns a mapping from the domain I to S. The first discriminator (D_S) takes the segmentation masks as input and tries to predict whether they are real or generated, according to the domain S. In contrast, the second generator (G_I) learns a mapping from the domain S to I, and is used only to improve the training. Furthermore, the second discriminator (D_I) receives an image as input and predicts whether this image is real or generated, according to domain I.

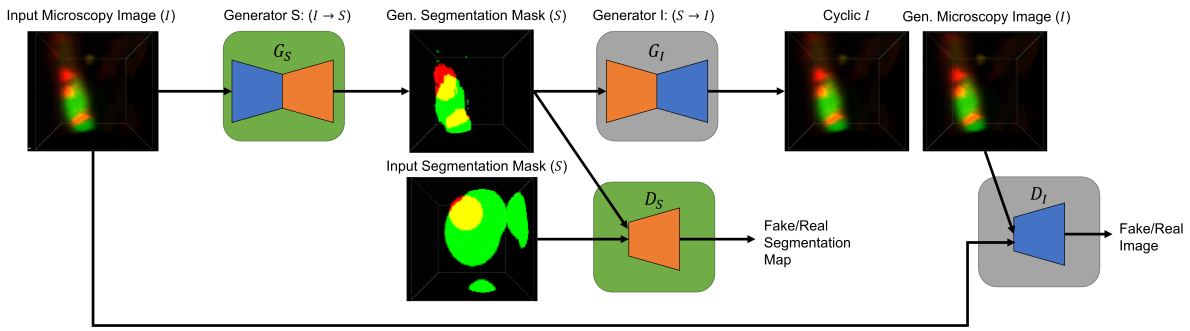


Figure 4.3: CycleGAN schematic for the proposed approach.

Network Architecture

The generator architecture used in the original CycleGAN model is intended for a 2D image-to-image translation task. Therefore, for our dataset, this approach needs to be adapted to perform a 3D image-to-image translation. This can be achieved by replacing the 2D convolutional layers of the original CycleGAN with a 3D convolutional layers. The discriminator architecture to be implemented is based on the PatchGAN [7], which includes Leaky ReLU for nonlinearity. The layers of the discriminator architecture must also be converted from 2D to 3D. The detailed architecture of the generator and discriminator used in the CycleGAN model employed in this work are shown in Figures 4.4 and 4.5, respectively.

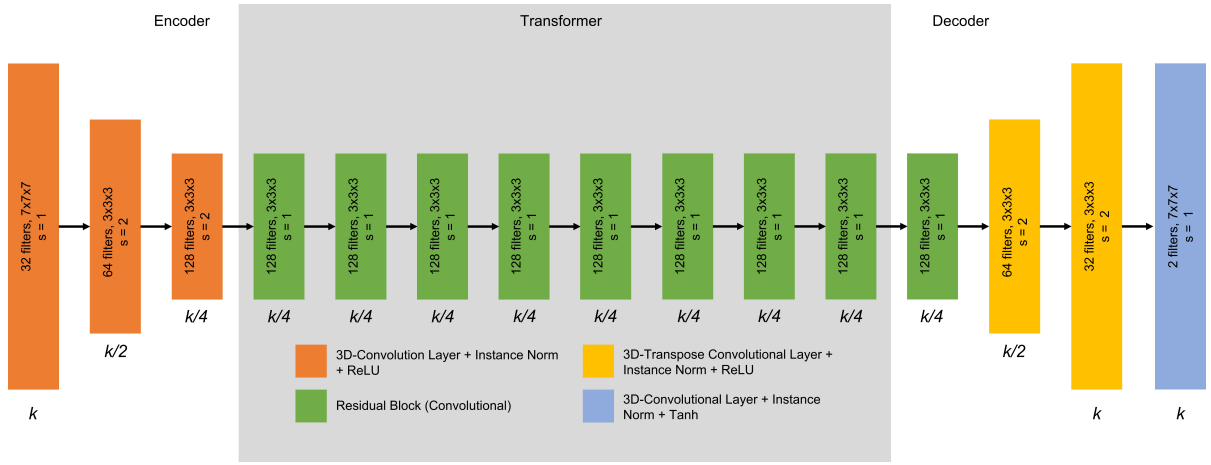


Figure 4.4: Implemented CycleGAN generator architecture. The Residual Block consist of a 3D convolutional layer with instance normalization and ReLU as the activation function, followed by a second convolutional layer also with instance normalization. The output of this block is the concatenation of the output of the second layer with the input layer of the block.

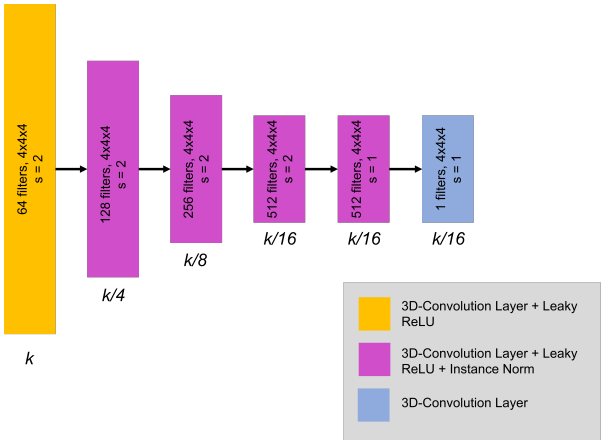


Figure 4.5: Implemented CycleGAN PatchGAN discriminator architecture.

In the figures 4.4 and 4.5 above, each box corresponds to a block in the model. For each block, the type of convolutional layer and the normalization and activation function used after each layer are

described in the figure legend. The size of the feature map that each layer outputs is provided at the bottom of the boxes with respect to the size of the input image k. In each box, the number of filters, the size of these filters, and the stride used in each convolutional layer are indicated.

Loss functions

The loss function used to train CycleGAN comprises four terms. These terms are described below.

Adversarial Loss. The adversarial term encourages the mapping functions to translate from one domain to the other. First, the negative log likelihood objective from $\mathcal{L}_{GAN}$ (Equation 2.2) is replaced by a least-squares loss, since according to [8], this loss is more stable during training and produces higher quality results. Therefore, the adversarial loss for the first GAN (G_S, D_S), for the already mentioned notation, is given by,

$$\mathcal{L}_{adv}(G_S, D_S) = \mathbb{E}_{s\sim p_{data}(s)}[(D_S(s) - 1)^2] + \mathbb{E}_{i\sim p_{data}(i)}[(D_S(G_S(i)))^2]. \tag{4.1}$$

The adversarial loss for the second GAN (G_I, D_I), is written as,

$$\mathcal{L}_{adv}(G_I, D_I) = \mathbb{E}_{i\sim p_{data}(i)}[(D_I(i) - 1)^2] + \mathbb{E}_{s\sim p_{data}(s)}[(D_I(G_I(s)))^2]. \tag{4.2}$$

Cycle consistency loss. The cycle consistency loss, $\mathcal{L}_{cyc}$, was described in subsection 2.2.3 with an expression given by (2.10). Transferred to our notation, it has the following form,

$$\mathcal{L}_{cyc} = \mathbb{E}_{i\sim p_{data}(i)}[||G_I(G_S(i)) - i||_1] + \mathbb{E}_{s\sim p_{data}(s)}[||G_S(G_I(s)) - s||_1]. \tag{4.3}$$

Identity mapping loss. Lastly, the identity mapping loss, $\mathcal{L}_{id}$ has also been described in subsection 2.2.3 with an expression given by (2.11). Transferred to our notation, it has the following form,

$$\mathcal{L}_{id} = \mathbb{E}_{i\sim p_{data}(i)}[||G_I(i) - i||_1] + \mathbb{E}_{s\sim p_{data}(s)}[||G_S(s) - s||_1]. \tag{4.4}$$

Finally, the total loss is obtained by combining by combining the 4 terms:

$$\mathcal{L}(G_S, G_I, D_S, D_I) = \mathcal{L}_{adv}(G_S, D_S) + \mathcal{L}_{adv}(G_I, D_I) + \lambda_{cyc}\mathcal{L}_{cyc} + \lambda_{id}\mathcal{L}_{id} \tag{4.5}$$

where λ_{cyc} and λ_{id} are hyperparameters to optimize that control the relative importance of the two objectives. Thus, the objective function is:

$$\arg \min_{G_S,G_I} \max_{D_S,D_I} \mathcal{L}(G_S, G_I, D_S, D_I). \tag{4.6}$$

4.3 Comparison between different approaches

The performance of the proposed approach will be compared with two other known supervised techniques for segmenting nuclei and Golgi in fluorescence microscopy images, the 3D U-Net [6] and Vox2Vox, which is the 3D extension of the Pix2Pix model [7]. To train these models, the 3D volumes of fluorescence microscopy images and corresponding 3D segmentation masks are required.

The following subsections provide information on the implementation of these methods.

4.3.1 3D U-Net

The 3D U-Net architecture is used to predict two binary segmentation maps, one for the nuclei and other for the Golgi.

The implemented architecture, based on the one described in section 2.1 (figure 2.6), is demonstrated in figure 4.6.

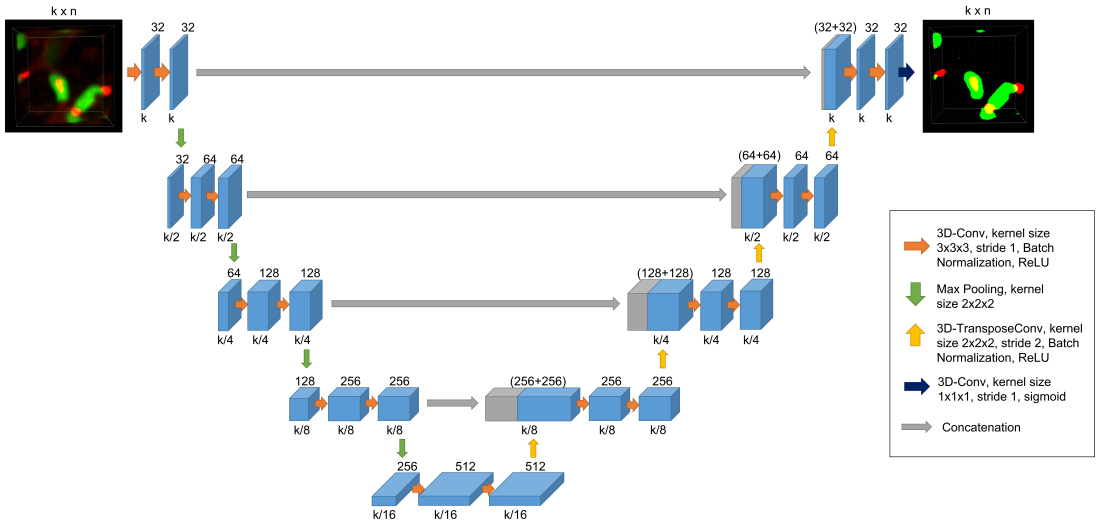


Figure 4.6: Implemented 3D U-Net architecture.

In the figure above, each blue box corresponds to a multi-channel feature map. The number of channels is indicated at the top of the box. The size of the feature map is provided at the bottom of the box with respect to the size of the input image k. Grey boxes represent copied feature maps. The arrows denote the different operations. The n (above the input and output images) is the number of channels of the input and output images, n=2 and n=3 for the 2-class and 3-class segmentation tasks, respectively. The output image in the figure is an example of the output for the 2-class segmentation task.

The DC is a widely used metric in the computer vision community to calculate the similarity between two images [53]. Therefore, the dice coefficient has been fitted to a loss function known as Dice Loss (DLoss):

$$DLoss(y, \hat{p}) = 1 - \frac{2y\hat{p} + 1}{y + \hat{p} + 1} \tag{4.7}$$

here y is the reference value, $\hat{p}$ is the probability value predicted by the model and 1 is added to both the numerator and denominator to ensure that the function is not undefined in edge case scenarios, e.g., when $y = \hat{p} = 0$. Unlike the original 3D U-Net model [6] previously described in section 2.1, the loss function used in training the model is the dice loss represented in equation 4.7.

4.3.2 Vox2Vox

Pix2Pix is a supervised GAN model designed for image-to-image translation, previously described in subsection 2.2.2. However, it is not capable of performing image-to-image translation at the 3D level. A 3D volume-to-volume network for segmentation used as an alternative to Pix2Pix is the Vox2Vox network.

Cirillo et al. [54] used a Vox2Vox approach in their work. For this work, it will be used a similar architecture. The generator model is built in the style of the U-Net and Res-Net [55] architectures, while the discriminator is built in the style of the PatchGAN [7] architecture. The implemented architectures are shown in figures 4.7 and 4.8.

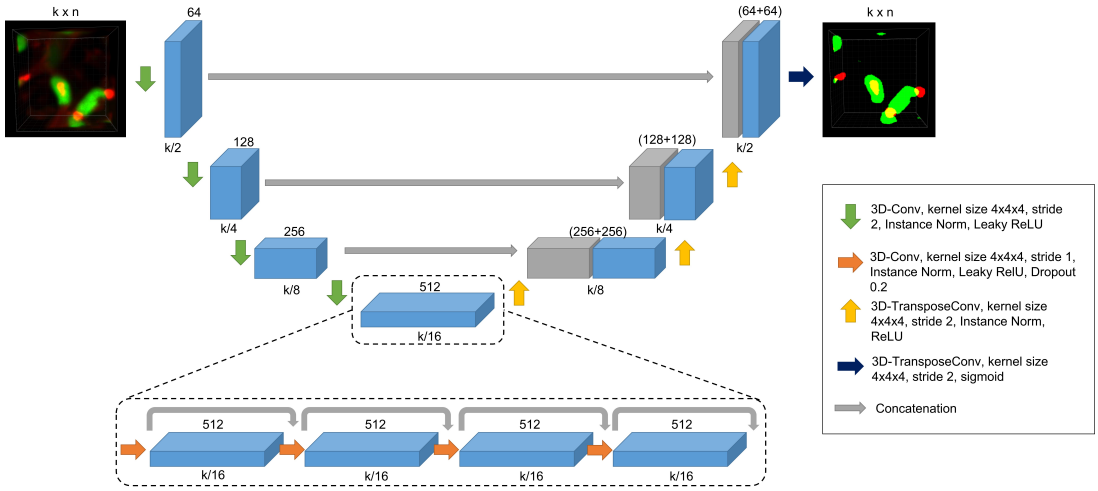


Figure 4.7: Implemented Vox2Vox generator architecture.

In the figures 4.7 and 4.8, each blue box corresponds to a multi-channel feature map. The number of channels is indicated at the top of the box. The size of the feature map is provided at the bottom of the box with respect to the size of the input image k. Grey boxes represent copied feature maps. The arrows denote the different operations. The n (above the input and output images in the generator architecture) is the number of channels of the input and output images. The output of the generator architecture is an example of the output for the 2-class segmentation task.

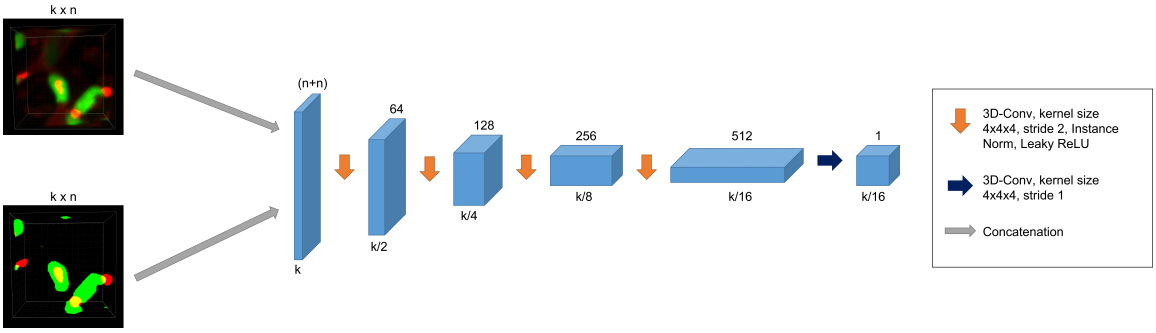


Figure 4.8: Implemented Vox2Vox discriminator architecture.

Another important part of the implementation of Vox2Vox model is the loss function. The discriminator loss, $\mathcal{L}_{disc}$, is the sum of the L_2 error of the discriminator output between the original image x and the respective ground-truth y with a tensor of ones, and the L_2 error of the discriminator output between the original image and the respective segmentation prediction $\hat{y}$ given by the generator with a tensor of zeros. This can be formulated as follows:

$$\mathcal{L}_{disc} = L_2[D(x, y), \mathbf{1}] + L_2[D(x, \hat{y}), \mathbf{0}]. \tag{4.8}$$

On the other hand, the generator loss, $\mathcal{L}_{gen}$, is the sum of the L_2 error of the discriminator output between the original image, x, and the corresponding segmentation prediction, $\hat{y}$, with a tensor of ones, and the DLoss, between the ground-truth and the generator output multiplied by the scalar weight coefficient $\alpha > 0$. This can be formulated in the following way:

$$\mathcal{L}_{gen} = L_2[D(x, \hat{y}), \mathbf{1}] + \alpha DLoss(y, \hat{y}). \tag{4.9}$$

4.4 Models Inference

As indicated in 4.1, the input size of the proposed models is 64x64x64. Therefore, to obtain the predicted segmentation mask of an entire crop, it is necessary to slide a 3D window of size 64x64x64 to segment the entire crop. To make the segmentation mask the same size as the original crop, we need to pad the X and Y axes of the original image so that the dimensions are $X_{pad} = (X\%step) \times step$ and $Y_{pad} = (Y\%step) \times step$, where $step$ is the number of pixels the 3D window moves during inference. The padding used is reflection padding. If you place a 3D window in the upper left corner of the padded crop a 64x64x64 patch becomes the input volume of the model used to create a sub-volume of the segmentation mask in the upper left. This 3D window is slid in x and y direction with a certain $step$ through the entire padded crop until the entire volume is processed.

4.5 Segmentation performance metrics

Pixel-based metrics are used to evaluate the performance of the different models. Specifically, precision, recall, and Dice Coefficient are used to evaluate these different approaches. The equations for these metrics are as follows:

$$Precision = \frac{n_{TP}}{n_{TP} + n_{FP}}, \tag{4.10}$$

$$Recall = \frac{n_{TP}}{n_{TP} + n_{FN}}, \tag{4.11}$$

$$DICE = \frac{2 \times n_{TP}}{(n_{TP} + n_{FP}) + (n_{TP} + n_{FN})} \tag{4.12}$$

where n_{TP} , n_{FP} and n_{FN} are defined as the number of true-positive, false-positive, and false-negative segmentation result pixels in an image, respectively. These metrics are calculated separately for the objects to be segmented: nuclei, Golgi and nucleus-Golgi pair. A higher DC indicates better intersection between the ground truth and the predicted segmentation masks. Low precision may indicate over segmentation and low recall may indicate under segmentation.

4.6 Execution time

The execution time of a deep neural network model can be divided into two intervals: the training time and the testing time. Here, the training time corresponds to the time it takes for a model to learn a particular task. The test time is the time required for a model to make a segmentation prediction for an image. In this work, the execution time is also considered as a measure of the performance of the different approaches. For supervised methods that require manual annotations, the time required to create these masks is also considered.

5

Experimental Results and Discussion

Contents

This chapter presents the experimental results and discussion regarding the segmentation of cell nuclei and Golgi in fluorescence microscopy images.

5.1 Computational Environment

In this work, all deep learning implementations are based on the open-source deep learning libraries Tensorflow and Keras [56, 57]. Tensorflow is an end-to-end machine learning platform that makes it easy to build and train machine learning models. Keras is a high-level application programming interface (API) of Tensorflow. In addition, all experiments were performed in Python 3.9 on a computer with the following specifications:

- Processor: Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz;
- Memory: 16 GB RAM;
- Storage: 250 GB SSD + 1 TB HDD;
- Graphics Processing Unit (GPU): NVIDIA GeForce GTX 1070 with 8 GB of VRAM.

5.2 Data Pre-Processing

This section presents the pre-processing methods applied to the images in the dataset presented in section 4.1.

From the histogram of intensity values in figure 5.1 of one of the microscopic images from the dataset, it's possible to verify that most of the pixels in the red and green channels have low intensity values. To correct this, we need to manipulate the intensity histogram of the image, which can be done by contrast stretching.

Contrast stretching should theoretically improve the learning ability of models by highlighting the contours of objects and emphasizing the difference between object and background. This helps convolutional layers extract information and features from images [58]. Contrast stretching can be easily done for an image using a formula that scales up the differences in pixel values. This up scaling must be applied to all color channels, so for the microscopic images used in this work, it must be applied to the red and green channels (as mentioned earlier, these images have no blue channel).

The contrast stretching can be done by transforming pixel x following formula:

$$f(x) = \frac{x - a}{b - a} \times 255 \tag{5.1}$$

where a is the lower limit to scale to, and b is the upper limit. Instead of using the minimum and maximum values in the contrast stretch as the lower and upper bounds, the 5 and 98 percentile values are used,

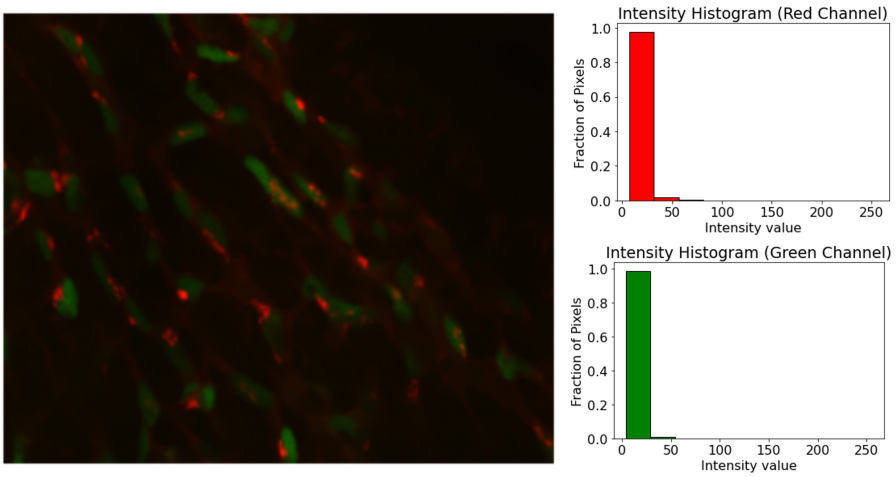


Figure 5.1: Slice of microscopic image from the dataset.

respectively. These values were chosen by visual inspection of the pre-processing results considering different percentile values. The values that best enhance the contrast of the Golgi and nuclei were selected.

The results of this process can be found in figure 5.2 and, together with the corresponding new intensity histograms for the red and green channels.

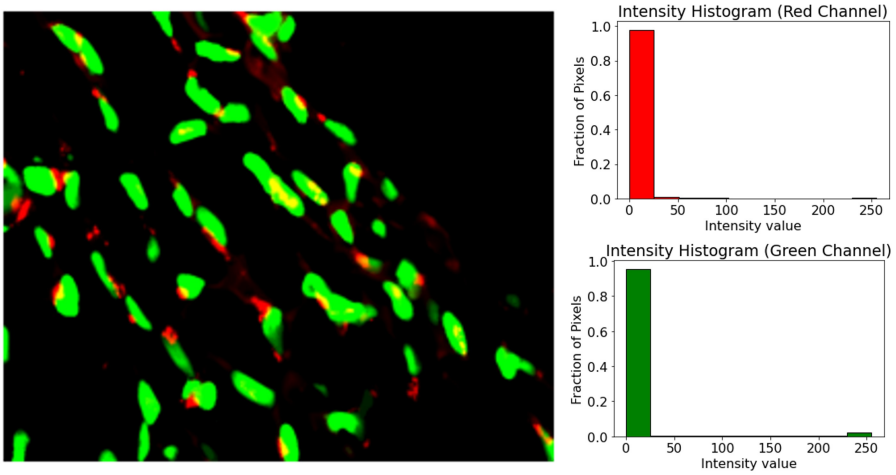


Figure 5.2: Same image slice of figure 5.1 after percentile contrast stretching.

Another simple step of the data pre-processing method applied to the dataset of microscopic images was to normalize the input images with values in the range [0,255] to values in the interval [0,1] by dividing each image value by 255. This process is important because when the original images are used as input of the Deep Neural Networks, computing these high numerical values requires a lot of computational resources and time. When the data is normalized, the pixel values are smaller, so the computational resources and time required to converge the model are greatly reduced.

5.3 Implementation Details, Results and Discussion

In this section the experimental results regarding the segmentation of cell nuclei and Golgi in fluorescence microscopy images obtained with the proposed models are presented and discussed. In each subsection, a different model is described and each subsection is divided into two parts. In **2 Class**, the results obtained for the models predicting two output probability maps (for nuclei and Golgi) are presented, and in **3 Class**, the results obtained for the models predicting three output probability maps (nuclei, Golgi and nucleus-Golgi pairs) are presented.

All models are tested on two of the eight crops in the dataset described in section 4.1. Shown in figure 5.3 are: the original test crops in (a) and (e), designated I_1^{orig} and I_2^{orig}, respectively; the crops obtained by applying the pre-processing method described in the previous section in (b) and (f), designated I_1^{Pre} and I_2^{Pre}, respectively; the ground truth segmentation masks for the models with 2 output channels in (c) and (g), denoted I_{21}^{gt} and I_{22}^{gt}, respectively; and the ground truth segmentation masks for the models with 3 output channels in (d) and (h), denoted I_{31}^{gt} and I_{32}^{gt}, respectively.

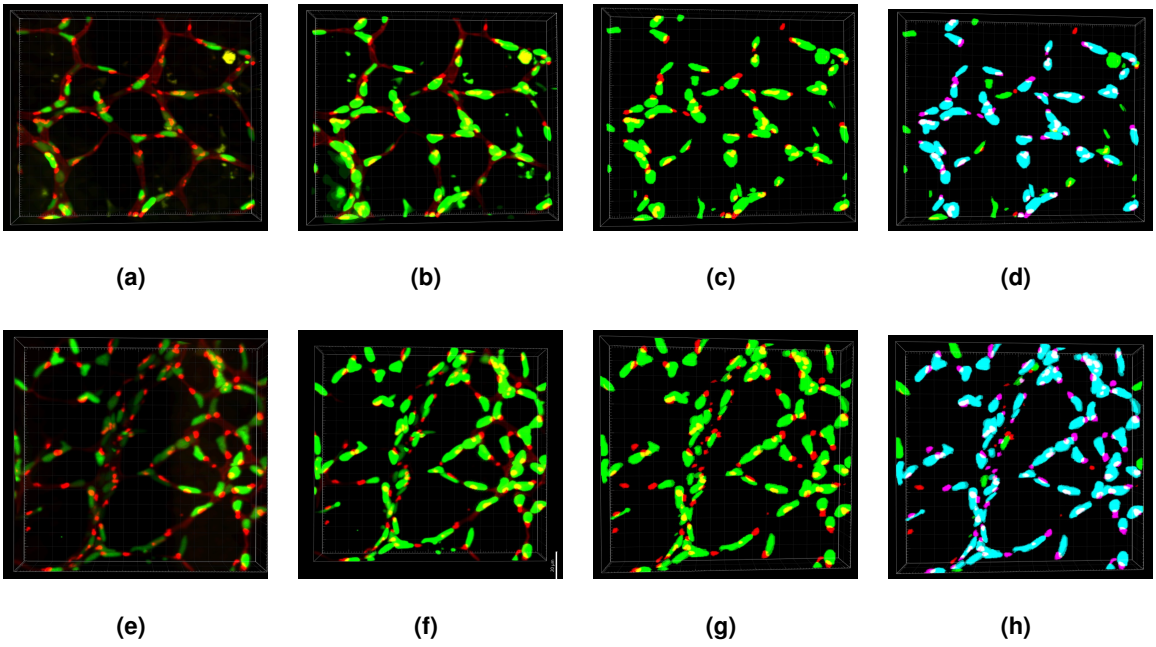


Figure 5.3: 3D visualization of crops used for testing the models: (a) I_1^{orig}; (b) I_1^{Pre}; (c) I_{21}^{gt}; (d) I_{31}^{gt}; (e) I_2^{orig}; (f) I_2^{Pre}; (g) I_{22}^{gt}; (h) I_{32}^{gt}.

As mentioned in section 4.1, the crops for training the proposed models were divided into equal-sized patches. Therefore, the training dataset for all proposed models consists of 433 patches with a size of 64x64x64 from 6 different crops.

5.3.1 3D U-Net

In this subsection the results obtained by training and testing the 3D U-Net model are presented and discussed.

Implementation Details

First, details on the choice of the model's hyperparameters are presented.

In all experiments, using the 3D U-Net model: Early stopping was applied to stop training when the validation loss did not improve for 10 epochs; the model was trained with the ADAM optimizer with a learning rate of 1e-3; using Xavier initialization for the model weights; a batch size of 4 and ReLU as activation functions, except in the last layer, which has a sigmoid activation function so that the output is n-probability maps (where n is the number of classes). A threshold of 0.5 is then applied to each probability map of the output of the U-Net to obtain the final segmentation mask.

2 Class

The U-Net model was first trained with the original dataset (figure 5.1). The obtained segmentation masks for the experiment are shown in figures 5.4 (b) and 5.4 (e), and table 5.1 lists the values obtained for the metrics used to evaluate the model which were described in section 4.5.

From the results, it can be concluded that in the original dataset the precision of the nuclei and especially the Golgi is very low, which indicates over-segmentation and can be verified by looking at the obtained images. This also leads to a very low dice coefficient. Therefore, to improve these results, the pre-processing method described in the previous section 5.2 was applied to the microscopic images and the U-Net model was re-trained with patches extracted from the pre-processed images.

The segmentation masks obtained for the pre-processed microscopic images are shown in figures 5.4 (c) and 5.4 (f) and table 5.1 provides the values of the metrics used to evaluate the model. Compared to the previous results, the pre-processing method improved the results significantly, especially the precision and Dice Coefficient. However, it still tends to over-segment, especially the Golgi.

Table 5.1: Average metric values after testing the 3D U-Net model on two microscopic images for a model trained with the original microscopic images (U-Net) and with the pre-processed microscopic images (U-Net + Pre)

	DC Nuclei	DC Golgi	Precision Nuclei	Precision Golgi	Recall Nuclei	Recall Golgi
U-Net	0,4497	0,3054	0,2912	0,1842	0,9895	0,9186
U-Net + Pre	0,7775	0,6938	0,7067	0,6500	0,8692	0,7683

There are two things that have been identified in the dataset that may lead to decreased precision of the model. The microscopic images have digital noise, which causes the model to classify this noise

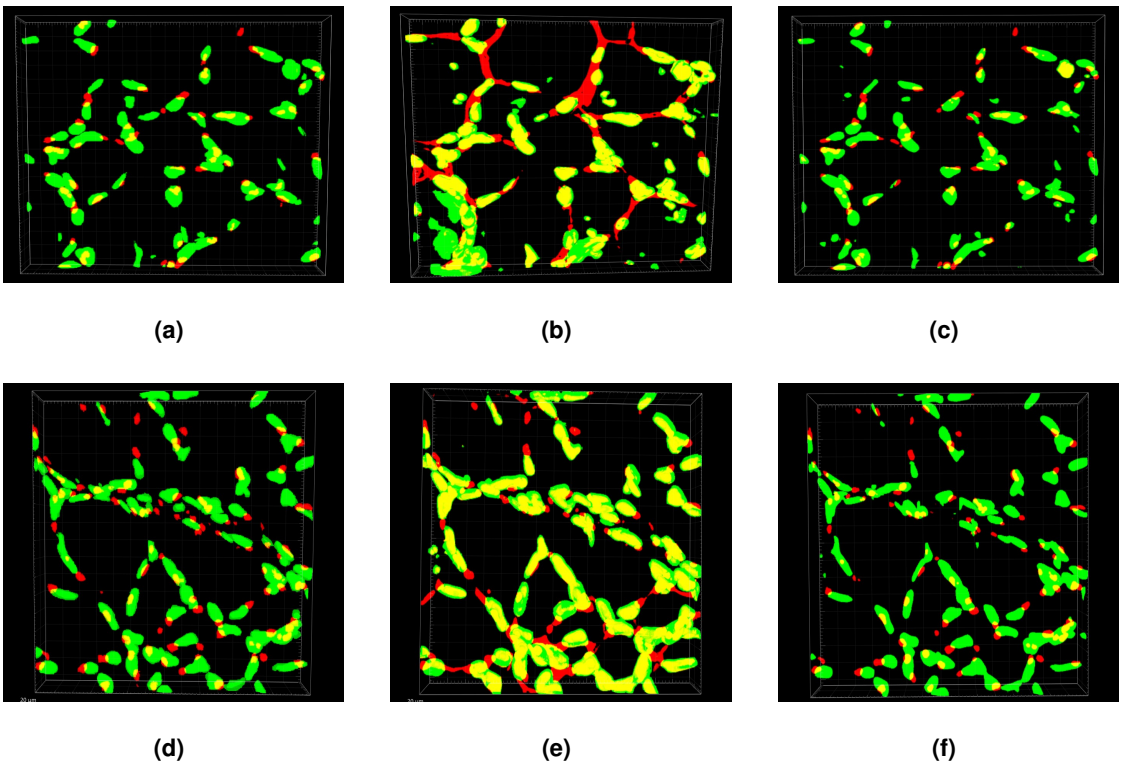


Figure 5.4: 3D visualization of segmentation masks: (a) 3D ground truth I_{21}^{gt}; (b) 3D U-Net model tested on I_1^{orig}; (c) 3D U-Net model tested on I_1^{Pre}; (d) 3D ground truth I_{22}^{gt}; (e) 3D U-Net model tested on I_2^{orig}; (f) 3D U-Net model tested on I_2^{Pre}.

as Golgi/Nuclei. This has been verified specifically for the Golgi, and an example of this noise can be seen in figures 5.5 (d) to (f) highlighted by the blue circles. Another reason that leads to lower precision is that the manual segmentation masks contain annotation/segmentation errors. In figures 5.5 (a) to (c), we see that a Golgi is not identified in the segmentation mask of our dataset, even though it is present in the microscopic image along with the nucleus and is identified correctly by the U-Net model.

The low recall can be explained by the fact that the microscopic images have low contrast in certain areas, even after applying the pre-processing method to the images. This causes the model to fail to recognise some Golgi/nuclei, which lowers the recall. This case is illustrated in figure 5.5 (d) to (e) by the orange circle.

To improve the results, different loss functions were tried to better segment the nuclei and Golgi, which would account for the imbalance between the background and the nuclei/Golgi, for example Focal Loss [59], but this did not lead to better results, instead caused the model to detect more noise. Another attempt was made by increasing the size of the data by applying transformations such as rotations and flips (reflections) to patches of the pre-processed dataset, but again this also did not produce better results but resulted in the model detecting more noise.

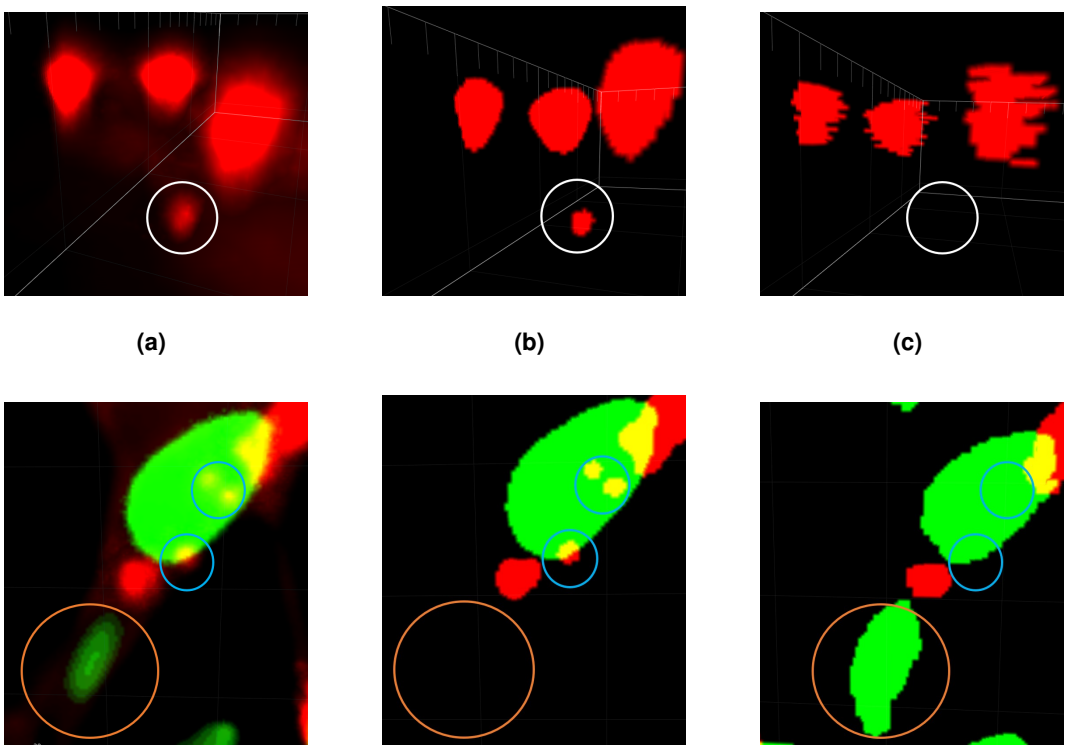


Figure 5.5: Example of ground truth mask imperfection highlighted by white circle; Example of digital noise highlighted by blue circles; and example of nondetection by 3D U-Net model highlighted by orange circle; (a) and (d) 3D microscopic section; (b) and (e) 3D U-Net segmentation mask; (c) and (f) 3D ground truth segmentation mask.

3 Class

In this approach, the number of output probability maps of the 3D U-Net is three, so we can classify the nuclei, Golgi and nucleus-Golgi pairs of the input images. All other parameters described in the implementation details are kept. The model is trained with the pre-processed dataset and respective ground truth masks.

After training, the performance of the model is tested with the two microscopic image volumes I_1^{Pre} and I_2^{Pre}. Figure 5.6 is a 3D visualization of the segmentation results for these volumes and table 5.2 shows the average quantitative results.

Table 5.2: Average metric values after testing the 3 class 3D U-Net model on two pre-processed microscopic images

DC Nuclei	DC Golgi	DC Pair	Precision Nuclei	Precision Golgi	Precision Pair	Recall Nuclei	Recall Golgi	Recall Pair
0,7916	0,6752	0,7601	0,7589	0,5935	0,6941	0,8278	0,8206	0,8403

From Table 5.2 we can see that, as expected, the Dice Coefficient, precision, and recall for the

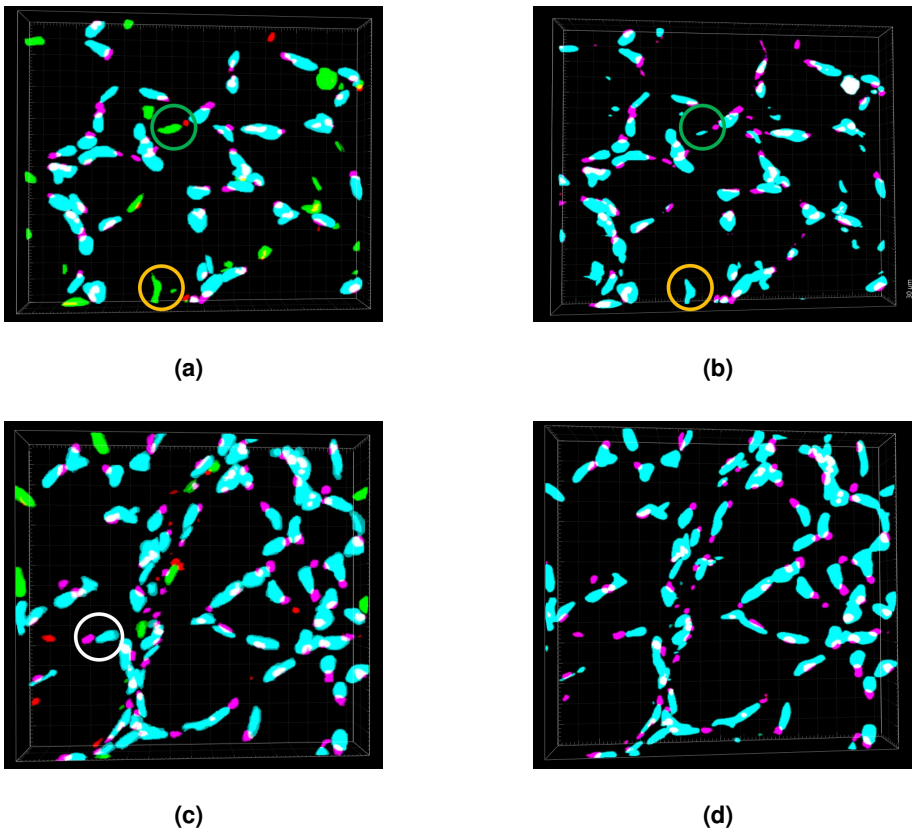


Figure 5.6: 3D visualization of segmentation masks: (a) 3D ground truth I_{31}^{gt}; (b) 3 class 3D U-Net model tested on I_1^{Pre}; (c) 3D ground truth I_{32}^{gt}; (d) 3 class 3D U-Net model tested on I_2^{Pre}.

nuclei and Golgi classes are comparable to the values obtained previously. Looking at the results for the nucleus-Golgi pairs class, we can see from the segmentation masks in figure 5.6 that the model is generally able to correctly identify this class, as reflected by a recall of 0.8403. However, the yellow circles in figure 5.6 mark cases where the model is apparently unable to distinguish that an isolated nucleus or Golgi are not part of a pair and therefore should not be identified in the segmentation mask for the nucleus-Golgi pairs. Therefore, it also detects a lot of digital noise, which is reflected in the lower precision of 0.6941. The reason the model cannot learn these cases could be that these cases of isolated nuclei and Golgi are relatively rare and do not occur often enough in the dataset for the model to distinguish them.

As mentioned above, the ground truth mask for the Golgi and the nuclei has some imperfections. The main one is the fact that in the cases where the nuclei and the Golgi pairs do not overlap, no connection is made between them, which makes it difficult for the model to distinguish, for example, an isolated Golgi from a paired one. An example of this case is highlighted by the white circle in figure 5.6. Additionally, some of the nucleus-Golgi pairs are not annotated in the ground truth masks, but are correctly identified

by the model, which also affects the precision results for the nucleus-Golgi pairs class. An example of this case is highlighted by a green circle in figure 5.6.

5.3.2 Vox2Vox

In this subsection the results obtained by training and testing the Vox2Vox model are presented and discussed.

Implementation Details

As mentioned earlier, the Vox2Vox model consists of two networks, a generator and a discriminator. The discriminator is used only to improve the training, the generator is the segmentation network we want to obtain. In these two networks, the following specifications were set for training: He normal initialization for the model weights; the Adam Optimizer with a learning rate of 2e-5; batch size of 4 and early stopping to stop training when the validation loss did not improve for 20 epochs, saving the weights of the best model based on the lowest validation loss.

Specifically for the Vox2Vox generator, the sigmoid activation function was used in the last layer so that the output has n-probability maps (where n is the number of classes). A threshold of 0.5 is then applied to each probability map of the output of Vox2Vox to obtain the final segmentation mask volume. The hyper-parameter α from the generator loss defined previously in equation 4.9 is set to $\alpha = 5$ as proposed in [54].

In a first phase, the model was trained with a loss function for the generator that gave the same weight to the nuclei and Golgi. However, with this loss function, the model had great difficulty detecting the Golgi on the images. This is likely due to the fact that many of the patches used to train the model do not contain Golgi, making it difficult for the model to learn how to detect and segment this class. To improve the results, the loss function of the generator was modified to give more weight to the Golgi dice coefficient loss. The term DLoss in the equation 4.9 was changed to

$$DLoss_{total} = \beta_1 DLoss(y_N, \hat{y}_N) + \beta_2 DLoss(y_G, \hat{y}_G) \tag{5.2}$$

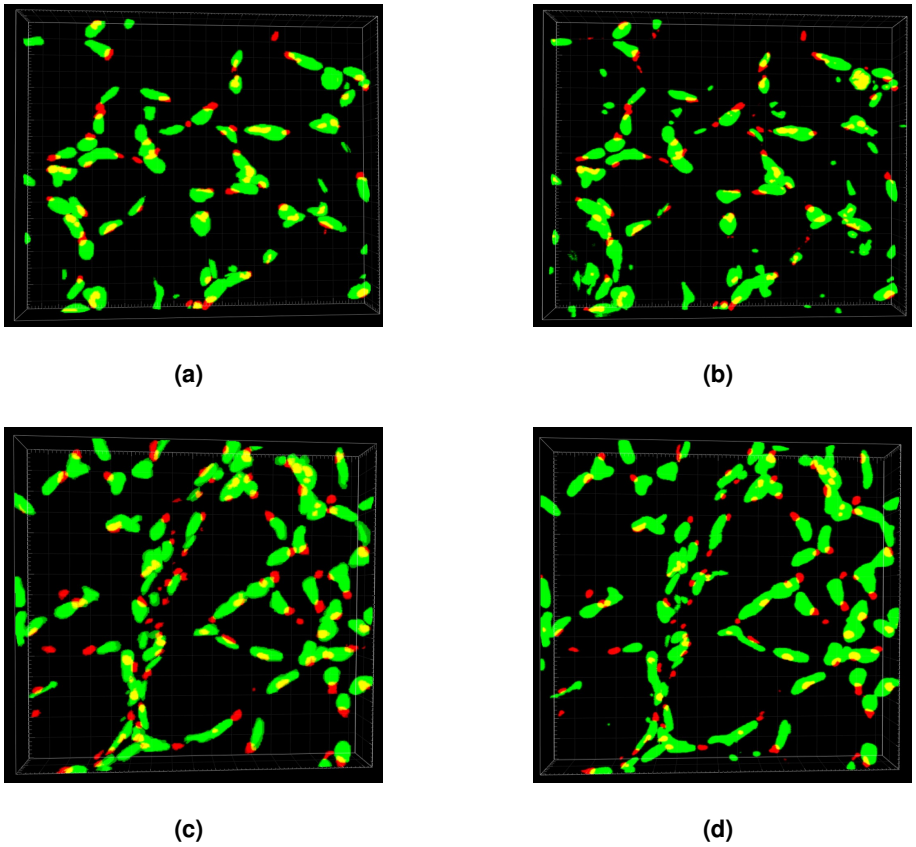
where y_N and y_G are the ground truth segmentation masks of nuclei and Golgi, respectively, and $\hat{y}_N$ and $\hat{y}_G$ are the predicted segmentation masks of nuclei and Golgi, respectively. We found the most appropriate values for β_1 and β_2 by trial and error, and after some experiments, we set $\beta_1 = 1$ and $\beta_2 = 6$.

2 Class

The Vox2Vox model was trained with the pre-processed dataset and tested with the two microscopic image volumes I_1^{Pre} and I_2^{Pre}. Figure 5.7 is a 3D visualization of the segmentation results for these volumes and table 5.3 shows the average quantitative metrics.

Table 5.3: Average metric values obtained from testing the 2 class Vox2Vox model on two pre-processed microscopic images

DC Nuclei	DC Golgi	Precision Nuclei	Precision Golgi	Recall Nuclei	Recall Golgi
0,7539	0,6883	0,6529	0,7078	0,8933	0,6917



(a) **(b)**

(c) **(d)**

Figure 5.7: 3D visualization of segmentation masks: (a) 3D ground truth I_{21}^{gt}; (b) 2 class Vox2Vox model tested on I_1^{Pre}; (c) 3D ground truth I_{22}^{gt}; (d) 2 class Vox2Vox model tested on I_2^{Pre}.

From the results for the dice coefficient, we can conclude that the model is successful in identifying most of the nuclei and Golgi in the microscopic image volumes. For the Golgi segmentation mask, we verified in figure 5.7 that the model correctly identifies all Golgi. However, the margins for all Golgi are smaller than the ground truth segmentation mask, resulting in lower recall. This is likely due to the fact that the ground truth segmentation masks have low resolution, as shown in Figure 5.5 (c), and also

to discontinuities in 3D, since these masks are created slide by slide in 2D. Consequently, the Vox2Vox model learns to generate images with this low resolution, which makes it difficult to segment the margins, especially of small objects like Golgi.

For the cell nuclei, we found that the model is very sensitive to noise in this class. This is particularly evident in the I_1^{Pre} image test, as part of this image contains a lot of background noise for the nucleus channel and this noise is detected by the model, resulting in lower precision. However, this also results in the model being able to detect Golgi that are difficult to classify, i.e., Golgi with low contrast in the segmentation mask.

3 Class

In this method, the number of output probability maps of Vox2Vox generator model is three, so we can classify the nuclei, Golgi and nucleus-Golgi pairs of the input images. All other parameters described in the implementation details are retained, except for the generator loss function, where we add another term to account for the Dice Loss of the nucleus-Golgi pairs class and set a gain β_3 = 1. The model was then trained using the pre-processed dataset and respective ground truth masks.

After training, the performance of the model is tested on the two microscopic image volumes I_1^{Pre} and I_2^{Pre}. Figure 5.8 is a 3D visualization of the segmentation results for these volumes and table 5.4 shows the average quantitative results.

Table 5.4: Average metric values obtained from testing the 3 class Vox2Vox model on two pre-processed microscopic images

DC Nuclei	DC Golgi	DC Pair	Precision Nuclei	Precision Golgi	Precision Pair	Recall Nuclei	Recall Golgi	Recall Pair
0,7464	0,6720	0,7198	0,7170	0,6596	0,6763	0,7784	0,7204	0,7721

If we compare the results presented in table 5.3 and 5.4 for the nuclei and Golgi classes, we verify that they remain relatively the same, as expected. Now, regarding the nucleus-Golgi pairs class, although the model is able to detect most of the nucleus-Golgi pairs in the images, the problem previously verified for the U-Net model remains in the Vox2Vox model, i.e., the model is not able to distinguish nucleus-Golgi pairs from individual nuclei and Golgi, which translates into lower precision.

In figure 5.8 (d), it is highlighted by yellow circles that for this class, the model has difficulty segmenting small objects such as Golgi, which lowers the recall. This is also related to the fact that the ground truth segmentation masks have low resolution and the model learns to generate low resolution images as well.

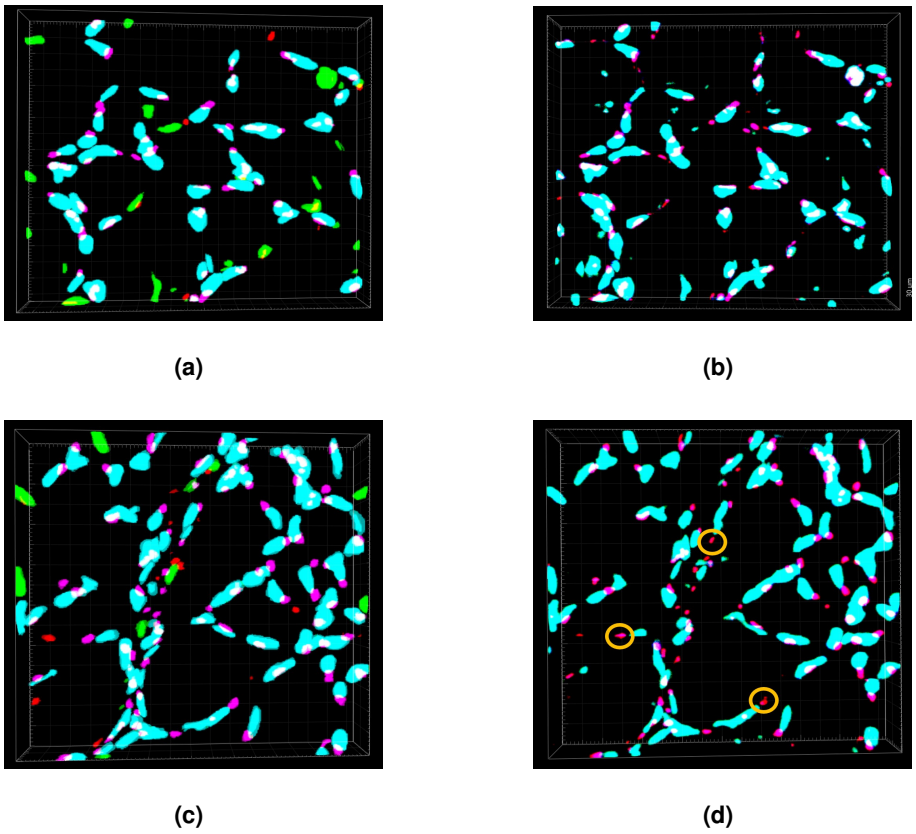


Figure 5.8: 3D visualization of segmentation masks: (a) 3D ground truth I_{31}^{gt}; (b) 3 class Vox2Vox model tested on I_{1}^{Pre}; (c) 3D ground truth I_{32}^{gt}; (d) 3 class Vox2Vox model tested on I_{2}^{Pre}.

5.3.3 Proposed Approach - CycleGAN

In this subsection the results obtained by training and testing the proposed unsupervised approach model CycleGAN are presented and discussed.

Implementation Details

As mentioned in section 4.2, the CycleGAN model consists of four interconnected networks, two generators and two discriminators. After training, we keep only the generator corresponding to the segmentation network (G_S), the rest of the networks are used only to improve training.

For training the CycleGAN we used: an initializer that generates weights with a normal distribution; the ADAM optimizer with a learning rate of 1e-3 and 5e-3 for the generators and discriminators, respectively; $\lambda_{cyc} = 10$ and $\lambda_{id} = 5$ in equation 4.5 as proposed in [8] and a batch size of 2.

Specifically for the generators, the activation function Tanh was used in the last layer so that the output has n-probability maps (where n is the number of classes). Therefore, for these experiments, the input images had to be normalized to be in the range [-1,1]. Then the output images are normalized

again to a range between [0,1]. For the segmentation network output (G_S), a threshold of 0.5 is then applied to each probability map to obtain the final segmentation mask volume.

During training, the performance of the model is tracked by using the generator models to generate translated versions of a few randomly selected images at the end of each epoch. The model is stopped when it reaches collapse mode, i.e., when it produces exactly the same output image for different input images. After training, we visually analyse the results obtained by CycleGAN for different epochs and select the model from the epoch that gives better results for the validation dataset.

Similar to Vox2Vox, we first trained the CycleGAN model by giving the Golgi and nuclei classes the same weight in the loss function. However, during training, we found that the model collapsed for the Golgi channel very quickly before it had a chance to improve. Therefore, similar to the Vox2Vox model, we changed the cycle consistency and identity loss to give more weight to the mean absolute error for Golgi. After some experiments the final weight given to this class was 3.5 and 1 to the nuclei class.

2 Class

The CycleGAN was trained in two different ways: first in a supervised manner using the pre-processed microscopic image dataset together with the corresponding manually labelled segmentation masks to ensure the correct implementation of the model, and then in an unsupervised way with the synthetic segmentation mask dataset described in subsection 4.1.1. Then, both models were tested with the two microscopic image volumes I_1^{Pre} and I_2^{Pre}. Figure 5.9 is a 3D visualization of the segmentation results for these volumes and table 5.5 shows the average quantitative metrics.

Table 5.5: Average metric values obtained from training the 2 class CycleGAN model in supervised and unsupervised way and testing these models on two pre-processed microscopic images

	DC Nuclei	DC Golgi	Precision Nuclei	Precision Golgi	Recall Nuclei	Recall Golgi
CycleGAN (Supervised)	0,7612	0,6545	0,6894	0,6004	0,8528	0,7450
CycleGAN (Unsupervised)	0,7664	0,6427	0,6903	0,5468	0,8629	0,8038

Based on figures 5.9 (b) and (e), we can verify that the supervised CycleGAN model successfully detects almost all nuclei and Golgi in the images. The advantage of this model is that, even if trained in a supervised manner, it does not require pairing of the microscopic image and the corresponding segmentation mask during training.

From figures 5.9 (b) and (e), we can also observe that the supervised CycleGAN model for the class of nuclei has difficulty detecting some nuclei that have low contrast in the microscopic images, which affects recall, and that it is very sensitive to digital noise, which affects the value of the precision. The same problems occur with the Golgi class, but are intensified due to the small size of the Golgi, which makes it difficult for the model to learn to detect it correctly and especially to segment well the borders.

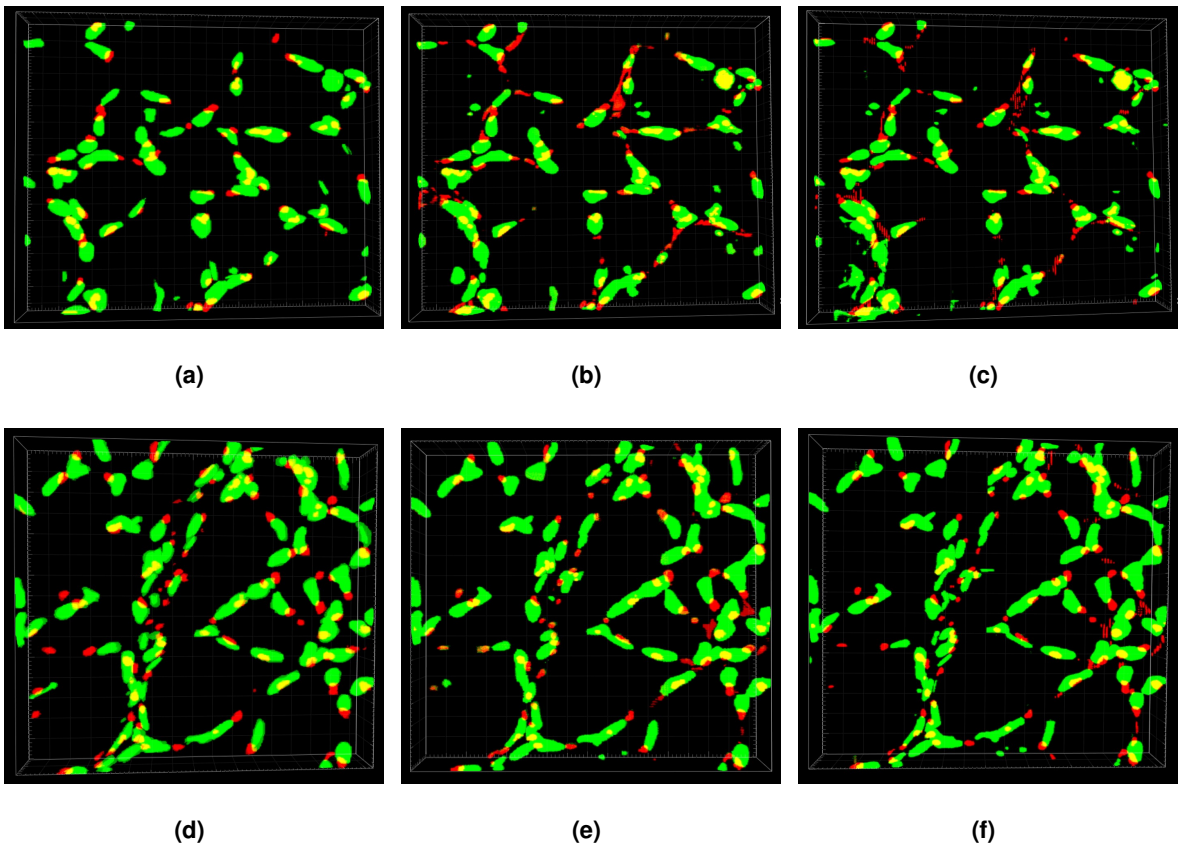


Figure 5.9: 3D visualization of segmentation masks: (a) 3D ground truth I_{21}^{gt}; (b) 2 class supervised CycleGAN model tested on I_1^{Pre}; (c) 2 class unsupervised CycleGAN model tested on I_1^{Pre}; (d) 3D ground truth I_{22}^{gt}; (e) 2 class supervised CycleGAN model tested on I_2^{Pre}; (f) 2 class unsupervised CycleGAN model tested on I_2^{Pre}.

If we now analyse the results of the unsupervised CycleGAN model, we find that, like the supervised model, it correctly detects almost all nuclei and Golgi in the images. From table 5.5, we verified that the results for the supervised and unsupervised CycleGAN models are very close. This suggests that the problems exhibited by the CycleGAN model are due only to the limitations of the model and not to the use of synthetic masks instead of the real ones to train the model. Actually, it has been verified that the model does not learn to generate masks with low resolution, as is the case with the supervised model, because the synthetic masks have better resolution than the manually labelled masks.

3 Channel

In this method, the number of output probability maps for the CycleGAN generator networks is three, so we can classify the nuclei, Golgi, and nucleus-Golgi pairs of the input images. All other parameters described in the implementation details are kept, except for cycle consistency and identity, where we add another term to account for the mean absolute error of the nucleus-Golgi pair class, and set a gain

equal to 1. The model was then trained with the pre-processed dataset and synthetic segmentation masks described in subsection 4.1.1.

After training, the performance of the model is tested on the two microscopic image volumes I_1^{Pre} and I_2^{Pre}. Figure 5.10 is a 3D visualization of the segmentation results for these volumes and table 5.6 shows the average quantitative results.

Table 5.6: Average metric values obtained from testing the 3 class unsupervised CycleGAN model on two pre-processed microscopic images

DC Nuclei	DC Golgi	DC Pair	Precision Nuclei	Precision Golgi	Precision Pair	Recall Nuclei	Recall Golgi	Recall Pair
0,7261	0,5942	0,6999	0,7174	0,5874	0,6711	0,7366	0,6133	0,7314

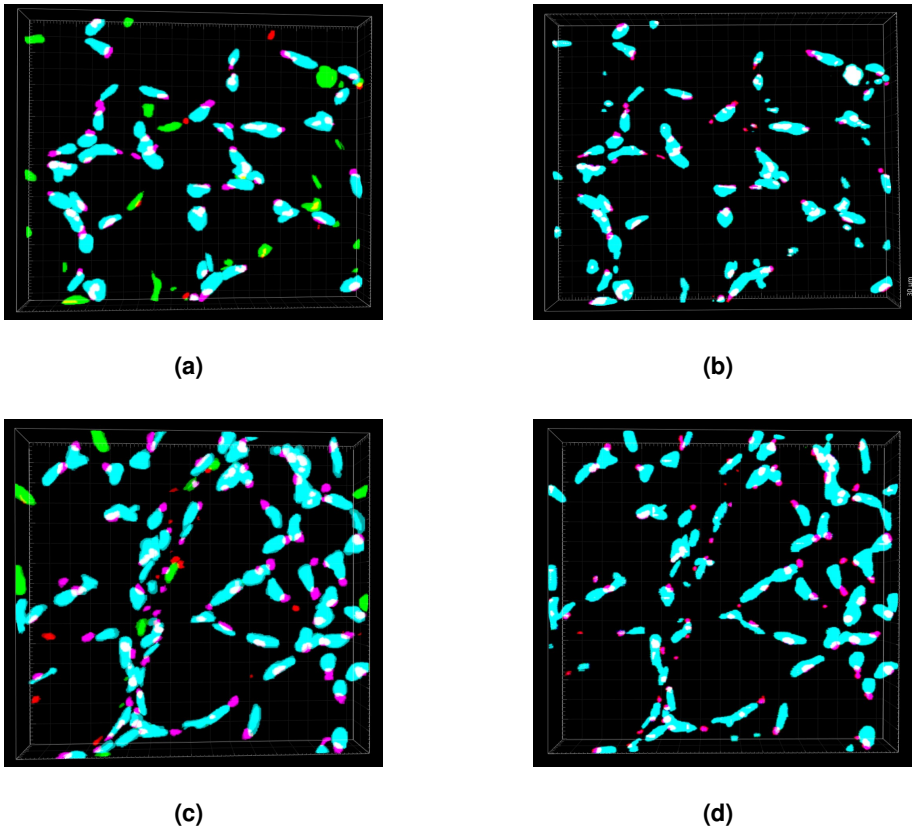


Figure 5.10: 3D visualization of segmentation masks: (a) 3D ground truth I_{31}^{gt}; (b) 3 class unsupervised CycleGAN model tested on I_1^{Pre}; (c) 3D ground truth I_{32}^{gt}; (d) 3 class unsupervised CycleGAN model tested on I_2^{Pre}.

From figure 5.9 and 5.10, we can observe that the model is less sensitive to digital noise and background clutter which is reflected in an improvement in precision for the nuclei and Golgi classes. However, it has more difficulty in detecting all nuclei and Golgi in the images

For the nuclei-Golgi pairs class, we can also see in figure 5.10 that, as with the previous U-Net and Vox2Vox models, the model is unable to distinguish isolated nuclei and Golgi from paired ones, but it

can detect most pairs in the image. An effect that was also confirmed is that for some isolated nuclei, the model generates a Golgi inside the nucleus instead of simply not classifying it as a pair.

5.3.4 Execution time

In this section, we present the time needed to train and test the models. As mentioned earlier, we have a total of 8 crops for training and testing the models. Where 6 of these crops are used to train the models and 2 of them are used for testing. It was estimated how long it takes to manually label nuclei and Golgi in a crop. The estimate was 28 hours. For training and testing time, the time needed to obtain these crops was considered. In the case of CycleGAN, the training time was added to the time needed to create the synthetic masks (40 minutes). The final values are presented in table 5.7.

Table 5.7: Time needed to train and test the models

	U-Net	Vox2Vox	CycleGAN
Training Time	(28h * 6) + 66 min	(28h * 6) + 74 min	40 min + 28,6 h
Testing Time	(28h * 2) + 35 seconds	(28h * 2) + 35 seconds	(28h * 2) + 35 seconds
Total Time	225,1 hours	225,23 hours	85,3 hours

5.3.5 Comparison between approaches

In this section, the segmentation results obtained with the supervised models U-Net and Vox2Vox are compared with the proposed unsupervised model CycleGAN. The results obtained for each class nuclei, Golgi and nucleus-Golgi pair are analysed separately.

Segmentation of the nuclei class is challenging because the microscopic images have some background clutter and some of the nuclei have low contrast in the images. For this class, the best dice coefficient was obtained for the U-Net model with 0,7775, but CycleGAN obtained a similar result with 0,7612 and then Vox2Vox with 0,7464. The precision and recall values for the CycleGAN and U-Net models are also very close, with the CycleGAN model only being inferior by less than one percent. The lower dice coefficient of Vox2Vox is due to its high sensitivity to digital noise, which explains its relatively low precision of 0,6529. However, it is the best of the 3 models at detecting Golgi, as it can detect and segment nuclei even when they have low contrast.

The challenges in classifying the Golgi class are mainly: the digital noise coming from particles such as dust entering the analysed sample and producing blurs (due to their autofluorescence) with high contrast in the red channel of the microscopic images, which can be mistaken for Golgi; and the small size of the Golgi, which makes detection and boundary segmentation difficult. For this class, the best dice coefficient was obtained for the U-Net model with 0,6938, followed by the Vox2Vox model with 0,6883 and

the CycleGAN model with 0,6427. Although the U-Net model had the best Dice Coefficient, because it was able to detect and segment the Golgi best on average, the Vox2Vox model obtained the best precision with 0,7078 and CycleGAN the best recall with 0,8038. Analysing the segmentation masks obtained for these models, we can conclude that the Vox2Vox model best segments the boundaries of the nuclei it detects and CycleGAN is the model that detects the most Golgi on the images.

For the nucleus-Golgi pair class, the main classification challenge is that some nuclei and Golgi do not form a pair and not all nucleus and Golgi of a pair overlap. For this class, the best dice coefficient was obtained for the U-Net with 0,7601, followed by the Vox2Vox model with 0,7198 and then CycleGAN with 0,6999. In terms of accuracy, all the models are very close to each other as they all have the same problem of incorrectly detecting individual nuclei and Golgi as being paired. However, the best model at detecting the pairs on the image is the U-Net. Probably the Vox2Vox model and CycleGAN should be more complex (more kernels or more layers) to be able to better segment this class.

As mentioned earlier, supervised models are highly dependent on the data on which they are trained. If these data have errors or are imperfect, these imperfections are propagated to the models. We have found that the ground-truth masks have problems, such as not all objects are segmented and the masks have low resolution. Also, it is very difficult to obtain these masks. With the CycleGAN model, we were able to obtain comparable results to the supervised models in less than half the time, as demonstrated in section 5.3.4. Since this model does not require extensive manual work and also has the advantage of being more transferable, i.e., it does not depend so much on the data on which it is trained, it learns better the actual distribution of the data.

6

Conclusion

Contents

Introductory text

6.1 Conclusions

6.2 Model Limitations and Future Work

Bibliography

[1] F. Xing and L. Yang, "Robust Nucleus/Cell Detection and Segmentation in Digital Pathology and Microscopy Images: A Comprehensive Review," *IEEE Reviews in Biomedical Engineering*, vol. 9, pp. 234–263, 2016. [Online]. Available: https://doi.org/10.1109%2Frbme.2016.2515127

[2] T. Hayakawa, S. Prasath, H. Kawanaka, B. Aronow, and S. Tsuruoka, "Computational Nuclei Segmentation Methods in Digital Pathology: A Survey," *Archives of Computational Methods in Engineering*, vol. 28, pp. 1–13, 01 2021.

[3] R. Benninger, M. Hao, and D. Piston, "Multi-photon excitation imaging of dynamic processes in living cells and tissues," *Reviews of physiology, biochemistry and pharmacology*, vol. 160, pp. 71–92, 02 2008.

[4] A. Hallou, H. Yevick, B. Dumitrascu, and V. Uhlmann, "Deep learning for bioimage analysis," *arXiv preprint arXiv:2107.02584*, 2021.

[5] F. Xing, Y. Xie, H. Su, F. Liu, and L. Yang, "Deep Learning in Microscopy Image Analysis: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4550–4568, 2018.

[6] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation," *ArXiv*, vol. abs/1606.06650, 2016.

[7] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5967–5976.

[8] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2242–2251.

[9] H. Narotamo, M. Ouarné, C. A. Franco, and M. Silveira, "Joint Segmentation and Pairing of Nuclei

and Golgi in 3D Microscopy Images," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, 2021, pp. 3017–3020.

[10] J. Chai, H. Zeng, A. Li, and E. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications*, vol. 6, p. 100134, 08 2021.

[11] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[12] X. Liu, L. Song, S. Liu, and Y. Zhang, "A Review of Deep-Learning-Based Medical Image Segmentation Methods," *Sustainability*, vol. 13, no. 3, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/3/1224

[13] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, "A Guide to Convolutional Neural Networks for Computer Vision," *Synthesis Lectures on Computer Vision*, vol. 8, no. 1, pp. 1–207, 2018.

[14] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[15] F. Nahhas, H. Shafri, M. Al-Zuhairi, B. Pradhan, and S. Mansor, "Deep Learning Approach for Building Detection Using LiDAR-Orthophoto Fusion," *Journal of Sensors*, vol. 2018, 08 2018.

[16] Y. Xing, L. Zhong, and X. Zhong, "An Encoder-Decoder Network Based FCN Architecture for Semantic Segmentation," *Wireless Communications and Mobile Computing*, vol. 2020, pp. 1–9, 07 2020.

[17] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[18] H. Wang, J. Lin, and Z. Wang, "Design Light-weight 3D Convolutional Networks for Video Recognition Temporal Residual, Fully Separable Block, and Fast Algorithm," *ArXiv*, vol. abs/1905.13388, 2019.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[20] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *Advances in Neural Information Processing Systems*, vol. 3, 06 2014.

[22] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative Adversarial Networks: An Overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[23] Y. Kang, S. Gao, and R. Roth, "Transferring Multiscale Map Styles Using Generative Adversarial Networks," *International Journal of Cartography*, vol. 5, 05 2019.

[24] F. Xing and L. Yang, "Robust Nucleus/Cell Detection and Segmentation in Digital Pathology and Microscopy Images: A Comprehensive Review," *IEEE reviews in biomedical engineering*, vol. 9, 01 2016.

[25] S. Lee, P. Salama, K. W. Dunn, and E. J. Delp, "Segmentation of fluorescence microscopy images using three dimensional active contours with inhomogeneity correction," in *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, 2017, pp. 709–713.

[26] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *INTERNATIONAL JOURNAL OF COMPUTER VISION*, vol. 1, no. 4, pp. 321–331, 1988.

[27] T. Chan and L. Vese, "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266–277, 2001.

[28] K. S. Lorenz, P. Salama, K. W. Dunn, and E. J. Delp, "Three dimensional segmentation of fluorescence microscopy images using active surfaces," in *2013 IEEE International Conference on Image Processing*, 2013, pp. 1153–1157.

[29] G. Paul, J. Cardinale, and I. F. Sbalzarini, "Coupling Image Restoration and Segmentation: A Generalized Linear Model/Bregman Perspective," *Int. J. Comput. Vision*, vol. 104, no. 1, p. 69–93, aug 2013. [Online]. Available: https://doi.org/10.1007/s11263-013-0615-2

[30] A. Rizk, G. Paul, P. Incardona, M. Bugarski, M. Mansouri, A. Niemann, U. Ziegler, P. Berger, and I. Sbalzarini, "Segmentation and quantification of subcellular structures in fluorescence microscopy images using Squassh," *Nature protocols*, vol. 9, pp. 586–96, 03 2014.

[31] O. Dzyubachyk, W. A. van Cappellen, J. Essers, W. J. Niessen, and E. Meijering, "Advanced Level-Set-Based Cell Tracking in Time-Lapse Fluorescence Microscopy," *IEEE Transactions on Medical Imaging*, vol. 29, no. 3, pp. 852–867, 2010.

[32] A. Dufour, V. Shinin, S. Tajbakhsh, N. Guillen-Aghion, J.-C. Olivo-Marin, and C. Zimmer, "Segmenting and tracking fluorescent cells in dynamic 3-D microscopy with coupled active surfaces," *IEEE Transactions on Image Processing*, vol. 14, no. 9, pp. 1396–1410, 2005.

[33] S. Arslan, T. Ersahin, R. Cetin-Atalay, and C. Gunduz-Demir, "Attributed Relational Graphs for Cell Nucleus Segmentation in Fluorescence Microscopy Images," *IEEE Transactions on Medical Imaging*, vol. 32, no. 6, pp. 1121–1131, 2013.

[34] S. Anwar, M. Majid, A. Qayyum, M. Awais, M. Alnowami, and K. Khan, "Medical image analysis using convolutional neural networks: A review," *Journal of Medical Systems*, vol. 42, p. 226, 10 2018.

[35] F. Xing, Y. Xie, and L. Yang, "An Automatic Learning-Based Framework for Robust Nucleus Segmentation," *IEEE Transactions on Medical Imaging*, vol. 35, no. 2, pp. 550–566, 2016.

[36] N. Kumar, R. Verma, S. Sharma, S. Bhargava, A. Vahadane, and A. Sethi, "A Dataset and a Technique for Generalized Nuclear Segmentation for Computational Pathology," *IEEE Transactions on Medical Imaging*, vol. 36, no. 7, pp. 1550–1560, 2017.

[37] A. Carpenter, T. Jones, M. Lamprecht, C. Clarke, I. Kang, O. Friman, D. Guertin, J. Chang, R. Lindquist, J. Moffat, P. Golland, and D. Sabatini, "CellProfiler: Image analysis software for identifying and quantifying cell phenotypes," *Genome biology*, vol. 7, p. R100, 02 2006.

[38] H. Qu, P. Wu, Q. Huang, J. Yi, G. M. Riedlinger, S. De, and D. N. Metaxas, "Weakly Supervised Deep Nuclei Segmentation using Points Annotation in Histopathology Images," in *Proceedings of The 2nd International Conference on Medical Imaging with Deep Learning*, ser. Proceedings of Machine Learning Research, vol. 102. PMLR, 08–10 Jul 2019, pp. 390–400. [Online]. Available: https://proceedings.mlr.press/v102/qu19a.html

[39] Z. Zhao, L. Yang, H. Zheng, I. H. Guldner, S. Zhang, and D. Z. Chen, "Deep learning based instance segmentation in 3D biomedical images using weak annotation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2018, pp. 352–360.

[40] H. Chen, Q. Dou, L. Yu, J. Qin, and P.-A. Heng, "VoxResNet: Deep voxelwise residual networks for brain segmentation from 3D MR images," *NeuroImage*, vol. 170, pp. 446–455, 2018, segmenting the Brain. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1053811917303348

[41] D. J. Ho, C. Fu, P. Salama, K. W. Dunn, and E. J. Delp, "Nuclei Segmentation of Fluorescence Microscopy Images Using Three Dimensional Convolutional Neural Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 834–842.

[42] C. Fu, D. J. Ho, S. Han, P. Salama, K. W. Dunn, and E. J. Delp, “Nuclei segmentation of fluorescence microscopy images using convolutional neural networks,” in *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, 2017, pp. 704–708.

[43] D. J. Ho, C. Fu, P. Salama, K. W. Dunn, and E. J. Delp, “Nuclei detection and segmentation of fluorescence microscopy images using three dimensional convolutional neural networks,” in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, 2018, pp. 418–422.

[44] Y. Li and L. Shen, “cC-GAN: A Robust Transfer-Learning Framework for HEp-2 Specimen Image Segmentation,” *IEEE Access*, vol. 6, pp. 14 048–14 058, 2018.

[45] W. Hu, H. Sheng, J. Wu, Y. Li, T. Liu, Y. Wang, and Y. Wen, “Generative Adversarial Training for Weakly Supervised Nuclei Instance Segmentation,” in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 3649–3654.

[46] F. Mahmood, D. Borders, R. J. Chen, G. N. McKay, K. J. Salimian, A. S. Baras, and N. J. Durr, “Deep Adversarial Training for Multi-Organ Nuclei Segmentation in Histopathology Images,” *CoRR*, vol. abs/1810.00236, 2018. [Online]. Available: http://arxiv.org/abs/1810.00236

[47] C. Fu, S. Lee, D. J. Ho, S. Han, P. Salama, K. W. Dunn, and E. J. Delp, “Three Dimensional Fluorescence Microscopy Image Synthesis and Segmentation,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 2302–23 028.

[48] D. J. Ho, S. Han, C. Fu, P. Salama, K. W. Dunn, and E. J. Delp, “Center-Extraction-Based Three Dimensional Nuclei Instance Segmentation of Fluorescence Microscopy Images,” *2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, May 2019. [Online]. Available: http://dx.doi.org/10.1109/BHI.2019.8834516

[49] K. W. Dunn, C. Fu, D. J. Ho, S. Lee, S. Han, P. Salama, and E. J. Delp, “DeepSynth: Three-dimensional nuclear segmentation of biological images using neural networks trained with synthetic data,” *Scientific reports*, vol. 9, no. 1, pp. 1–15, 2019.

[50] K. Yao, K. Huang, J. Sun, and C. Jude, “AD-GAN: End-to-end Unsupervised Nuclei Segmentation with Aligned Disentangling Training,” 2021.

[51] C. Chu, A. Zhmoginov, and M. Sandler, “CycleGAN, a Master of Steganography,” 2017.

[52] K. Yao, K. Huang, J. Sun, L. Jing, D. Huang, and C. Jude, “Scaffold-A549: A Benchmark 3D Fluorescence Image Dataset for Unsupervised Nuclei Segmentation,” *Cognitive Computation*, vol. 13, pp. 1–6, 11 2021.

[53] S. Jadon, “A survey of loss functions for semantic segmentation,” in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, 2020, pp. 1–7.

[54] M. D. Cirillo, D. Abramian, and A. Eklund, “Vox2Vox: 3D-GAN for Brain Tumour Segmentation,” *CoRR*, vol. abs/2003.13653, 2020. [Online]. Available: https://arxiv.org/abs/2003.13653

[55] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[56] M. Abadi *et al.*, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[57] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[58] J. Kim, *Image Enhancement for Improving Object Recognition*. Dordrecht: Springer Netherlands, 2014, pp. 73–106. [Online]. Available: https://doi.org/10.1007/978-94-017-9075-8_4

[59] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” 2017. [Online]. Available: https://arxiv.org/abs/1708.02002