

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
INSTITUTO DE INFORMÁTICA  
CURSO DE CIÊNCIA DA COMPUTAÇÃO

AUGUSTO DALCIN PEITER

**Semantic Segmentation of Cells in  
Microscope Images: Evaluating Data  
Augmentation Techniques**

Work presented in partial fulfillment  
of the requirements for the degree of  
Bachelor in Computer Science

Advisor: Prof. Dr. Claudio Rosito Jung

Porto Alegre  
April 2023

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof<sup>ª</sup>. Carlos André Bulhões Mendes

Vice-Reitora: Prof<sup>ª</sup>. Patricia Helena Lucas Pranke

Pró-Reitoria de Graduação: Prof<sup>ª</sup>. Cíntia Inês Boll

Diretora do Instituto de Informática: Prof<sup>ª</sup>. Carla Maria Dal Sasso Freitas

Coordenador do Curso de Ciência de Computação: Prof. Marcelo Walter

Bibliotecária-chefe do Instituto de Informática: Alexsander Borges Ribeiro

*“The struggle itself ... is enough to fill a man's heart.*

*One must imagine Sisyphus happy..”*

— ALBERT CAMUS, THE MYTH OF SISYPHUS

## **AGRADECIMENTOS**

Agradeço a minha família que me suportou (em ambos sentidos) no processo de desenvolvimento desse trabalho. Escrever um TCC é um processo bastante cansativo e requer bastante resiliência. Ambos os meus pais me ajudaram muito nesse aspecto, principalmente no suporte moral, manter o pique durante esse período não é fácil.

Gostaria de agradecer aos meus amigos e colegas também. Eu recebi muita ajuda no desenvolvimento desse trabalho, desde coisas pequenas como ajuda com as tecnologias usadas, relatos e dicas de quem já passou por esse processo e até mesmo ajuda na revisão do texto.

Finalmente gostaria de agradecer todos os professores e funcionários da UFRGS. Muitos deles tiveram grande impacto em minha formação acadêmica. Agradeço em especial meu orientador Claudio Jung, que me apoiou durante todo o processo, estando sempre disponível para discutir, marcar reuniões e resolver dúvidas. Eu sei que nem todo orientador se prontifica a se fazer tão acessível assim.

## ABSTRACT

Semantic image segmentation is a topic of computer vision that can be applied to many different fields of research. In biomedical research it is important to be able to distinguish the object of interest in the image from the background, particularly for microscopy images of cells. The segmentation of cells is a time consuming process if done by hand, many approaches to this problem use Deep Learning models to solve this. A common trait to these approaches is they are trained on a specific dataset and typically perform poorly when there is a shift on the domain of images to evaluate compared to the training dataset. We propose to analyze both the extent of domain shift/overfitting and mitigation techniques with the use of data augmentation. To evaluate our approach we will compare models with and without data augmentation cross validating through intra- and inter-dataset testing.

**Keywords:** Semantic segmentation. data augmentation. domain-shift. cell segmentation.

## Mitigando domain shift em segmentação de células usando data augmentation

### RESUMO

Segmentação semântica de imagens é um tópico de visão computacional que pode ser aplicado em vários domínios de pesquisa. No campo de pesquisa de biomedicina é importante ter a capacidade de distinguir um objeto de interesse em uma imagem do fundo, particularmente para imagens de microscopia de células. A segmentação de células é um processo demorado se feito a mão, muitas abordagens para esse problema usam modelos de Deep Learning para resolver isso. Uma característica em comum dessas abordagens é que elas são treinadas em um dataset específico e tipicamente não geram resultados bons quando há uma mudança no domínio de imagens a ser avaliadas comparado ao dataset de treinamento. Nós propomos analisar ambos a extensão do problema de domain-shift/overfitting e técnicas de mitigação através de data augmentation, usando validação cruzada através de testes intra e inter-dataset.

**Palavras-chave:** Segmentação Semântica, Data Augmentation, Domain-Shift, Segmentação de Células.

## LIST OF ABBREVIATIONS AND ACRONYMS

CTC	Cell Tracking Challenge
CNN	Convolutional Neural Networks
FCN	Fully Convolutional Network
ReLU	Rectified Linear Unit
RA	RandAugment
TA	TrivialAugment
RLR	Random Local Rotation
scSE	spatial and channel Squeeze and Excitation
SE	Squeeze and Excitation
IoU	Intersection-Over-Union
DSC	Dice similarity coefficient
DIC	Differential Interference Contrast
BF	Bright Field
Fluo	Fluorescent
PhC	Phase Contrast

## LIST OF FIGURES

Figure 1.1	Semantic Segmentation .....	11
Figure 1.2	Different kinds of augmentations .....	12
Figure 2.1	Example of different cell types present in dataset .....	15
Figure 2.2	Sample image before and after Otsu Thresholding .....	16
Figure 2.3	Watershed Segmentation.....	17
Figure 2.4	Watershed Segmentation on Cells .....	17
Figure 2.5	U-Net Architecture .....	19
Figure 2.6	UNet++ Architecture, black indicates the original U-Net, green and blue show dense convolution blocks on the skip pathways, and red indicates deep supervision.....	20
Figure 2.7	Different types of data augmentation .....	21
Figure 2.8	Transformations considered by RandAugment .....	21
Figure 2.9	Example of policy in search space.....	22
Figure 2.10	Comparison of GPU times and Test Error on different datasets for each approach.....	23
Figure 3.1	The EfficientUNet++ decoder is based on Unet++ decoder implementation (The components outside of the backbone, which is the encoder). .....	25
Figure 3.2	Comparison of EfficientUNet++ Block vs Unet++ Block.....	25
Figure 3.3	IoU Calculation visualized. ....	26
Figure 3.4	Sigmoid function visualized .....	27
Figure 3.5	The random local rotation data augmentation strategy. Operator represents pointwise multiplication. ....	28
Figure 3.6	Comparison of similar augmentations with CutMix. ....	29
Figure 4.1	Fluo-N2DL-HeLa cell type mask before and after threshold.....	32
Figure 4.2	Example of augmentations applied.....	34
Figure 4.3	Example of prediction on Fluo-N2DH-SIM+ of Scenario 4. ....	36



## LIST OF TABLES

Table 3.1 List of Augmentations in our work .....	30
Table 4.1 Samples per cell type.....	32
Table 4.2 List of parameters for our models .....	33
Table 4.3 Augmentation subsets for training .....	37
Table 4.4 Initial training of baseline model vs All Augmentations on DIC-C2DH- HeLa cell type .....	37
Table 4.5 Scenario 1: Train on DIC-C2DH-HeLa .....	38
Table 4.6 Scenario 2: Train on Fluo-N2DH-SIM+ .....	38
Table 4.7 Scenario 3: Train on All Cell Types, Evaluate on DIC-C2DH-HeLa .....	38
Table 4.8 Scenario 4: Train on All Cell Types, Evaluate on Fluo-N2DH-SIM+ .....	38

## CONTENTS

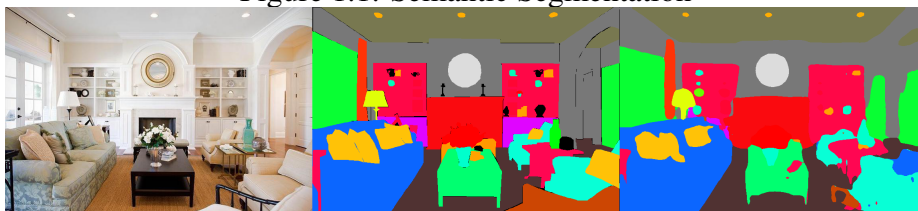
<b>1 INTRODUCTION.....</b>	<b>11</b>
<b>2 BACKGROUND AND RELATED WORKS.....</b>	<b>14</b>
2.1 Datasets .....	14
2.2 Classical Computer Vision Approaches.....	15
2.3 Deep Learning Approaches.....	17
2.4 Data Augmentation .....	19
<b>3 METHODOLOGY .....</b>	<b>24</b>
3.1 Model.....	24
3.2 Metrics .....	26
3.3 Data Augmentation Techniques.....	27
<b>4 EXPERIMENTAL RESULTS .....</b>	<b>31</b>
4.1 Dataset Preparation .....	31
4.2 Model configuration.....	33
4.3 Data Augmentation .....	33
4.4 Proposed Training and Evaluation.....	34
<b>5 CONCLUSION .....</b>	<b>39</b>
<b>REFERENCES.....</b>	<b>41</b>

## 1 INTRODUCTION

As biomedical studies advance, the ability to efficiently segment cells in microscopic imaging, isolating each cell inside an image from the background, becomes crucial in several applications. Although humans can visually determine cell morphology and delineate their borders, automating this process using semantic segmentation with deep learning networks can facilitate the processing of large amounts of biomedical images and help with analyzing the data. Considering the idea of automating this process, the introduction of image processing techniques such as semantic segmentation and instance segmentation was a big step towards shortening the amount of labor needed to process an image.

The purpose of a semantic segmentation algorithm is to categorize the elements of an image into different classes, assigning a unique label to each pixel. In the case of biomedical images, it can be used to differentiate among different cells, the background, cell nuclei, different types of tissues, and more, all into different categories to be visualized separately. This is represented in a segmentation mask, which is the output of the semantic segmentation and is an image that indicates which pixels in an image belong to a particular object or region. For instance, in cell segmentation a semantic mask can identify and segment cells from the background, the cells would have a unique label and the pixels corresponding to the cells would be set to a particular value in the mask. This can be visualized in the Figure 1.1, where we have the original image, the annotated mask and the output of a semantic segmentation prediction.

Figure 1.1: Semantic Segmentation



From left to right: Test Image, Ground Truth, Predicted Result

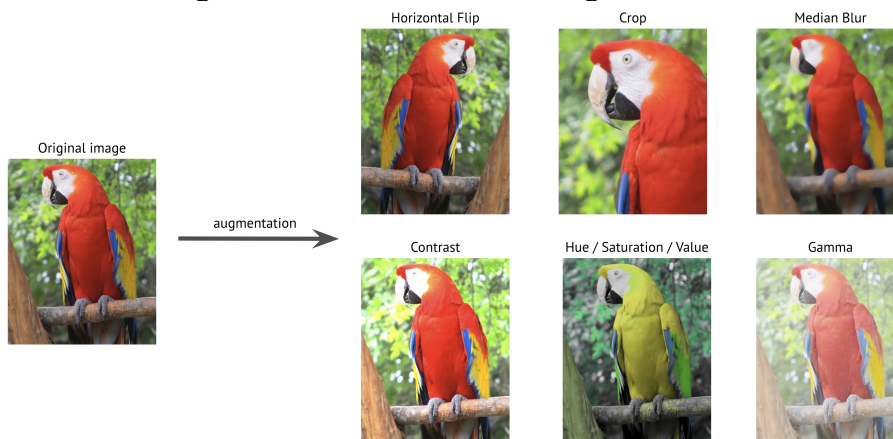
Source: (ZHOU et al., 2016; ZHOU et al., 2017)

There are cases in which semantic segmentation is not suitable for a task. One example is cell counting in microscopy images that are densely packed together: in this scenario, cells tend to overlap, making it difficult to tell how many there are in a given image. This problem can be addressed with instance segmentation algorithms, which are algorithms that can delineate and differentiate each instance of objects in an image (i.e.

identify each person in a group of people as a separate instance).

A common problem for many of the machine learning solutions that exist nowadays is they often fail to handle changes between training (source) and test (target) input distributions, as mentioned by Sun, Feng and Saenko (2015) – this problem is called domain shift. This is a problem that affect models that are trained on specific datasets and applied to different ones. A possible solution is to bridge the gap between datasets for domain shift, such as minimizing target and source distance or re-evaluating weights and features. Another way that does not require making external changes is through image augmentation, which consists of creating new training examples from the existing ones. To make a new sample, you slightly change the original image. For instance, it is possible to make a new image a little brighter; cut out a piece from the original image; make a new image by mirroring the original one, as can be seen in Figure 1.2. In regards to semantic segmentation some data augmentation are applied both on the image and the segmentation mask, like geometric transformations i.e., translation, rotation, scale.

Figure 1.2: Different kinds of augmentations



Source: (BUSLAEV et al., 2020)

The goal of this work is to check the extent of domain shift/overfitting in the context of cell segmentation, and explore mitigation strategies based on data augmentation. More precisely, we use a state-of-the-art segmentation architecture called EfficientUNet++ (SILVA et al., 2021), and explore various modern data augmenting techniques that aim to bridge the domain gap that exists in the field of cell segmentation, when models are trained on one dataset and tested on another. Our work has several parameters alongside a combination of various successful data augmentation techniques. These specifications are as follows:

- Training on the CTC dataset, which doesn't have a large amount of data like Ima-

geNet, thus increasing the importance of data augmentation.

- Use of a mixture of RandAugment (CUBUK et al., 2019) and TrivialAugment (MÜLLER; HUTTER, 2021) learning policies for practical data augmentation with a reduced search space modified with additional augmentations that can be applied.
- Alongside standard data augmentation techniques, implementation of two promising techniques in regards to the field of cell segmentation, CutMix by (YUN et al., 2019) and Random Local Rotation (RLR) (ALOMAR; AYSEL; CAI, 2023).

To analyze the effectiveness of the generalized model approach through data augmentation, we will perform a comparison of the models with and without data augmentation in an intra- and inter-dataset validation scheme.

The rest of this work is organized according to the following. Chapter 2 elaborates on the necessary technical background concepts to understand this work and explores works similarly related to these problems. In Chapter 3 we introduce our proposed solution. Chapter 4 pertains to the results and analysis of our implementation. Finally in Chapter 5, we present our conclusion.

## 2 BACKGROUND AND RELATED WORKS

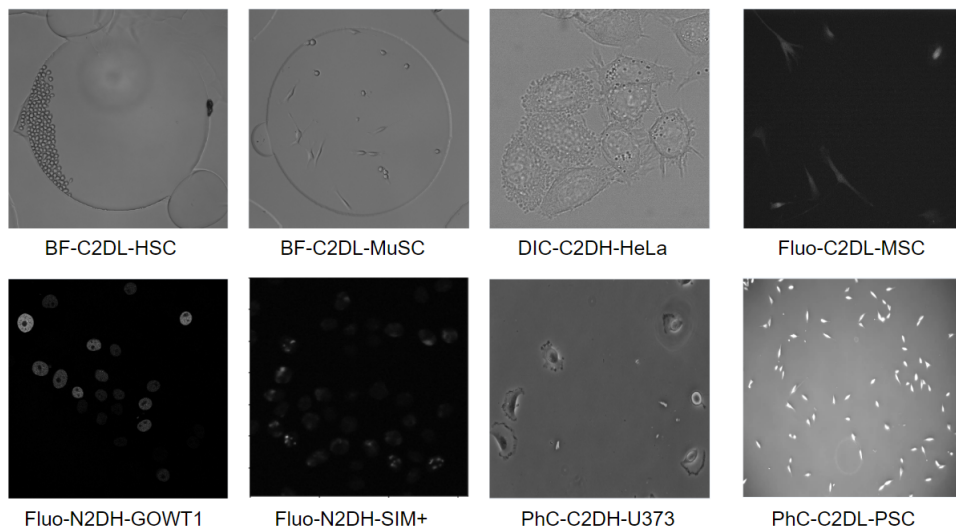
There are many differences in the process of machine learning compared to how a human learns information. Regarding semantic segmentation, to achieve the capability of classifying elements in an image, either through an image processing algorithm or a deep learning method, there is a need to recognize patterns on an image to be able to categorize its pixels. The concept of a pattern can have multiple degrees of complexity, a simple algorithm could detect all the straight lines in an image, or the edges of an object, but through the use of neural networks it can go as far as detecting human shapes. For the more complex goals, it is essential to have an ample amount of image samples through which a deep learning model can train and learn to recognize and classify new images. The quality of a dataset determines how difficult the learning process will be. It is not always possible to guarantee that a dataset has good quality samples inside, sometimes the images can be blurry, or noisy, or there may be few samples to learn from. Many of these are common problems that can be encountered when working with a dataset. This chapter abides by the following order. Section 2.1 provides information about the dataset used throughout this work. In Section 2.2, we describe classical computer vision approaches to semantic segmentation. In Section 2.3, we explore the deep learning approaches to semantic segmentation. Finally, Section 2.4 briefly describes data augmentation and presents different approaches on the subject through recent works.

### 2.1 Datasets

The dataset used in this work is an initiative presented by (ULMAN et al., 2017) aimed at promoting the development and objective evaluation of cell segmentation and tracking algorithms. The Cell Tracking Challenge (CTC) incentive provides various datasets that consist of 2D and 3D time-lapse video sequences of fluorescent counter-stained nuclei or cells moving on top or immersed in a substrate, along with 2D Bright Field, Phase Contrast, and Differential Interference Contrast (DIC) microscopy videos of cells moving on a flat substrate. The videos cover a wide range of cell types and quality (spatial and temporal resolution, noise levels, etc.) In addition, they provide 2D and 3D videos of synthetic fluorescently stained cells and nuclei of different shapes and motion patterns.

There are several different types of cells in this dataset. For our purposes, however,

Figure 2.1: Example of different cell types present in dataset



Source: (ULMAN et al., 2017)

we will consider them to pertain to the same class – a cell. Hence, each pixel in the images will be classified as either background or cell, yielding a binary classification problem. In order to keep our work on a simpler scope we will avoid the 3D data and only focus on 2D imagery. These images are suitable for our use case because the number of annotated samples is not large, which motivates the use of data augmentation for obtaining more training samples. Also, the different cell types are visually different, which might cause a domain gap when training a model with one cell type and then evaluating it with another.

Each cell type dataset is classified according to a naming convention separated in three parts. The first name denotes microscopy modality, which can be either fluorescence (Fluo), differential interference contrast (DIC), bright field (BF) or phase contrast (PhC). The following name describes the cell image’s capture characteristics, firstly the staining, either nuclear (N) or cellular (C), the dimensionality (2D or 3D) and the resolution (low (L), high (H)). The last name is the cell type or model organism used. Through the information provided by the name of the cell type datasets, we are given an idea of which dataset images provide more information and are thus less likely to suffer from domain shift.

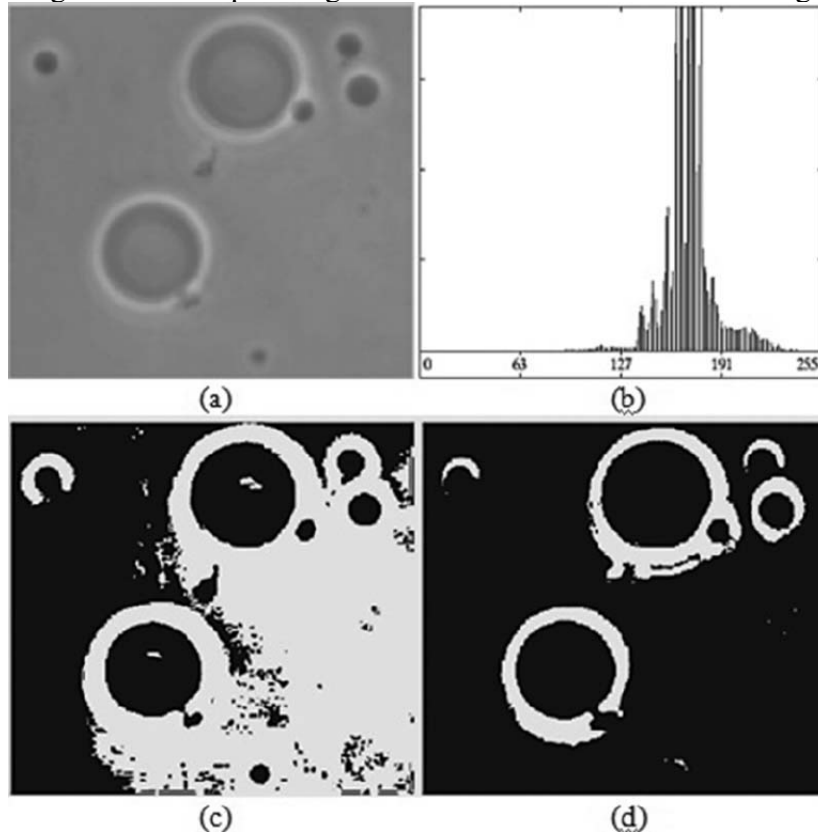
## 2.2 Classical Computer Vision Approaches

Image segmentation is a fundamental problem that arises in many different domains besides biomedical research, as such, a lot of approaches have been developed and

improved over the years to solve this problem. The simplest methods use traditional computer vision techniques to perform segmentation, without relying on AI-based solutions.

In regards to traditional approaches for image segmentation that do not rely on deep learning, the most basic implementation is simple binarization. The underlying principle is to classify an image pixel into foreground and background by selecting a threshold point in regard to pixel intensity. Although a threshold can be chosen manually, it is preferable to make the process automatic. Otsu's method by Otsu (1979) is a well-known technique for binary thresholding. In this approach, the threshold is determined by minimizing intra-class intensity variance, or equivalently, by maximizing inter-class variance, as described by the author. This method can also be extended to perform multi-level thresholding, allowing image classification beyond two classes. In this approach it is also possible to apply post processing techniques like closing or dilation along with a connected components algorithm to further improve results.

Figure 2.2: Sample image before and after Otsu Thresholding



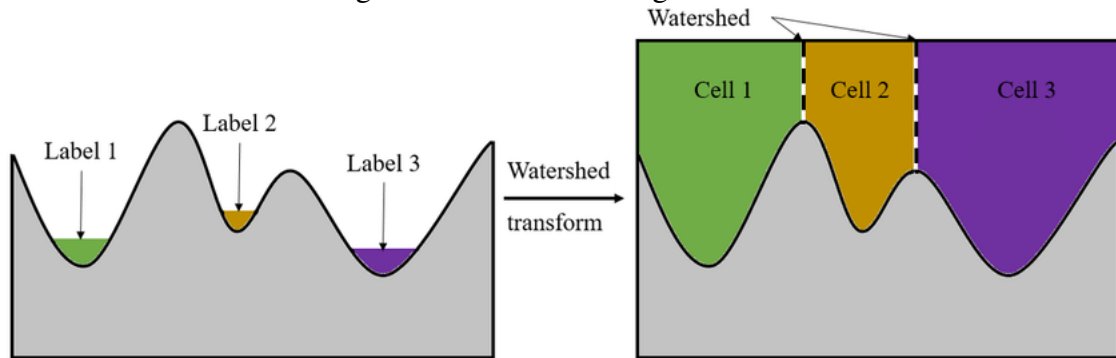
Source: (NOROUZI et al., 2014)

A more complex approach is the watershed algorithm, which is a well-known technique in the field of computer vision. The general concept relates to the idea of a geological watershed, in which the image is treated as a topological map and the bright pixels are treated as mountains and the dark pixels are treated as valleys; the segmentation



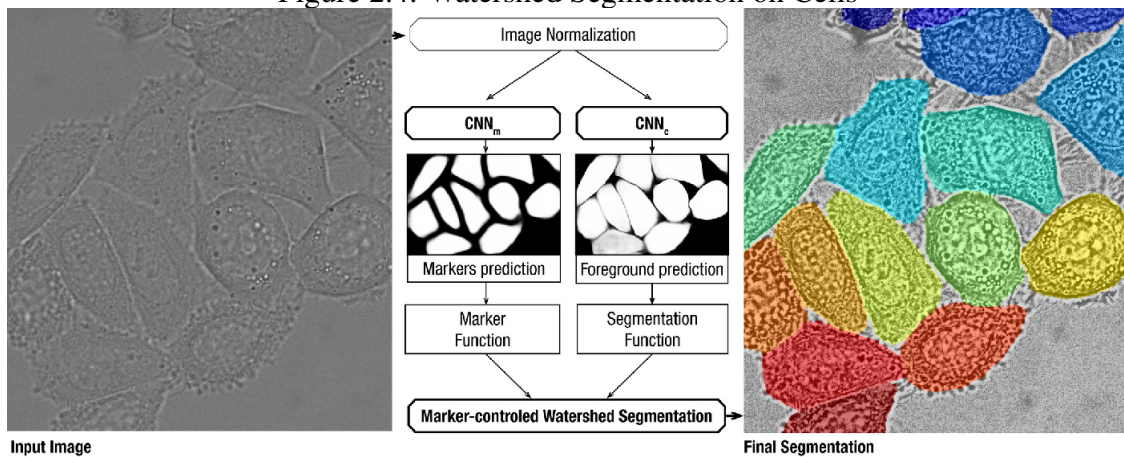
proceeds akin to water filling in the spaces and stopping on the borders of the objects in the images. For this algorithm to function, it is necessary to determine markers for each object present in the image, including the background, as these will be the starting points for the water to fill in the image.

Figure 2.3: Watershed Segmentation



Source: (ZHENG et al., 2021)

Figure 2.4: Watershed Segmentation on Cells



Source: (LUX; MATULA, 2020)

## 2.3 Deep Learning Approaches

When talking about machine learning solutions to image segmentation, a common approach is to use of encoder-decoder structures based on convolutional neural networks (CNNs), which are neural networks with convolutional layers. As defined by O'Shea and Nash (2015), the layer's parameters focus on the use of matrices called learnable kernels. When the data hits a convolutional layer, it convolves each filter across volumetric data, which entails the spatial dimensionality of the input and the different color channels, to produce a 2D activation map.

Some of the initial approaches using convolutional neural networks achieved promising results, most notably the architecture proposed by Simonyan and Zisserman (2014) for image classification, which had a large network with millions of parameters on the ImageNet dataset with 1 million training images. Another approach proposed by Cireřan et al. (2012) computed the class label of a pixel using as input the image intensities in a square window centered on the pixel itself. Both networks produced very good results at the time, but each solution had drawbacks, being the need for a large training dataset or a very slow training speed.

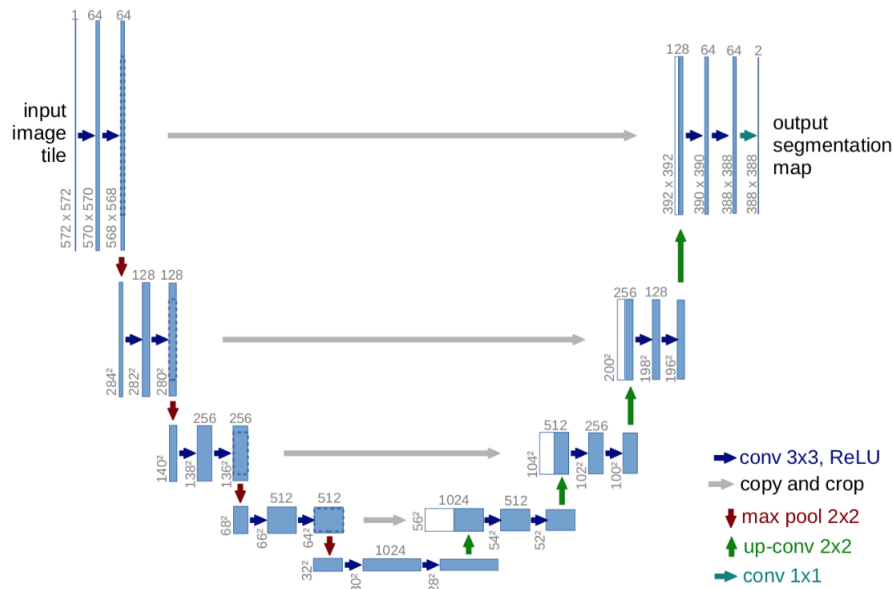
One of the most influential networks for deep learning in biomedical applications is U-Net by Ronneberger, Fischer and Brox (2015). It has great performance, regarding segmentation accuracy, and mitigates the previously stated weaknesses, as it can work with relatively small datasets and has a swift training speed, not to mention having more precise predictions. Its architecture is based on a fully convolutional network. A fully convolutional network (FCN), according to Minaee et al. (2020), is a network that has only convolutional or pooling layers, without the fully connected layers common in other types of networks. This enables it to take an image of arbitrary size and produce a segmentation map of the same size as opposed to a classification score. The main characteristic of this architecture is its U-shape, as seen in Figure 2.5.

The network consists of a contracting path and an expansive one. An input sample passes first through the contracting path, which consists of sequential applications of two  $3 \times 3$  convolutions, rectified linear unit (ReLU) and  $2 \times 2$  max pooling operations with stride 2. This repetition causes downsampling of the image for each step of the sequence, in which the number of feature channels is doubled. As for the expansive path, the sequence that repeats is upsampling of feature map,  $2 \times 2$  convolution that halves the feature channels, concatenation of the cropped feature map from the same level on the contracting side, two  $3 \times 3$  convolutions ending with a ReLU. The last layer, in which the output is generated has a  $1 \times 1$  convolution to map the feature map into the desired number of class channels. The network trains using both input images along with their segmentation maps, using the stochastic gradient descent implementation by (JIA et al., 2014). The energy function, defined by Ronneberger, Fischer and Brox (2015), is computed by a pixel-wise softmax over the final feature map. To compute the loss function for the network, cross entropy was used.

To ensure that the network can learn in contexts with small sample sizes for training data, it is important to implement data augmentation to increase the amount of samples

available. In order to improve the network training, augmentations regarding shift and rotation invariance, as well as deformation robustness and gray value variations, and notably elastic deformations are key to improve performance with very few training samples.

Figure 2.5: U-Net Architecture



Source: (RONNEBERGER; FISCHER; BROX, 2015)

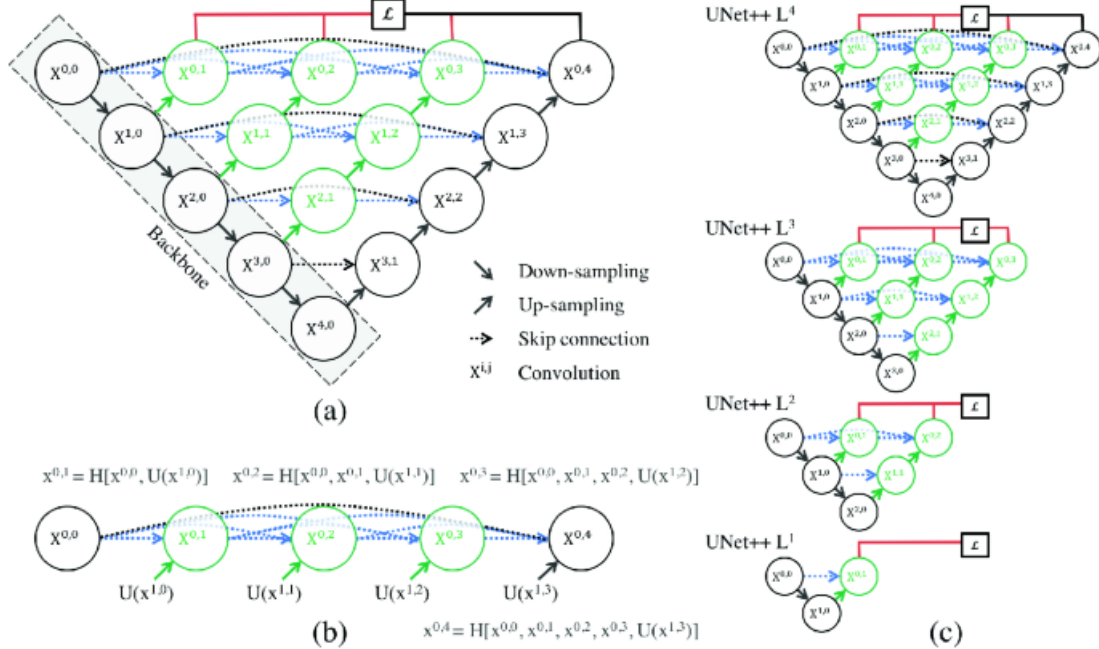
The U-Net network can be considered a pioneer in the field, as this work is a well known name in the field of semantic segmentation. Due to this renown, many different papers have expanded and branched variations of this architecture, improving various aspects of the network.

A direct successor to U-Net is the Unet++ by Zhou et al. (2018). This model is an encoder-decoder architecture that improves on the original U-Net implementation by connecting the encoder and decoder through a series of nested dense convolutional blocks. The purpose of these additional connections is to bridge the semantic gap of the feature maps in the encoder and decoder prior to fusion, this can be seen in Figure 2.6. The network we will base our work on can be considered a second generation successor to U-Net, as it is based on the UNet++ architecture, it is the EfficientUNet++ network, we describe the model in detail in Chapter 3.

## 2.4 Data Augmentation

Throughout the years, many techniques have been developed to improve machine learning approaches for semantic segmentation. A prominent technique used in most

Figure 2.6: UNet++ Architecture, black indicates the original U-Net, green and blue show dense convolution blocks on the skip pathways, and red indicates deep supervision.



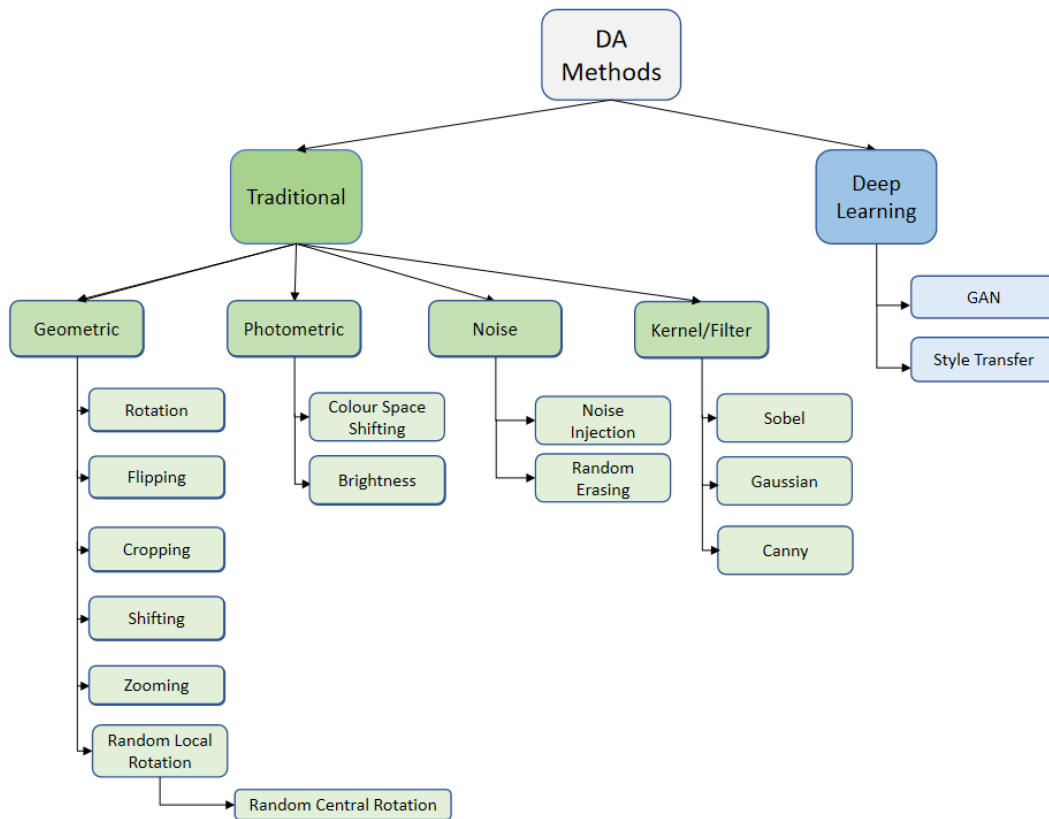
Source: (ZHOU et al., 2018)

datasets that are not significantly large or that lack image diversity is data (image) augmentation. These augmentations can be classified by grouping them according to the type of transformation they apply, as seen in Figure 2.7.

In the beginning, the choice of which data augmentations to apply to a dataset was made by analyzing and picking the transformations that yielded better results, or simply selecting commonly used augmentations. However, selecting the set of augmentation primitives, their magnitudes and combination possibilities is not a trivial task. There are two approaches to optimizing the selection of augmentation policies to use: A brute force approach which relies on manually selecting and testing which data augmentations provide better results, and another approach is based on relying on deep learning algorithms to learn which data augmentations perform better by training on dataset.

One approach in particular which has remarkably good performance and does not rely on Deep Learning techniques is the RandAugment by (CUBUK et al., 2019). Typically, for each data augmentation that is chosen to be applied to a dataset, there are parameters that can be configured to fine-tune the transformation, producing subtle or more prominent changes to an image. Besides configuring these values, it is possible to set the probability of applying said augmentation to the image. Hence, it is possible for an image to have a varying amount of transformations done to it while training a model. RandAugment considers a subset of 14 different transformations to apply, as

Figure 2.7: Different types of data augmentation



Source: (ALOMAR; AYSEL; CAI, 2023)

can be seen in Figure 2.8. This procedure will always apply  $N$  transformations chosen at random, from the available possibilities in the list. At last, the augmentations selected will then be applied to an image according to a magnitude parameter, which will determine the intensity of all the transformations to be performed. This implementation performs exceptionally well considering the simplicity of its concept and that the transformations that are applied are simple geometric and photometric transformations. It is comparable to recent Deep Learning approaches like AutoAugment by Cubuk et al. (2018).

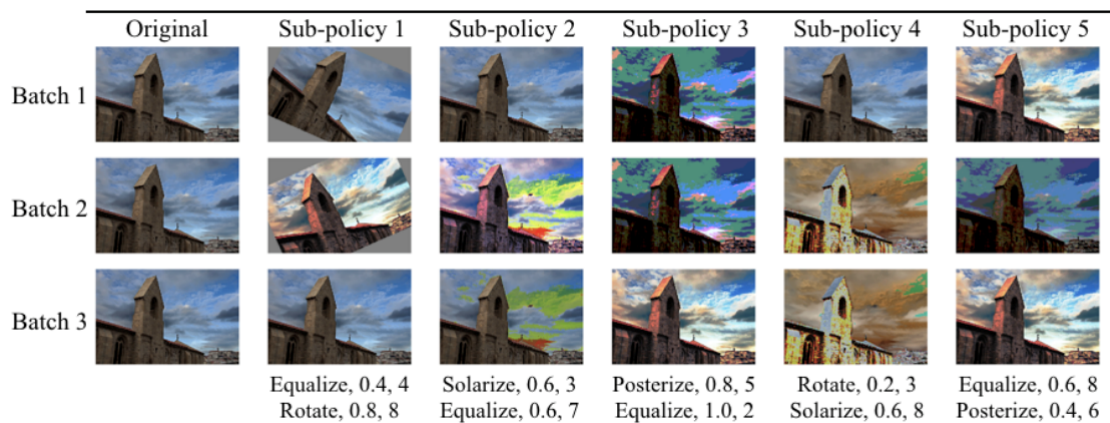
Figure 2.8: Transformations considered by RandAugment

- identity
- autoContrast
- rotate
- solarize
- color
- posterize
- contrast
- brightness
- sharpness
- shear-x
- shear-y
- translate-x
- translate-y

Recently, other approaches that have been gaining traction use Deep Learning to determine the best data augmentation policies to apply to a specific dataset, like AutoAugment by Cubuk et al. (2018), or more recent works like Population Based Augmentation by (HO et al., 2019) and FasterAutoAugment by (HATAYA et al., 2019).

The Deep Learning approaches are based on data augmentation policies learned automatically. The initial work that started this field was AutoAugment by Cubuk et al. (2018), which casts the problem of finding the best augmentation policy as a discrete search problem. This work is divided into a search algorithm and a search space. The algorithm trains a neural network with a sampled augmentation policy, which contains the transformation to be performed, its probability of being applied, and the magnitude of the operation. The validation accuracy of the network is used as feedback for fine-tuning the algorithm. The search space contains the policies, and for each policy there is a group of five pairs of image operations to be applied, with their respective probabilities and magnitudes as parameters. Examples of augmentation policies using this strategy can be seen in Figure 2.9.

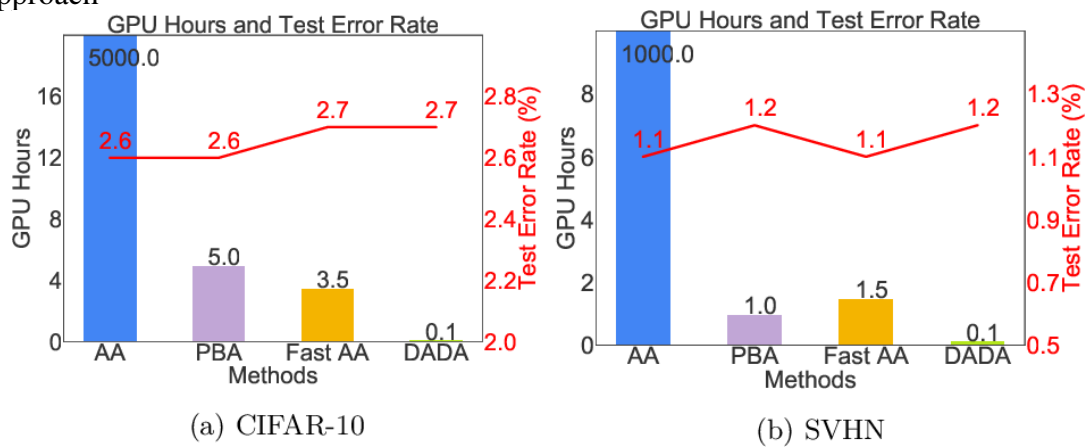
Figure 2.9: Example of policy in search space



Source: (CUBUK et al., 2018)

AutoAugment displays results that compare favorably in image classification through reinforcement learning, but it is not devoid of problems. The optimization task is computationally expensive, using 5000 GPU hours for one augmentation search according to Li et al. (2020). Since not a lot of people have access to these kinds of computational resources or time, many advancements have been made in different works to improve these costs. Population-Based Augmentation tackled this problem through the use of optimization based on population. Meanwhile, Fast AutoAugment by Lim et al. (2019), an approach that is pointedly an upgrade to AutoAugment, used bayesian optimization for solving the augmentation search as a density matching problem. As observed by Li et al. (2020) though PBA and Fast AA greatly improve search speed, augmentation policy learning remains rather slow, e.g. PBA still needs 5 GPU hours for one search on the reduced CIFAR-10 dataset. On Figure 2.10 below the performance of the optimization problems can be observed.

Figure 2.10: Comparison of GPU times and Test Error on different datasets for each approach



Source: (LI et al., 2020)

### 3 METHODOLOGY

In order to evaluate the impact of domain shift/overfitting in regard to cell segmentation, we will implement a series of data augmentation techniques aiming to mitigate their effects. Taking into account these techniques, we can then analyze the performance of a model trained with this generalized approach through augmentation and without it on the various cell samples inside CTC dataset in an intra- and inter-dataset validation comparison.

The model chosen for this work is the EfficientUNet++ Silva et al. (2021). It was selected due to being relatively recent, an improvement on the UNet++ Zhou et al. (2018) network, which is itself a direct successor of U-Net Ronneberger, Fischer and Brox (2015) architecture and having comparable results with these models. The efficiency noted in the model’s name is also an important characteristic to consider for this work, which has a small scope of focus and resources.

#### 3.1 Model

The EfficientUNet++ is a fully convolutional neural network for ordinary and medical image semantic segmentation. The network is composed of an encoder and a decoder. The encoder extracts features of different spatial resolutions, which are fed to the decoder through skip connections, being selected from an existing encoder architecture. The decoder combines its own feature maps with the ones from skip connections to produce accurate segmentation masks. The EfficientUNet++ decoder architecture is based on the UNet++, a model composed of nested U-Net-like decoder sub-networks as can be seen in Figure 3.1.

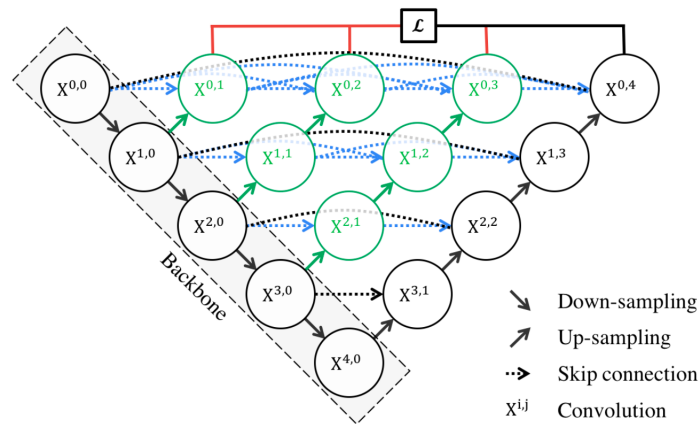
As defined by Silva et al. (2021), the EfficientUNet++ replaces the UNet++’s blocks with residual bottlenecks with depthwise convolutions to increase performance and computational efficiency, which have feature maps that are processed with concurrent spatial and channel squeeze and excitation (scSE) blocks Hu et al. (2017) to enhance performance., this can be seen in Figure 3.2. Each convolution is followed by batch normalization Ioffe and Szegedy (2015) and Hardswish Howard et al. (2019), except for the last  $1 \times 1$  convolution, which is not activated.  $C$  and  $C'$  are the numbers of input and output channels. Feature map height and width are not altered. The bottleneck ratio,  $b$ , is set to 1, and the number of convolution groups,  $g$ , is equal to the number of input channels,



making the  $3 \times 3$  convolution depthwise. The scSE block uses a squeeze ratio of 1.

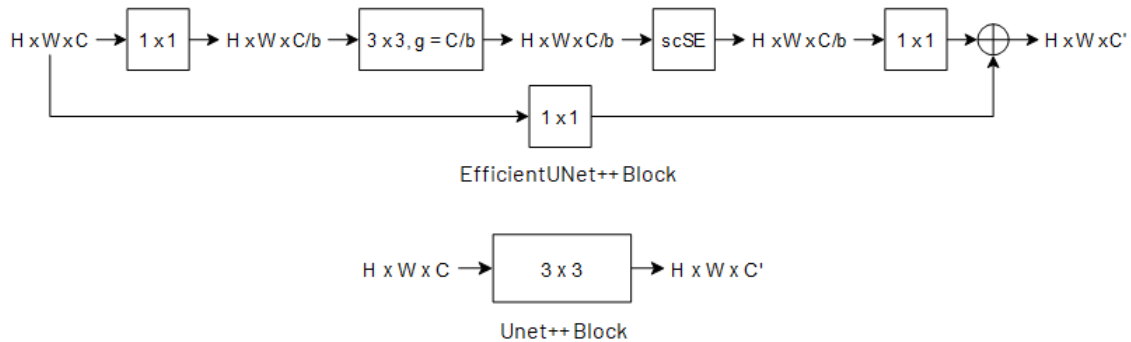
The network synergizes well with EfficientNet encoders proposed by Tan and Le (2019). Due to their efficient visual representations (i.e., using few channels to represent extracted features), EfficientNet encoders require few computations from the decoder, they implement a compound scaling method, which use a compound coefficient  $\phi$  to uniformly scale network width, depth, and resolution in a principled way. There are a selection of different EfficientNet encoders available, with an increasing amount of parameters, from the lowest B0 with 5M parameters up to the highest B7 with 66M. In this work, we use the EfficientNet-B0 encoder for training, as the proposed problem does not require a higher scale.

Figure 3.1: The EfficientUNet++ decoder is based on Unet++ decoder implementation (The components outside of the backbone, which is the encoder).



Source: (ZHOU et al., 2018)

Figure 3.2: Comparison of EfficientUNet++ Block vs Unet++ Block.




Source: (SILVA et al., 2021)

### 3.2 Metrics

To see how accurate a model is for semantic segmentation both when training a model and during its evaluation, it is very important to choose an appropriate way of measuring the results. There are many different ways to measure the “quality” of a model, and the simplest is through pixel accuracy: the percentage of pixels in your image that are classified correctly. Although this metric is valid, it would evaluate poorly for our task, because high pixel accuracy does not correlate to superior semantic segmentation. This is due to class imbalance, which happens when an image has a disproportionate amount of pixels to the classes. Considering that we have images with different cell sizes, it is important to take this into account. The measure we chose is Intersection-Over-Union (IoU), also known as Jaccard Index, this is a very straightforward metric and commonly used for semantic segmentation and it measures the overlap between the predicted segmentation and the ground truth mask divided by the union of both.

Figure 3.3: IoU Calculation visualized.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Source: Google Images

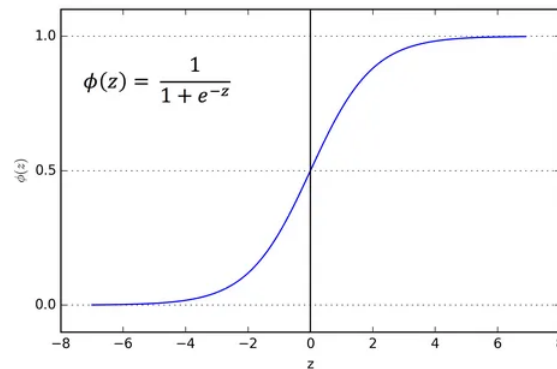
Focusing on the objective of applying semantic segmentation to biomedical imagery, the loss function selected for this work is the Dice Loss. It is calculated from the Dice similarity coefficient (DSC), which measures the similarity of two samples – in this case, a predicted segmentation and a ground truth. Considering that DSC can range from 0-1 (0-100%) in terms of similarity, the Dice Loss is the difference between the samples, below on Figures 3.1 and 3.2 are displayed the formulas for DSC and Dice Loss.

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|} \quad (3.1)$$

$$d = 1 - \frac{2|X \cap Y|}{|X| + |Y|} \quad (3.2)$$

To determine the output of semantic segmentation in a neural network, it is necessary to use an activation function, which derives output from a set of input values to a node. There are several different activation functions that can be used for the semantic segmentation task, but given the scope of our problem, classifying a pixel between background and cell, the most suitable metric for activation is the sigmoid function. Defined in Figure 3.4 below, the sigmoid function has two outputs, 0 or 1, which matches our purposes exactly.

Figure 3.4: Sigmoid function visualized



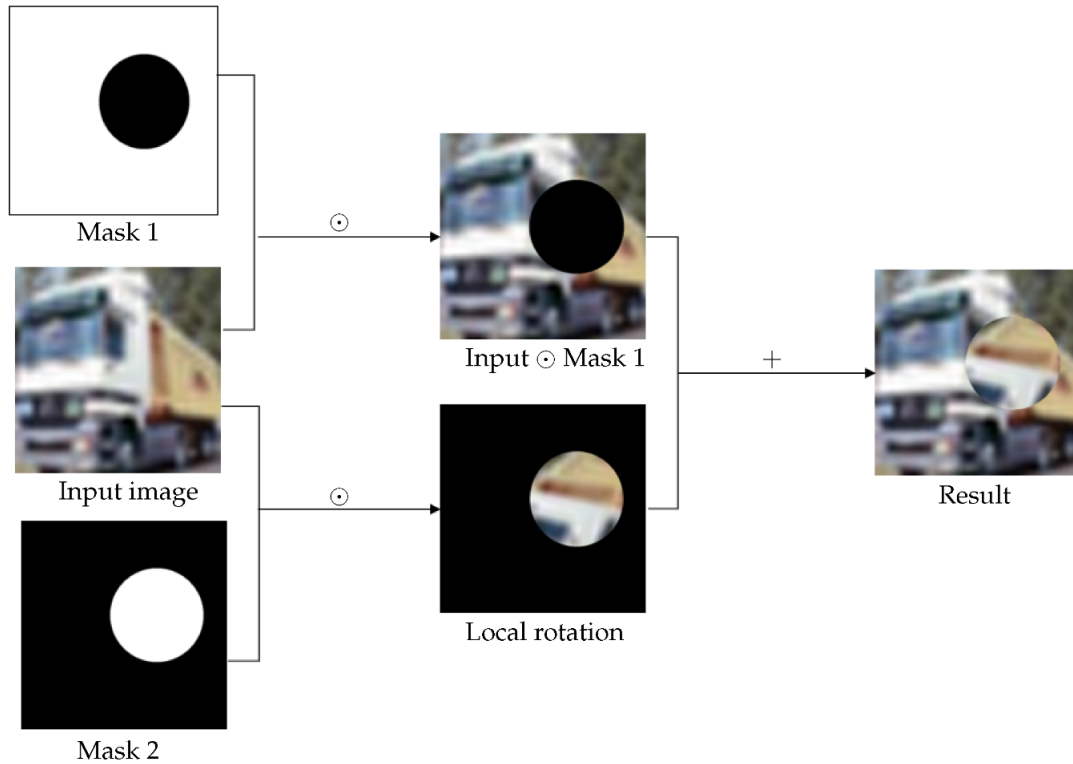
Source: Google Images

### 3.3 Data Augmentation Techniques

For this work, a few recently developed data augmentation techniques that appeared promising in regard to semantic segmentation on biomedical images were selected according to their simplicity and the positive results observed from a few different works regarding cell segmentation.

One particular work which analyzed many different approaches to semantic segmentation and proposed its own approach is Alomar, Aysel and Cai (2023). Their work proposes the Random Local Rotation (RLR) augmentation, which we adopt as another technique for training the model. The RLR is an augmentation technique that, given an input sample, randomly selects a circular area inside the sample and rotates it by a random number of degrees. This can be better exemplified in Figure 3.5 below.

Figure 3.5: The random local rotation data augmentation strategy. Operator  $\odot$  represents pointwise multiplication.



Source: (ALOMAR; AYSEL; CAI, 2023)

Aiming to mitigate domain shift problems when training on one set of data and testing on another, we adopt the CutMix augmentation by Yun et al. (2019). It is a transformation inspired by regional dropout strategies, which guide a model to consider less discriminative features of an object for classification, which in turn improves network generalization and object localization capabilities. Unlike regional dropout, which remove relevant pixel information from the image, thus leading to a loss in the image and inefficiency when training, CutMix solves this problem, instead of simply removing pixels, it replaces the missing pixels with a patch from another image as observed in Figure 3.6. The segmentation masks are also mixed in the same manner as the images, thus guaranteeing both image and mask have no uninformative pixels. This also improves localization by making the model work on identifying objects which can be partially hidden from view.

TrivialAugment by Müller and Hutter (2021) is an approach that is similar to RandAugment Cubuk et al. (2019). It works on the following principle: when an input image is received, it randomly samples an augmentation from a set list of augmentations and applies the transformation to the image with a magnitude that is sampled at random from a set of possible magnitudes which ranges from  $\{0, \dots, 30\}$ , finally returning the augmented

Figure 3.6: Comparison of similar augmentations with CutMix.



Source: (YUN et al., 2019)

image. The main difference from RA lies in the fact that TA only applies one augmentation per image with a randomly selected magnitude given a set of possible magnitudes for each sample. In contrast, RA applies  $N$  augmentations with the same selected magnitude for all samples. In our work, we use a modified TA implementation, allowing multiple augmentations per image like RA, but keeping the random magnitude like in TA. We utilize a slightly modified augmentation space, i.e., the set of augmentations, as seen in Table 3.1.

Table 3.1: List of Augmentations in our work

<i>Augmentation</i>	<i>Effect</i>
NoOp	Identity transformation, does not change sample.
Equalize	Equalizes the image histogram.
Solarize	Invert all pixel values above a threshold.
ColorJitter	Randomly changes the brightness, contrast, and saturation of an image.
RandomBrightnessContrast	Randomly changes brightness and contrast of the input image.
Sharpen	Sharpens the input image and overlays the result with the original image.
RandomGridShuffle	Divides the image by a grid into cells, then randomly shuffle the grid cells on image.
InvertImg	Invert the input image by subtracting pixel values from 255.
ElasticTransform	Elastic deformation of images as described by Simard, Steinkraus and Platt (2003)
GaussNoise	Applies gaussian noise to the input image.
RLR	Applies Random Local Rotation augmentation previously described.
CutMix	Applies CutMix augmentation previously described

Source: (BUSLAEV et al., 2020)

## 4 EXPERIMENTAL RESULTS

In this chapter, we present the setup for training our proposed generalized model for cell segmentation. We also analyze the results against a model without our proposed strategies. We describe the initial data preparation alongside the parameters used in the model, as well as how the dataset will be used in training. It is important to note that our environment is established in `Google Colab`, and all of our training and testing is done with this environment in mind as there are hardware limitations to consider regarding GPU, memory size, and storage limits.

### 4.1 Dataset Preparation

As previously seen in Figure 2.1, the CTC dataset is separated into various different cell types, each one containing many images along with their ground truth masks, captured in a time-lapse sequence. There are different types of reference segmentation masks available in this dataset, as defined by Ulman et al. (2017)

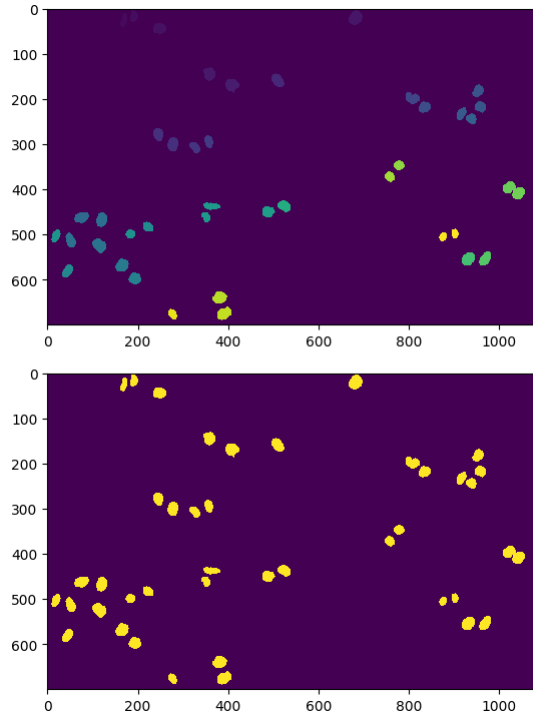
- Absolute truth corpus (ground truth) containing exact reference annotations, available only for simulated datasets;
- Gold-standard corpus (gold truth) containing human-made reference annotations, obtained as a consensual or majority opinion of several human experts;
- Silver-standard corpus (silver truth) containing computer-generated reference annotations, obtained as the majority opinion over the results of several automatic analysis methods submitted by former challenge participants.

In this work, we will use only silver truth masks and when available ground truth masks, for simulated data. This is due to there not being gold truth masks for all images in a given cell type, meanwhile, the other kinds of masks are available for all images and thus it is preferable to use them when training.

Due to the nature of our environment and the way the CTC dataset is organized, it is necessary to establish some limitations on our data. Firstly, all images and segmentation masks need to be resized to  $256 \times 256$  to be a valid input for our model. The CTC challenge broaches more than just semantic segmentation: it also entails object tracking, thus the segmentation mask is originally an instance segmentation mask, meaning each cell has a unique pixel value, allowing distinguishing each individual cell instance. Considering

our objective, we can simplify the segmentation masks by applying a simple threshold operation to them, leaving only two distinct pixel values, background and cells, as seen in Figure 4.1.

Figure 4.1: Fluo-N2DL-HeLa cell type mask before and after threshold.



Source: The Authors

Each cell type in the CTC dataset has a different amount of samples to train on, and some have very few compared to others as seen in Table 4.1, which can lead to training imbalance. To avoid this problem, we always sample the lowest amount of images available, which is from Fluo-C2DL-MSD with 96 samples, regardless of cell type, thus ensuring that sample size does not skew our model’s training.

Table 4.1: Samples per cell type

<i>Cell type</i>	<i>Amount of images</i>
BF-C2DL-HSC	3528
BF-C2DL-MuSC	2752
DIC-C2DH-HeLa	168
Fluo-C2DL-MSD	96
Fluo-N2DH-GOWT1	184
Fluo-N2DH-SIM+	215
Fluo-N2DL-HeLa	184
PhC-C2DH-U373	230
PhC-C2DL-PSC	600



## 4.2 Model configuration

We use two models to run our experiments and compare results. The first is our baseline model, which runs without any of our proposed data augmentation techniques. The second model trains with our selection of data augmentation techniques. Both models are EfficientUNet++ networks and they have the same parameters, seen in Table 4.2. This ensures that the primary difference in performance lies in our proposed techniques, which facilitates comparison. To avoid overfitting, we implement early stopping, a technique that stops the model training when the validation loss does not decrease through several iterations – the number of iterations until the training stops is controlled by the patience parameter.

Table 4.2: List of parameters for our models

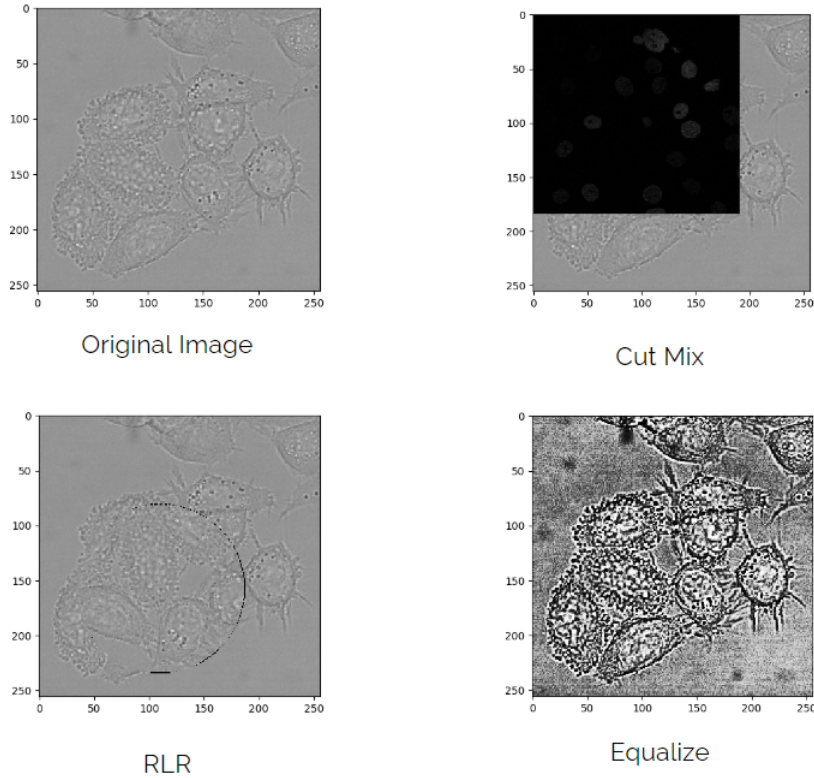
<i>Parameter</i>	<i>Value</i>
Optimizer	Adam
Train Validation Split	Training: 80%, Validation: 20%
Batch Size	Training: 8, Validation: 1
Learning Rate	$10^{-3}$
Patience	10
Encoder Weights	None or Imagenet (DENG et al., 2009)
Max Epochs	150 or Early Stop

## 4.3 Data Augmentation

Due to the relatively small size of our dataset, it is important to work with data augmentation to give the model more samples to learn from. On each epoch during training, the model receives the images to learn, on this step of the process when retrieving the image and segmentation mask, we use our modified TA function to randomly select an augmentation from the available list, described in Table 3.1, and apply it to the image or in cases like geometric augmentations, to both. Illustrated in Figure 4.2 is an example of augmentations applied to a sample image from DIC-C2DH-HeLa.

In order to understand which of the augmentations from the set that we use provides a greater impact on the evaluation performance of our model, we perform our experiments with smaller subsets of our collection of augmentations, along with the full set as well, thus allowing us to compare the performance of augmentations. In Table 4.3 we describe the subsets we use.

Figure 4.2: Example of augmentations applied.



Source: The Authors

#### 4.4 Proposed Training and Evaluation

For our initial experiments, we first evaluate how the baseline model performs when training on one cell type and evaluating on a completely different type, this is done with no initial pre trained encoder weights, which means we use random weight initialization.

As can be observed in Table 4.4, the results of the baseline model evaluate poorly on other cell types compared to the one it trained on. This behavior is an obvious example of the domain shift problem. Comparing the results with a model trained using all the data augmentations discussed in Table 3.1 and using ImageNet pre-trained weights for its encoder, which are then re-trained, as opposed to freezing the weights, the new model performs much better and the use of data augmentation improves the results. We can observe that not all cell types are improved by the same amount, which can be attributed to the similarity between images in some cell types and limitations on the effectiveness of CutMix augmentation, due to it sampling random image crops from all cell types, there are cases where the resulting augmented image may not provide relevant information. This can be mitigated by narrowing the cell types CutMix samples from. For the following

experiments, we initialize encoder weights with ImageNet pre-trained weights, which improves model learning and reduces training time.

Considering the results so far, we can explore the impact of the proposed augmentation subsets in Table 4.3 by training on four distinct scenarios, which can give us a better understanding of the influence of the cell type the model is trained on along with the augmentations applied. These scenarios ensure we can observe the effects of small and large domain shifts, along with the amount of training samples. All scenarios are trained with pre-trained weights of ImageNet and further re-trained. The training scenarios are the following:

- Scenario 1: Train on a cell type that is distinct both in its morphology and in the environment that it was captured in compared to the cell type that we evaluate. For this scenario, we choose the DIC-C2DH-HeLa samples, and we evaluate against all cell types. The cells are very big compared to others and the background is distinct (Observe the difference in Figure 2.1).
- Scenario 2: Train on a cell type that is similar both in morphology and in the environment that it was captured to the cell type we evaluate. For this scenario, we choose to train on the Fluo-N2DH-SIM+. The cells have a similar size and background to the other Fluorescent samples.
- Scenario 3: Train on all cell types but the one we evaluate on, use a distinct cell type like in Scenario 1. In this scenario we evaluate on DIC-C2DH-HeLa.
- Scenario 4: Train on all cell types but the one we evaluate on, use a similar cell type like in Scenario 2. In this scenario we evaluate Fluo-N2DH-SIM+.

Table 4.5 shows the results for scenario 1. We can see mixed results between Baseline and Set 1, some cell types like Fluo-N2DH-GOWT1 improved by 37% while others like BF-C2DL-HSC deteriorated by 24%. Sets 2 and 3 did not show any significant results in comparison, which shows that focusing on a particular kind of data augmentation, be it photometric or geometric, is not as effective as using a broader mixture of augmentations like Set 1. To affirm that this is due to choosing a cell type that is distinct from the others, we need to observe the results of 4.6. Notably, Set 4 of augmentations behaves as expected, the cell type that the CutMix augmentation focuses on has an improved IoU Score, at the expense of the others. It is also possible to note the influence of the quality of the captured images due to microscopy modality (quality of capture from best to worst DIC > BF > PhC > Fluo) and resolution quality. Cell types with high (H) resolution have

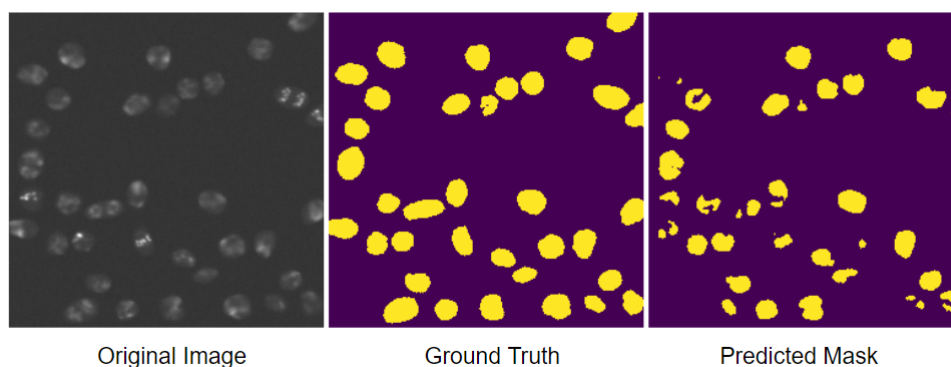
better performance overall and on baseline model DIC has better compatibility with BF and PhC capture modalities, which have similar qualities rather than Fluo.

The results of scenario 2, seen in Table 4.6, show very different results from Table 4.5, which conforms with our hypothesis that cell type distinctiveness influences how effective augmentations will be. We can observe that, in this scenario, each test case had notable results. Set 2, which contains Photometric augmentations, displayed improvements in cell types with differences in visual aspects like color and brightness. On the other hand, Set 3, which contains Geometric and Kernel augmentations, showed improvements in cell types with similar sizes but different shapes to the one we trained on as observed in PhC-C2DL-PSC, which improved by 15% compared to Baseline. We can also observe that Fluo performed better on other Fluo cell types and PhC due to being closer in image capture quality rather than BF and DIC.

Table 4.7 shows the results for scenario 3, in which we compare the Baseline, and augmentation Sets 1 and 5. We can observe that the same improvement that Set 1 displayed in Table 4.5 is not present this time. This shows that the broader the number of samples available during training, the smaller the impact of generalized augmentation when considering a specific cell type. This time, Set 5 results in a 42% improvement, without degrading the overall performance of the model on the other cell types by too much, only around 1%.

The final scenario, summarized in Table 4.8, shows that when evaluating on a cell type that has similarities to the ones present in the training process there is an improvement in both the cell type evaluated and the other cell types.

Figure 4.3: Example of prediction on Fluo-N2DH-SIM+ of Scenario 4.



Source: The Authors

Table 4.3: Augmentation subsets for training

<i>Name of augmentation subset</i>	<i>Augmentations</i>	<i>Description</i>
Set 1: All Aug + General CutMix	NoOp, Equalize, Solarize, ColorJitter, RandomBrightness-Contrast, Sharpen, RandomGridShuffle, InvertImg, Elastic-Transform, Gauss-Noise, RLR, CutMix	Applies all augmentations, CutMix can sample all cell types
Set 2: Photometric Aug	NoOp, Equalize, Solarize, ColorJitter, RandomBrightness-Contrast, InvertImg	Contains Photometric transformations, changes to color channel/pixel value
Set 3: Geometric/Kernel Aug	NoOp, Sharpen, Gauss-Noise, RandomGridShuffle, Elastic Transform, RLR	Contains Geometric and Kernel transformations
Set 4: All Aug + Specific CutMix	NoOp, Equalize, Solarize, ColorJitter, RandomBrightness-Contrast, Sharpen, RandomGridShuffle, InvertImg, Elastic-Transform, Gauss-Noise, RLR, CutMix	Applies all augmentations, CutMix can sample from one specific cell type
Set 5: Specific CutMix	CutMix	Apply sampling from the cell type that will be evaluated

Table 4.4: Initial training of baseline model vs All Augmentations on DIC-C2DH-HeLa cell type

<i>Cell type</i>	<i>BaseLine</i>	<i>All Augmentations</i>
IoU Score %		
BF-C2DL-HSC	17.99	<b>20.95</b>
BF-C2DL-MuSC	6.82	<b>11.4</b>
DIC-C2DH-HeLa	84.87	<b>92.82</b>
Fluo-C2DL-MSD	10.6	<b>16.98</b>
Fluo-N2DH-GOWT1	11.83	<b>64.42</b>
Fluo-N2DH-SIM+	7.53	<b>52.29</b>
Fluo-N2DL-HeLa	0.001	0.001
PhC-C2DH-U373	20.58	<b>27.39</b>
PhC-C2DL-PSC	9.47	<b>11.53</b>

Table 4.5: Scenario 1: Train on DIC-C2DH-HeLa  
*Cell type*

<i>Cell type</i>	<i>Baseline</i>	<i>Set 1</i>	<i>Set 2</i>	<i>Set 3</i>	<i>Set 4</i>
IoU Score %					
BF-C2DL-HSC	<b>45.86</b>	20.95	24.23	44.81	38.10
BF-C2DL-MuSC	10.16	<b>11.40</b>	5.44	3.33	7.92
<b>DIC-C2DH-HeLa</b>	92.19	<b>92.82</b>	92.78	91.89	91.68
<b>Fluo-C2DL-MS</b>	3.73	16.98	0.84	4.19	<b>28.32</b>
Fluo-N2DH-GOWT1	27.38	<b>64.42</b>	33.81	11.37	43.30
Fluo-N2DH-SIM+	8.74	<b>52.29</b>	4.74	1.16	7.92
Fluo-N2DL-HeLa	<b>20.41</b>	0.001	0.92	0.02	0.001
PhC-C2DH-U373	<b>45.84</b>	27.39	13.00	40.88	32.65
PhC-C2DL-PSC	<b>17.95</b>	11.53	1.97	6.21	4.66

\*Set 4 CutMix samples Fluo-C2DL-MS specifically

Table 4.6: Scenario 2: Train on Fluo-N2DH-SIM+  
*Cell type*

<i>Cell type</i>	<i>Baseline</i>	<i>Set 1</i>	<i>Set 2</i>	<i>Set 3</i>	<i>Set 4*</i>
IoU Score %					
BF-C2DL-HSC	<b>12.97</b>	2.61	3.04	6.13	0.99
BF-C2DL-MuSC	7.92	<b>19.18</b>	11.80	8.44	16.93
DIC-C2DH-HeLa	<b>14.68</b>	0.75	2.43	5.57	0.62
Fluo-C2DL-MS	40.69	<b>42.60</b>	35.21	<b>42.04</b>	40.74
<b>Fluo-N2DH-GOWT1</b>	75.57	72.98	74.04	75.08	<b>81.80</b>
<b>Fluo-N2DH-SIM+</b>	<b>89.59</b>	88.21	89.50	88.08	88.83
Fluo-N2DL-HeLa	4.70	26.52	<b>32.75</b>	1.84	24.45
PhC-C2DH-U373	28.64	36.04	<b>36.94</b>	31.20	33.70
PhC-C2DL-PSC	24.89	27.37	24.83	<b>39.75</b>	24.20

\*Set 4 CutMix samples Fluo-N2DH-GOWT1 specifically

Table 4.7: Scenario 3: Train on All Cell Types, Evaluate on DIC-C2DH-HeLa

<i>Cell type</i>	<i>Baseline</i>	<i>Set 1</i>	<i>Set 5*</i>
IoU Score %			
All Cell Types	<b>85.61</b>	83.65	84.73
DIC-C2DH-HeLa	5.57	8.63	<b>47.48</b>

\*Set 5 CutMix samples DIC-C2DH-HeLa specifically

Table 4.8: Scenario 4: Train on All Cell Types, Evaluate on Fluo-N2DH-SIM+

<i>Cell type</i>	<i>Baseline</i>	<i>Set 5*</i>
IoU Score %		
All Cell Types	86.17	<b>87.00</b>
Fluo-N2DH-SIM+	46.29	<b>53.78</b>

\*Set 5 CutMix samples Fluo-N2DH-SIM+ specifically

## 5 CONCLUSION

In this work we have observed the problem of domain shift and overfitting in the context of cell segmentation through extensive experiments. We have also implemented several different mitigation strategies based on modern data augmentation techniques, namely: TA Müller and Hutter (2021), RLR (ALOMAR; AYSEL; CAI, 2023) and CutMix Yun et al. (2019) with varying degrees of success.

In our tests, we used a model based on the EfficientUNet++ (TAN; LE, 2019) architecture with the EfficientNet-B0 encoder. We train two versions of this model on the CTC dataset, which contains several samples of different cell types. The first is a model without using any of our data augmentation techniques, this can be used as a baseline for comparison with the next model which implements the proposed data augmentations. We also tested different subsets of our proposed augmentations in different scenarios to narrow down what is the best approach for each situation.

To analyze the effectiveness of the proposed generalized model with data augmentation, we performed experiments on both a baseline and our approach with an intra- and inter-dataset validation scheme. The results showed that when the training sample is small and there is a big domain shift between datasets, the best approach is to implement all the proposed augmentations to broaden the dataset as much as possible. In this scenario when the domain shift is small it is best to tailor the augmentations to best match the desired dataset domain, i.e, if the cells have different shapes apply geometric augmentations; if the background has a different color apply photometric transformations. We have also observed that when the training dataset is bigger and already has a broad variety of samples, the best approach is to use CutMix directly on the dataset domain that is to be evaluated. It is important to note that there is variability when training and testing models, due to the random nature of training weights and augmentations, but this was not evaluated in this work due to time constraints.

We have briefly touched on a few different works, which had approaches that were applicable to this work. We discuss these implementations along with others that were researched but not mentioned as possibilities for future improvements in this work.

A difficult part of this work was choosing which data augmentation primitives to adopt. Many of the selected transformations came from already established augmentation spaces like in RandAugment (CUBUK et al., 2019). This could be further improved by implementing one of the Deep Learning approaches to data augmentation like Au-

toAugment (CUBUK et al., 2018), which would optimize the choice of augmentations according to our dataset.

A unique approach to data augmentation that was briefly observed in (ALOMAR; AYSEL; CAI, 2023) is the ObjectAug (ZHANG; ZHANG; XU, 2021), which proposes object-level data augmentation for semantic segmentation, which is accomplished by applying other data augmentation techniques only to the objects in the image sample. This process is done by interpreting the segmentation masks to extrapolate what parts of the image pertain to the object to augment. This could be applied to our work, changing a cell's color could make it harder or easier to distinguish from background, geometric transformations could produce more variations in cell shape.



## REFERENCES

- ALOMAR, K.; AYSEL, H. I.; CAI, X. Data Augmentation in Classification and Segmentation: A Survey and New Strategies. **Journal of Imaging**, v. 9, n. 2, 2023. ISSN 2313-433X. Available from Internet: <<https://www.mdpi.com/2313-433X/9/2/46>>.
- BUSLAEV, A. et al. Albuementations: Fast and Flexible Image Augmentations. **Information**, v. 11, n. 2, 2020. ISSN 2078-2489. Available from Internet: <<https://www.mdpi.com/2078-2489/11/2/125>>.
- CIREŞAN, D. C. et al. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In: **Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2**. Red Hook, NY, USA: Curran Associates Inc., 2012. (NIPS'12), p. 2843–2851.
- CUBUK, E. D. et al. AutoAugment: Learning Augmentation Policies from Data. 5 2018. Available from Internet: <<http://arxiv.org/abs/1805.09501>>.
- CUBUK, E. D. et al. RandAugment: Practical automated data augmentation with a reduced search space. 9 2019. Available from Internet: <<http://arxiv.org/abs/1909.13719>>.
- DENG, J. et al. ImageNet: A large-scale hierarchical image database. In: **2009 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2009. p. 248–255.
- HATAYA, R. et al. Faster AutoAugment: Learning Augmentation Strategies using Backpropagation. 11 2019. Available from Internet: <<http://arxiv.org/abs/1911.06987>>.
- HO, D. et al. Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules. In: CHAUDHURI, K.; SALAKHUTDINOV, R. (Ed.). **Proceedings of the 36th International Conference on Machine Learning**. PMLR, 2019. (Proceedings of Machine Learning Research, v. 97), p. 2731–2741. Available from Internet: <<https://proceedings.mlr.press/v97/ho19b.html>>.
- HOWARD, A. et al. Searching for MobileNetV3. 5 2019. Available from Internet: <<http://arxiv.org/abs/1905.02244>>.
- HU, J. et al. Squeeze-and-Excitation Networks. 9 2017. Available from Internet: <<http://arxiv.org/abs/1709.01507>>.
- IOFFE, S.; SZEGEDY, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2 2015. Available from Internet: <<http://arxiv.org/abs/1502.03167>>.
- JIA, Y. et al. Caffe: Convolutional Architecture for Fast Feature Embedding. 6 2014. Available from Internet: <<http://arxiv.org/abs/1408.5093>>.
- LI, Y. et al. DADA: Differentiable Automatic Data Augmentation. 3 2020. Available from Internet: <<https://arxiv.org/abs/2003.03780>>.
- LIM, S. et al. Fast AutoAugment. 5 2019. Available from Internet: <<http://arxiv.org/abs/1905.00397>>.

LUX, F.; MATULA, P. Cell Segmentation by Combining Marker-Controlled Watershed and Deep Learning. 4 2020. Available from Internet: <<http://arxiv.org/abs/2004.01607>>.

MINAEE, S. et al. Image Segmentation Using Deep Learning: A Survey. 1 2020. Available from Internet: <<http://arxiv.org/abs/2001.05566>>.

MÜLLER, S. G.; HUTTER, F. TrivialAugment: Tuning-free Yet State-of-the-Art Data Augmentation. 3 2021. Available from Internet: <<http://arxiv.org/abs/2103.10158>>.

NOROUZI, A. et al. Medical Image Segmentation Methods, Algorithms, and Applications. **IETE Technical Review**, v. 31, p. 199–213, 4 2014.

O'SHEA, K.; NASH, R. An Introduction to Convolutional Neural Networks. 11 2015. Available from Internet: <<https://arxiv.org/abs/1511.08458>>.

OTSU, N. A Threshold Selection Method from Gray-Level Histograms. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 9, n. 1, p. 62–66, 1979.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. 5 2015. Available from Internet: <<http://arxiv.org/abs/1505.04597>>.

SILVA, J. L. et al. Encoder-Decoder Architectures for Clinically Relevant Coronary Artery Segmentation. 6 2021. Available from Internet: <<https://arxiv.org/abs/2106.11447>>.

SIMARD, P. Y.; STEINKRAUS, D.; PLATT, J. C. Best practices for convolutional neural networks applied to visual document analysis. In: **Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings**. [S.l.: s.n.], 2003. p. 958–963.

SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. 9 2014. Available from Internet: <<http://arxiv.org/abs/1409.1556>>.

SUN, B.; FENG, J.; SAENKO, K. Return of Frustratingly Easy Domain Adaptation. 11 2015. Available from Internet: <<https://arxiv.org/abs/1511.05547>>.

TAN, M.; LE, Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. 5 2019. Available from Internet: <<https://arxiv.org/abs/1905.11946>>.

ULMAN, V. et al. An objective comparison of cell-tracking algorithms. **Nature Methods**, v. 14, n. 12, p. 1141–1152, 2017. ISSN 1548-7105. Available from Internet: <<https://doi.org/10.1038/nmeth.4473>>.

YUN, S. et al. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. 5 2019. Available from Internet: <<https://arxiv.org/abs/1905.04899>>.

ZHANG, J.; ZHANG, Y.; XU, X. ObjectAug: Object-level Data Augmentation for Semantic Image Segmentation. 1 2021. Available from Internet: <<http://arxiv.org/abs/2102.00221>>.

ZHENG, T. et al. Research on Distance Transform and Neural Network Lidar Information Sampling Classification-Based Semantic Segmentation of 2D Indoor Room Maps. **Sensors**, v. 21, p. 1365, 3 2021.

ZHOU, B. et al. Semantic understanding of scenes through the ade20k dataset. **arXiv preprint arXiv:1608.05442**, 2016.

ZHOU, B. et al. Scene Parsing through ADE20K Dataset. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2017.

ZHOU, Z. et al. UNet++: A Nested U-Net Architecture for Medical Image Segmentation. 7 2018. Available from Internet: <<http://arxiv.org/abs/1807.10165>>.