ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Dottorato di Ricerca in Data Science and Computation

Supervised and Weakly Supervised Counting-By-Segmentation: The Fluorescent Microscopy use case

Settore Concorsuale:
02/A1 – Fisica Sperimentale delle Interazioni Fondamentali

Settore Scientifico Disciplinare:
FIS/01 Fisica Sperimentale

Coordinatore Dottorato
Prof. Andrea Cavalli

Candidato
Roberto Morelli

Supervisore
Prof. Antonio Zoccoli

Co-supervisore
Prof. Lorenzo Rinaldi

XXXIII Ciclo
Esame Finale Anno 2023

Abstract

This thesis' work is focused on object counting in fluorescent microscopy images. In these pictures, neuronal cell activity is detected to study the mechanisms underlying torpor. To prove the relevance of this biological effect is essential to quantify the number of activated neurons in localized slices of mouse and mice brains represented in the images analyzed in this work. Usually, this amount of cells is provided by human operators through manual annotation. This task is very time-consuming delaying the outcome of the experiments and wasting the researcher's time. Moreover, the fatigue and the subjectivity of the annotators can introduce bias in the count value affecting also the experimental evidence. For these reasons, we investigate a deep-learning-based procedure to automatize this task.

Specifically, we based our work on two of the main convolutional-neural-network (CNNs) state-of-the-art architectures: UNet and ResUnet family model. While different approaches exist to face the counting task, we resort to the counting-by-segmentation strategy to provide also a justification of the objects considered during the counting process. The supervised ground-truth labels used under this framework are binary masks in which the coordinate corresponding to the cell's object's location are represented by white pixels while the surrounding background is black. However, these target masks are time-expensive to produce and often impractical to generate. So, together with the supervised case, we also explore a weakly-supervised learning strategy exploiting only dot annotations. This latter kind of label is represented by a set of coordinates that represent a point (usually near the center) of the objects to detect. We illustrate two viable options to generate the pseudo-labels used for the model training depending on the dataset objects' features.

Sometimes, not only the annotation process can prevent the model training but the same data availability may represent a bottleneck. So, exploiting the two datasets used in this work, we quantify the advantages in terms of data reduction and counting performance boost obtainable with a transfer-learning approach and, specifically, a fine-tuning procedure. With the spirit of open science in mind, we released the dataset used for the supervised use

case and all the pre-training models. Moreover, the last part of the work is dedicated to the design of a web application used to share both the counting process pipeline developed in this work and the models pre-trained on the dataset analyzed in this work.

A tutte le persone "presenti"
e in modo particolare a quelle "future"
Ad Agata

Amico mio! Quello che ti ho già detto tante volte, te lo ripeto, anzi te lo grido: o questo, o quello, aut-aut!

S. Kierkgaard

Chiunque può arrabbiarsi: questo è facile; ma arrabbiarsi con la persona giusta, e nel grado giusto, ed al momento giusto, e per lo scopo giusto, e nel modo giusto: questo non è nelle possibilità di chiunque e non è facile.

Aristotele

Acknowledgments

Ringrazio il mio supervisor Antonio Zoccoli che ha permesso di continuare il mio lavoro di tesi anche dopo le mie scelte lavorative. Ugualmente ringrazio Lorenzo Rinaldi che ha contribuito all'organizzazione dei contenuti della tesi e alla loro accurata revisione. Ricordo con piacere i miei due anni a Bologna e il proficuo lavoro con Luca Clissa oltre che con tutti i ragazzi dell'aula 104. Approdato a Genova ho avuto la fortuna di trovare un'ambiente altrettanto accogliente e stimolante per proseguire il mio lavoro. Ringrazio tutte le persone che almeno una volta in questi quattro anni mi hanno chiesto come procedeva la tesi. Ringrazio i miei genitori e mia sorella, che hanno retto le mie variazioni emotive facendomi desiderare e mirare alla degna conclusione di questo lavoro.

Contents

List of Figures

Chapter 1

Introduction

Deep learning is one of the most fruitful branches of the AI world and provides an appealing set of tools to automatize tasks like the object counting considered in this thesis. Among these algorithms we largely take advantage of state-of-the-art convolutional-neural networks (CNNs) (Jimenez-del Toro et al. [2017], Greenspan et al. [2016]). Specifically, our work strongly relies on the well-known UNet architecture described the first time in the work Ronneberger et al. [2015] to address a segmentation problem on images coming from the biological research domain.

Starting from this architecture we developed a model that quantifies the activity of neuronal cells in microscopy fluorescent images. These pictures are acquired during the investigation of the torpor onset mechanism concerning small-size mammals like mice and rats. The scope is to capture the network of neuron communication that is specifically activated during this particular metabolic state. We address this task following the counting-by-segmentation approach, that is, we first localize the objects during the segmentation step, and then we count them. This approach requires, under a supervised learning frame, the availability of binary ground-truth masks. The first part of this work is focused on the best-case scenario of binary mask availability that largely facilitates the training of our deep learning model. However, ground-truth masks are quite labor-intensive to obtain and require great time and effort. In many cases, we may have a problem getting such labels jeopardizing

the training of a deep learning model. To lose this constraint we explored a strategy to face the same counting-by-segmentation relying only on dot annotations that are a kind of labels significantly easier to obtain. This case is reported as a case of weak supervision and is the bulk of the second part of this work.

To present these results we worked on two distinct datasets that focus on different genes expression involving the torpor onset: the CTb and c-Fos genes. A further step we want to take during this thesis is to understand how to reduce the number of images needed for the model training by exploiting transfer learning techniques and, specifically, a fine-tuning procedure. We quantify this amount by comparing models trained from scratch with models fine-tuned on the target domain varying the number of images and evaluating their detection scores. All these efforts are thought to be a solid step toward the sharing of data and knowledge to foster research in this field. For this reason, we published the supervised dataset (Fluorescent Neuronal Cells) and the pre-trained models (cell counting yellow repository). Moreover, a web application is designed to make available the entire cell counting pipeline tool. Users from all over the world could potentially interact with the pre-trained model via a graphical interface to make inferences on their custom dataset.

Structure of the thesis

After an introduction to the computer vision discipline and the convolutional neural network architectures largely used in this work, we briefly describe the differences between the main learning paradigms, focusing on the supervised, weakly supervised, and transfer learning approaches. Then we present the main topic of the thesis concerning object counting in the fluorescent microscopy domain. The remaining chapters are structured accordingly:

Chapter 2 present the counting-by-segmentation approach under the supervised learning framiework. This chapter essentially reports the work described in Morelli et al. [2021b]. Chapter 3 reports the novel approach developed during this thesis to address the counting-by-segmentation task without

relying on the ground-truth masks but only using dot annotations. We report two different approaches tailored to two distinct datasets with their peculiar characteristics. Chapter 4 is focused on the transfer learning approach used to transfer feature knowledge from a source to a target domain. We extensively compared scratch models with fine-tuned models to quantify the performance boost by varying the number of images available from the target domain. We aim to highlight this method's benefit in the data scarcity context. Finally, Chapter 5 presents a web-app development aimed to share such trained models, fostering research in the biomedical imaging field.

1.1 Computer vision and deep learning

Computer vision is a field of artificial intelligence that focuses on enabling algorithms to interpret and comprehend visual data from the world around them. Typical tasks are, for example, the objects' recognition and identification from images and videos. Deep learning algorithms are even more preferred for digital image processing over standard processing techniques. In the following we describe the fundamentals of digital image processing and the use of deep learning algorithms in the field.

Digital images

A digital image can be defined as a matrix of values in which each element, the *pixel*, represents intensity within a range of discrete values. For example, in the simplest case, a two-dimensional matrix gives the representation of a *grayscale* scene in which the pixel values range from a minimum, black, and a maximum value, white. In a more realistic case, modern digital cameras store three different channels (RGB) representing the frequencies of the electromagnetic spectrum of red, green, and blue colors. The image is then the superposition of these three channels. These two examples do not exhaust the set of all possible digital image types, which depend essentially on the electromagnetic bands stored in the different channels of an image. One of the extreme cases is represented by $\mathbf{hyper-spectral}$ images containing up

to several hundred bands and ranging from $200nm$ up to $2500nm$. Even when the physical process underlying the creation of an image becomes more complex, as in the case of x-ray radiographs, it is always possible to obtain a digital image through discretization into a matrix of values. Digitalization brings the great advantage to process this data automatically through standard image processing methods or machine learning algorithms.

Image processing

Digital images can be represented in two different domains:

- spatial domain;
- frequency domain.

To understand how simple processing works, it is useful first to dwell on spatial representation. As anticipated earlier, representation in the spatial domain involves discretization into a matrix in which pixels represent its intensity point by point. From these matrices, by operating certain operations it is possible to process the image to correct its appearance by removing some noise or to extract certain features of interest. These operations take place using filters that are 2 or 3-dimensional matrix with a typical spatial size of $3x3$ or $5x5$. The mathematical operation underlying these processes is convolution, the analytical form of which is given below:

$$(f * g)[n_1, n_2] = \sum_{m_1=0}^{M_1} \sum_{m_2=0}^{M_2} f[m_1, m_2] \cdot g[n_1 - m_1, n_2 - m_2] \qquad (1.1)$$

Where, f and g are respectively the image and the kernel used in the convolution operation, n_1 and n_2 are the indices of the output image and m_1 and m_2 are the indices of the input image. The result of the convolution is a new image with a different appearance. We can have an image filtered from the noise or with some enhanced details like for example the horizontal or vertical lines. By composing different filters, it is then possible to emphasize

the structure of specific objects composed by such elementary features. Image detection is the result of a concatenation of filters applied to highlight a specific target object. Another relevant task usually performed on digital images is object segmentation. This operation consists of producing a binary image where the pixels belonging to the object to detect are white producing an entirely connected boolean structure. The rest of the image is completely black and represents the background.

Deep learning

Computer vision is one of the fields in which neural networks have seen the most successful applications. In this case, the most widely used architectures are the convolutional networks largely inspired by the workings of the visual cortex. The name convolutional networks come from the first network described in Yann LeCun's LeCun et al. [1995] paper. These architectures operate using convolution operations to extract features (characteristic elements) as described above. These filters, also known as **kernel**, slide pixel by pixel on the input image performing convolution operations. The resulting images provide many different representations of the input image which help the model to accomplish a specific task, like object detection or segmentation for example.

One of the first relevant architectures is $\mathbf{LeNet-5}$ LeCun et al. [2015], opening the way to new and more powerful architectures. The structure of $LeNet-5$ is very simple and includes two convolutional layers followed by two pooling layers. These last layers are used to reduce the resolution of the images to encode a smaller representation of the image and save computational resources. Finally, two fully connected layers process the convolution results and output the images' class prediction.

Initially, these networks were widely used for digit classification on datasets such as **MNIST** that are frequently used to benchmark new competing architectures. Another public dataset used for the same scope is **ImageNet**. This dataset collects different categories of commonly used objects with their relative label, i.e., the response we want the network to give us whenever

it processes that specific image. Starting in 2010, from year to year, new architectures have followed as winners in competitions based on the classification of these images achieving increasing accuracy. For example, **AlexNet** Krizhevsky et al. [2012], is the first CNN to win this competition inaugurating the era of CNNs. However, between the first network described, LeNet-5, and the AlexNet there is a significant increment of the parameters: from 60 thousand to 60 million. This huge number of parameters, proportional to the number of kernels of the network, allows a better description of the data. Such huge models represent usually the backbone adopted to define new architectures. A famous example is **VGGNet** Simonyan and Zisserman [2015] that counts 11 different layers with an increasing number of filters. An excessive number of parameters can cause many problems, from training difficulties to the risk to overfit the dataset. So **GoogLeNet** Szegedy et al. [2015], start to decrease the number of parameters excluding the expensive fully-connected without to be even deeper of its predecessor. This architecture also introduces a novel *inception module* which enables making convolutions operations parallelly. The result is a more computationally efficient network with higher accuracy on ImageNet than its precursors. Another breakthrough work is related to the introduction of the **ResNet** architecture He et al. [2016] that includes an innovative element: the **residual block**. This unit allows the result of a previous convolutional block to be summed with the result of the next one ensuring better convergence due to better gradient propagation through the convolutional layers.

A turning point in the world of computer vision is also marked by the introduction of the **UNet** model Ronneberger et al. [2015], which takes its name from its peculiar structure shown in figure 1.1 . In fact, in contrast to previous models, the convolutional and pooling layers alternate forming two paths: towards the first one the input image is gradually reduced in size and while passing into the second one the processed image recovers its original size. In the middle of the two branches, a **bottleneck** unit represents the point of maximum compression of the image and forces the network to extract only the relevant information from the scene. Such architecture, composed of an encoder (compression path) and a decoder (reconstruction path) also

shows great effectiveness in tasks such as segmentation.

Convolutional Neural Networks (CNNs), have demonstrated their ability to outperform state-of-the-art techniques in a variety of computer vision applications. As a result, researchers in academia and industry have begun to explore the use of CNNs in fields such as medical imaging and bioinformatics, where the potential impact is significant. Successful examples range from classification and detection of basically any kind of objects (Krizhevsky et al. [2012], Redmon et al. [2016]) to generative models for image reconstruction Cheng et al. [2018] and super-resolution Ledig et al. [2017]. CNN's have been used for tasks such as the identification and localization of tumors (Havaei et al. [2017], Vandenberghe et al. [2017], Ciresan et al. [2012, 2013]), as well as detection of other structures like lung nodules (Jiang et al. [2018], Meraj et al. [2020], Su et al. [2021]), skin and breast cancer, diabetic foot Alzubaidi et al. [2021], colon-rectal polyps Korbar et al. [2017] and more, showing great potential in detecting and classifying biological features (Lundervold and Lundervold [2019], Sahiner et al. [1996], Yadav and Jadhav [2019]).

1.2 Learning Paradigm

Data-driven models, like the architectures introduced in the previous section (Section 1.1), are used for a large number of applications. However, each algorithm needs a training phase before resolving a specific computer vision task. There are several learning paradigms in machine learning, including:

- **Supervised learning**: In this paradigm, the model is trained on labeled data, where the correct output is provided for each input. The model then makes predictions on new data based on what it has learned from the training data.

- **Unsupervised learning**: In this paradigm, the model is not provided with labeled data, and must find patterns and relationships in the input data on its own. This can be used to discover hidden structures in the data or to group similar data points.

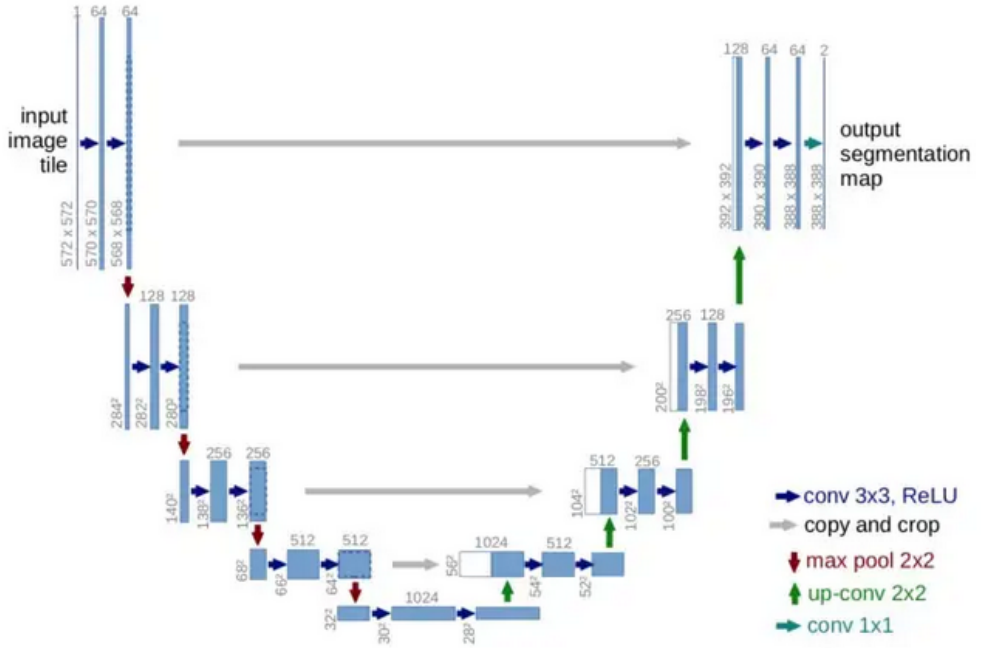


Figure 1.1: **Unet architecture.**. The architecture is composed by, and encoding path, a bottleneck, and a decoding path used to recover the initial input dimension.

- **Reinforcement learning**: In this paradigm, the model learns to take actions in an environment to maximize a reward. This can be used to train agents to perform complex tasks, such as playing video games or controlling robots.
- **Semi-supervised learning**: In this paradigm, the model is trained on a combination of labeled and unlabeled data. This can be useful when there is a lot of data available, but only a small portion of it is labeled.
- **Transfer learning**: In this paradigm, a model that has been trained on one task is used as the starting point for training on a related task. This can save time and resources, and can often improve performance on the new task.

Overall, the choice of learning paradigm depends on the specific problem that the model is being trained to solve, and the availability of labeled and unlabeled data. The list can be longer but most of the application is covered by the first three cases. For the scope of the thesis, in the remaining part of this section, we go deeper into the description of the supervised learning and we introduce the weakly supervised learning which represents respectively the framework of the use cases described in Chapter 2 and Chapter 3. Finally, an introduction to the transfer learning approach is reported to facilitate the introduction to the work presented in the Chapter 4.

Supevised learning

Supervised learning is a type of machine learning where the model is trained on labeled data. This means that the input data is accompanied by the correct output labels, so the model can learn to predict the correct output given a new input. During training, the model tries to learn the relationship between a (set of) response/target variable Y based on a set of predictors/inputs $\boldsymbol{X}$. This relationship can be formalized as follows:

$$Y = f\left(\boldsymbol{X}; \boldsymbol{\theta}\right) + \epsilon \tag{1.2}$$

where $\boldsymbol{\theta}$ is a parameter vector that defines the precise form of the function f. In the case of tabular data, the $\boldsymbol{X}$ is a matrix in which the number of columns is the predictor that defines each sample/ row of the matrix. If the input is images, $\boldsymbol{X}$ is a tensor.

In practice, the model with randomly initialized weights starts by guessing the association between some input data $\boldsymbol{x}$ and the corresponding response. The difference between the predicted output and the corresponding label, y, is quantified using a *loss function*. This is the objective function to minimize thanks to the supervision given from the knowledge of the labels y. Then, from the value of this function, the gradient is computed and backpropagated to update the weights value of the model following the rule:

$$w = w - \alpha * \Delta_w \tag{1.3}$$

where α is the learning rate and is a hyperparameter to set before the training and Δ_w is a vector that contains the weight updates of each weight coefficient. The weights update is computed each time some inputs are provided to the model. Exist different strategies to define the pace of this update. In **batch learning**, the model is trained on the entire dataset at once, and the weights are updated after each epoch (i.e., one pass through the entire dataset). In **online learning**, the model is trained on individual examples one at a time, and the weights are updated after each example. In the middle, there is the **mini batch** training that consists in picking a predefined number of samples before updating the weights. When all the samples are selected a training **epoch** ends and the mini-batches are sampled again following the same scheme as before.

The advantage of this approach is that the previous knowledge provided by the labels is leveraged to supervise the training, helping the model learn the right mapping between predictors and targets. For this reason, the SL paradigm is widely used in practice and has a long list of successes for many different learning tasks (e.g. spam filtering, fraud detection, image classification, and stock price forecasting). However, most of the data in a real-world scenario are produced without labels. This prevents the adoption of SL

techniques to learn from such data unless undertaking a (probably costly) labeling/annotation phase before their analysis.

Weakly supervised learning

The difference between weakly supervised learning and supervised learning concern the kind of labels employed during the training. Instead of requiring fully labeled training examples, weakly supervised learning techniques can make use of partially labeled data or even just large amounts of unlabeled data, combined with some form of weak supervision such as heuristics or other forms of noisy labels. This type of supervision can be especially useful in situations where it is difficult or expensive to obtain fully labeled training data, enabling models to still learn useful patterns and make predictions without the need for a large amount of human annotation. However, it is important to note that the quality of the model's predictions may be lower than those produced by fully supervised learning methods due to the inherent uncertainty in the weak labels.

Weak supervision can also be beneficial in cases where the desired output is not well-defined or where there is significant variability in the data. For example, in natural language processing tasks, it can be difficult to define a precise set of rules for determining the meaning of a sentence. In such cases, weak supervision can be used to extract useful information from the data through the use of heuristics, even if the labels are not completely accurate.

Overall, weak supervision can be a useful tool for training machine learning models when expert-labeled data is scarce or when the task at hand is too complex for manual labeling. For this reason, weakly supervised methods have been explored over the years. Standing to the definition of weakly supervised learning, we can distinguish three typical situations Zhou [2018]:

- **incomplete supervision**: where only a subset of training data is given with labels;
- **inexact supervision**: where the training data are given with only coarse-grained labels;

- **inaccurate supervision**: where the given labels are not always ground-truth.

The use cases discussed in Chapter 3 concerns the second definition. Indeed, given out segmentation task, we only have a dot annotation kind of labels that we try to exploit to generate coarse-grained ground truth masks for the segmentation problem.

Transfer Learning

As described in previous chapters, deep learning algorithms need a large amount of labeled data proportional to the complexity of the model to train effectively. In many research fields having a large-scale well-labeled dataset is particularly expensive if not unfeasible. Transfer learning is an approach that seeks to alleviate this problem by exploiting the information a model has acquired by training on one dataset to solve a new task on a different dataset. In fact, in common practice, it may happen to have at one's disposal several datasets inherent to the same domain but of which only a part of them is largely labeled. In these cases it is possible, by transfer learning, to obtain a model that by exploiting the information acquired from the supervised datasets manages to perform a new task on a second dataset that would otherwise be difficult to analyze.

This procedure may have positive effects, reducing the number of images required on the new dataset to get a better model, but only if there is enough similarity between the source and target datasets. Otherwise, we can also experience a negative effect, degrading the model performances. In fact, transfer-learning acts like a weights initialization that can provide an advantage during the optimization process. If the minimum relative to the source task turns out to be farther than a random initialization of weights, however, it becomes clear what has been said so far about the possible negative effect of transfer learning.

When we talk about transfer learning we usually refer to a domain rather than a dataset. A domain consists of two parts, the feature space X and the associated marginal probability $P(X)$ where $X = x1, ...x_n \, \epsilon \, X$. For a given

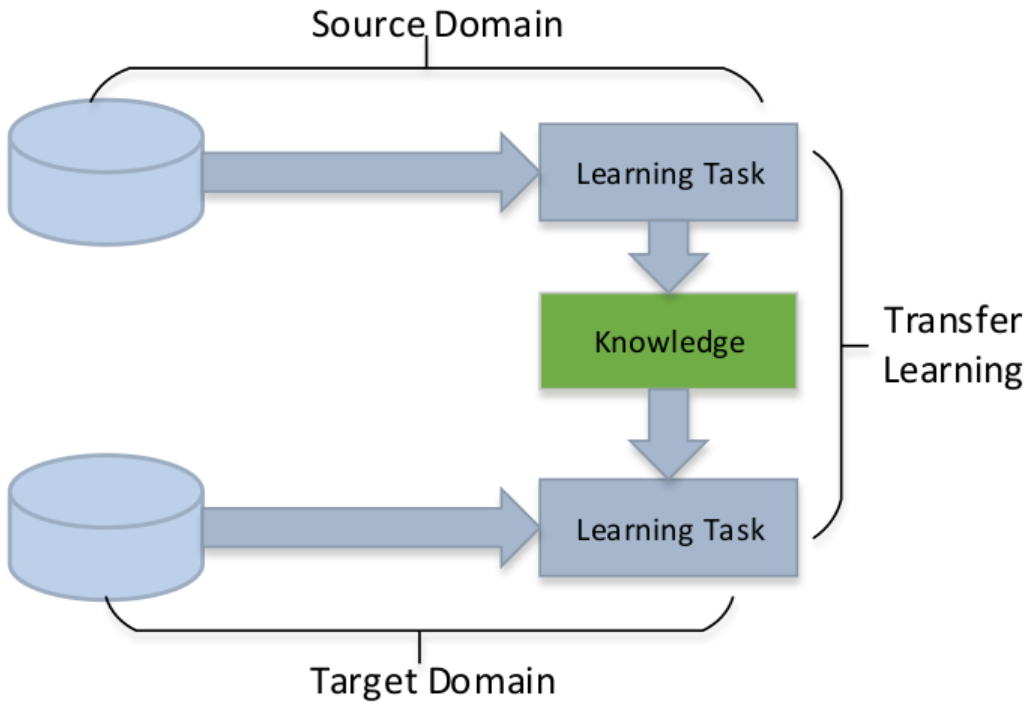


Figure 1.2: **Transfer learning.** The knowledge acquired on the source domain can benefit the learning of a new task on a target dataset.

domain D, a task T consists of two parts, the label space Y and the function $f(.)$ that is learned via the pair (x_i, y_i) of feature vectors and labels y_i with $x_i \epsilon X$ and $y_i \epsilon Y$. Now, we can state a definition of transfer learning. Given a source domain D_s and a relative task T_s, transfer-learning is a procedure whose aim is to ease the learning of a better function f_T related to a target task T_t on a target domain D_T by using related information from D_s and T_s, where $D_S \neq D_T$ or $T_S \neq T_T$.

The first shallow categorization of transfer learning depends on the relation between the source and target domain. Reacalling that $D = \{X, P(X)\}$, if $D_S \neq D_T$ means that $X_S \neq X_T$ and/or $P(x)_S \neq P(x)_T$. If $X_S \neq X_T$ we refer to heterogeneous transfer-learning, otherwise, we fall in the homogeneous transfer-learning case Weiss et al. [2016]. Instead, when $P(x)_S \neq P(x)_T$, the marginal distribution in the input space is different between the source and domain and this is one of the cases in which the transfer-learning can have a detrimental effect on the model training since the two datasets can present very different samples.

Transfer-learning is also defined by the task $T = \{Y, f(.)\}$ or equivalently $T = \{Y, P(Y|X)\}$. Therefore we can have both $Y_S \neq Y_T$ or $P(Y_s|X_s) \neq P(Y_t|X_t)$. In the first case, the label space, and so, the output of the task change moving from source to target task. In the latter case, the occurrence of the different classes is different.

For what concern the topic of this thesis we will focus on the case in which $D_S \neq D_T$ but with $T_S = T_T$, that is, the framework of the case reported in Chapter4. Also, concerning the same investigation, we will consider a transition between a supervised into a weakly-supervised domain that represents a case, not yet documented to the best of our knowledge.

Approaches

We can define four categories of transfer learning that are widely recognized by the research community Tan et al. [2018]:

- Instances-based deep transfer learning: reuse the instances from the source domain by appropriate weighting;

- Mapping-based deep transfer learning: learn a common feature space with better similarity for the two domains;
- Network-based deep transfer learning: exploit a network pre-trained on the source domain;
- Adversarial-based deep transfer learning: use adversarial training to find features suitable for both domains.

The first approach consists of finding similar instances between the source and target domain. These selected instances are weighted properly using a similarity criterion and are then used for the training of the model. Some algorithms like **TrAdaBoost** have been used (Yao and Doretto [2010],Huang et al. [2012]) to filter out instances in source domains that are dissimilar to the target domain. Other works like Xu et al. [2017] instead, have been focused on learning the weight for the selected instances.

Mapping-based deep transfer learning is a different approach aimed to learn a new data space where the instances coming from different sources are transformed to get closer. After that, all the instances can be jointly used to train the algorithm. One example of this approach is described in Pan et al. [2010], where transfer component analysis (TCA) is introduced. Other works compared the distribution of original data sources using a deep neural network, by means of an adaptation layer and an additional domain confusion loss. The aim is to learn a representation that is both semantically meaningful and domain invariant Tzeng et al. [2014].

Probably, the most known approach is still network-based deep transfer learning. This approach consists of a pre-trained stage during which the model learns some general features that can be reused on a different domain. Specifically, after the training on a source domain, a pre-defined set of layers is copied into a new network and frozen. The remaining part of layers is left free to train again on the new target samples. It is worth remembering that, generally, lower-level layers learn a general representation of the inputs. A clear example is the tendency of Gabor filters and color blobs to show up in the first layer of neural networks trained on natural images. These filters

are learned independently from the specific task. Indeed, these layers are usually copied into a new model and frozen to avoid their re-train during the fine-tuning stage. On the other hand, high-level layers learn task-specific features and so they need to be retrained to move on to another task. Several papers (Yosinski et al. [2014] Lee et al. [2022] Guo et al. [2019]) study the concept of the layers transferability from one domain to another to optimize the transfer-learning procedure. It is important to understand, for example, task by task, which are the layers to be left frozen and which need to be retrained. One of the main works in this sense is Yosinski et al. [2014] where the authors quantify the transferability of features from each layer of a neural network, which reveals their generality or specificity.

They also showed how transferability is negatively affected by two distinct issues: optimization difficulties related to splitting networks in the middle of fragilely co-adapted layers and the specialization of higher layer features to the original task at the expense of performance on the target task. In any case, they claim that also transferring features from a very distant task can translate into a boost of performance and generalization showing that these results still linger after substantial fine-tuning. Other works Lee et al. [2022] investigate further which layers should be unfrozen during the fine-tuning procedure focusing on the relation between the source and target dataset. They tested their results across several public datasets showing how, depending on the distribution shift, can be better to fine-tune a limited and specific set of layers. In Guo et al. [2019] the authors push forward this study by investigating instance by instance what layers should be unfrozen during the training. They use an auxiliary network that defines for each sample which block of layers of the main network to unfreeze.

Other aspects became relevant during the transfer learning. Usually, weight decay is applied as a regularization term to bind the value of the weight under a pre-defined value. Sometimes this approach can lead to a poor optimization process that neglects the information acquired during the first pre-training phase. In the work Li et al. [2020], it is shown how different regularization terms like di starting-point L^2 ($SP - L^2$) exploit better the weight initialization obtained after the pre-training stage. Indeed, usually,

only the L^2 parameter regularization, also known as weight decay, is used in machine learning to restricts the capacity of the trained model by restraining the effective size of the search space during optimization, implicitly driving the parameters towards the origin. Introducing the starting point SP obtained after a pre-training phase as the reference for parameter regularization may help to achieve a better convergence.

The last category regards adversarial-based deep transfer learning. It is the most recent approach but already shows some potential and is based on the assumption that *For effective transfer, a good representation should be discriminative for the main learning task and indiscriminate between the source domain and target domain* Tan et al. [2018]. Adversarial-based deep transfer learning accomplish with this task training a model to learn from one or more source domains and generalize to a target domain. The idea is to leverage the knowledge learned in the source domain(s) to improve the performance of the model in the target domain. In this technique, the transfer of knowledge from the source to the target domain is achieved through adversarial training. Specifically, the model is trained using a combination of two objectives: a classification objective and a domain-adversarial objective. The classification objective is used to classify the input data into different classes. The domain-adversarial objective is used to encourage the model to learn domain-invariant features. This is achieved by training a domain discriminator that tries to distinguish between the source and target domains, while the feature extractor (i.e., the main model) is trained to confuse the discriminator by producing features that are indistinguishable between the domains. During training, the feature extractor and the domain discriminator are trained alternately to optimize their respective objectives until a convergence criterion is met. Once the training is complete, the feature extractor can be used to extract useful features from the source domain that can be transferred to the target domain. This process is depicted in fig. 1.3. Practically, The adversarial layer aims to discriminate between the source of the features extracted from the front layer of the network. If the adversarial layer performs worse, it means that there is a small difference between the features, and consequentially better transferability across the domain is

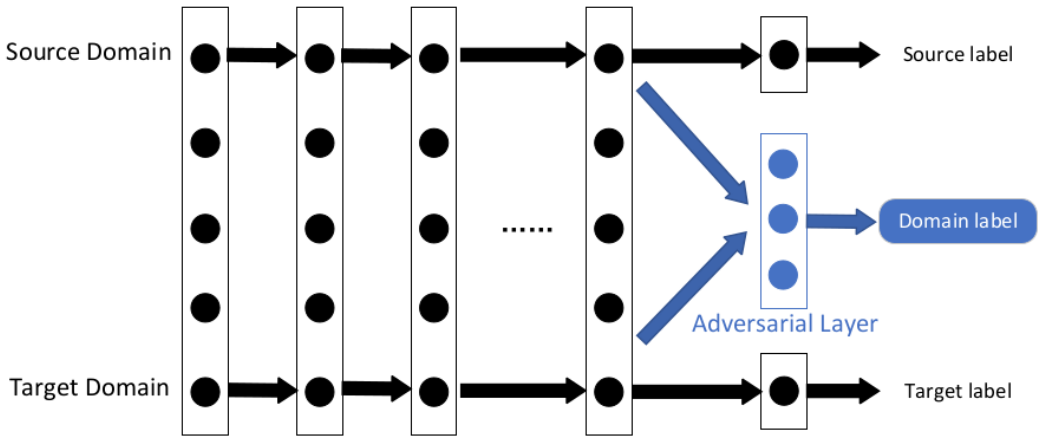


Figure 1.3: **Adversarial-based deep transfer learning.** The adversarial training promote to define a good shared representation between source and target domain.

guaranteed.

1.3 Fluorescent microscopy

A fluorescence microscope is an optical microscope that uses fluorescence instead of, or in addition to, scattering, reflection, and attenuation or absorption. Fluorescence is a luminescence phenomenon that occurs in some molecules, denominated fluorophores that are able to emit light when they are in electronically excited states. These molecules, absorbing light at a specific wavelength, transit from the ground state energy to an excited one. When it happens, the fluorophore becomes unstable and releases the absorbed energy by emitting light of a longer wavelength to get back to the ground state. The difference between the exciting and emitted wavelengths, known as the Stokes shift, is the critical property that makes fluorescence so powerful. To visualize only fluorescent objects, all excitation light is filtered out while the emitted fluorescence is allowed to be seen. Fluorophores are molecules that are utilized for their fluorescent properties. When these compounds absorb light energy, changes in the electronic, vibrational and rotational states of the molecule can occur. Oftentimes, this absorption of

energy causes an electron to move to a different orbital farther away from the nucleus, transitioning to an excited state. Eventually, the absorbed energy is released through vibrational relaxation and fluorescence emission, returning the fluorophore to its low-energy ground state Sanderson et al. [2014].

Many works aimed to capture biological effects are based on this method. The fluorophore is designed to couple with the molecular structure of the tissues to monitor. Then, to asses, the evidence of an underlying biological process, a quantification of the tissue activity should be assessed. At this point became relevant to count how much fluorescence occurs in the several areas under study (Hitrec et al. [2019], Hitrec et al. [2021] da Conceição et al. [2020]). Many studies, for example, investigating the neuron's communication network established during the torpor onset, rely on the counting of the cells that activated during the transition to this particular metabolic state. Indeed, these cells, stained with fluorophore became detectable thanks to the light emission. Torpor is characterized by reduced body temperature and depressed metabolism representing a defense weapon for some animals to face hostile environmental conditions. A deeper understanding of the key factors promoting the torpor condition may help to trigger its activation also on human patients. Indeed, coming to the human applications, it is thought that this approach can bring many benefits when dealing with patients that need invasive surgery Bouma et al. [2012] Bellamy et al. [1996]. Also, triggering the torpor on humans could be a means to make feasible long interplanetary trips where the side effect of space travel can hazard the health of the passengers (Cerri et al. [2016] Cerri et al. [2021]) especially concerning the resistance to radiations (Puspitasari et al. [2021] Puspitasari et al. [2021]).

Experiments that use molecular fluorescence to study biological processes often rely on semi-automatic techniques to acquire and process images correctly and to count interesting biological structures. These techniques involve multiple steps such as area selection, white balance, calibration, and color correction in order to identify structures of interest (Dentico et al. [2009], Gillis et al. [2016], Luppi et al. [2019]). However, this process can be time-consuming and tedious, leading to errors due to operator fatigue. Additionally, it can be challenging to distinguish structures of interest from

the background, leading to subjective and arguable counts. For this reason, starting in the next section, we introduce the feasibility of a methodology based on a deep learning algorithm. Specifically, we target a convolutional neural network (CNN) as the key architecture to automate the counting of objects of interest in molecular fluorescence images. This approach will facilitate and speed up future research in this field by saving time and human effort, and by eliminating operator fatigue errors.

1.4 Counting objects

Counting objects in digital images is a common task with many real-world applications (Segui et al. [2015], Arteta et al. [2016], Paul Cohen et al. [2017], Rahnemoonfar and Sheppard [2017]) and various methods have been proposed to automate the process (Ciresan et al. [2013], Ciresan et al. [2012], Lempitsky and Zisserman [2010], Kraus et al. [2016], Raza et al. [2017] Faustino et al. [2009]). One of the approaches most frequently used for the counting task is based on the counting-by-regression approach ([Hoekendijk et al., 2021], Xue et al. [2016] Hobley and Prisacariu [2022]). Under this strategy, convolutional neural networks are trained to predict the cells' number that is present inside pictures. For example, the authors of Xue et al. [2016] cast the cell counting task as a regression problem using the global cell count as the image label. They first split the images into different patches and, after an augmentation process, a convolutional neural network architecture is trained by means of a Euclidean loss function to regress the correct number of cells. They also provide a density map but its low spatial resolution doesn't seem to provide a clear localization of the cells. Instead, leveraging the attention mechanism, in the work Hobley and Prisacariu [2022] is presented a method that learns to count referenceless exploiting only the total amount of objects as supervision. They show that a general feature space with a global context can enumerate instances in an image without a prior on the object type present.

Sometimes we have some further information other than the simple amount of objects in the images. For example, dot annotations are a type of label

frequently used to localize the objects since they are less time-expensive to obtain respect to the bounding box and especially respect to the pixel-level annotation (segmentation masks). Precisely, these labels represent the coordinates of a point (preferably the center) inside the object to localize. Usually, in these cases, the target generation consists of placing a gaussian probability kernel on top of each dot annotated point representing the center of an object to detect. The regression, in this case, consists of predicting the pixel-by-pixel probability to have an object center at that specific point He et al. [2021].

Count-level annotations and dot annotations represent respectively a weak and a full-supervised label when counting is the final scope of the work. Sometimes, to reduce the time spent on labeling, these two labels are used together like in the work Lei et al. [2021] that describes a successful application of this strategy. Counting-by-detection is another approach used to quantify the number of interesting objects inside a picture. In this case, the method provides bounding-boxes labels to learn how to detect an object and one of the most used architectures is certainly the YOLO one Redmon et al. [2015]. For example in the work Alam and Islam [2019] the authors present a machine learning approach for automatic identification and counting of three types of blood cells using the 'you only look once' (YOLO) object detection and classification algorithm. To further refine the localization of the detected objects, another approach may be pursued. Counting-by-segmentation improves the model's detection and localization capabilities by incorporating semantic labels for each pixel in the ground-truth masks. This enhances the ability to discern the precise boundaries of each object through pixel-wise classification. The final count is determined by analyzing groups of connected pixels ([Hernández et al., 2018] Morelli et al. [2021b] Morelli et al. [2021a]). The main step of this approach is represented by a segmentation step. Especially for the biomedical domain, UNet Ronneberger et al. [2015] model represents a pillar architecture for such a task and is frequently adopted to segment nuclei, cells and other organs (Zeng et al. [2019], Hu et al. [2019], Weng et al. [2019], Le'Clerc Arrastia et al. [2021]). Starting from the next chapter, we describe the application of such an approach under two different supervision

conditions. In chapter 2 we frame under a fully-supervised learning paradigm while in chapter 3 we pursue the same approach but rely on dot-annotation that represents only weak supervision concerning the segmentation output.

Chapter 2

Supervised approach

2.1 Introduction

In this chapter, we undertake the cell counting problem in biological images when ground truth segmentation masks are available. Among the available approaches illustrated in the introduction section 1.3, we pursue the counting-by-segmentation one to favor a better localization of the stained cells. Our work aims to facilitate and speed up the cell counting process by developing a CNN that detects objects of interest without human intervention. The use of a Deep Learning model would not only save time and effort but would also eliminate operator fatigue errors and reduce the subjectivity of borderline cases. We also define different experiment set-ups to analyze the contribution of some specific training procedures introduced to get better segmentation and detection metrics. Specifically, we introduced a novel weighted map to penalize the error on the boundaries of the touching cells. Moreover, we describe a strategy to oversample underrepresented artifacts during the training. We evaluated the different architectures and training designs through an ablation study.

We will start with a brief introduction of deep learning methods frequently used for the segmentation task. Then the cTB dataset used for the experiments is introduced. After that, we define the ablation studies scheme and the model architectures compared. Finally, we discuss the implications

of our work and outline possible directions for future research. All the results provided in this chapter are built on the work Morelli et al. [2021a]. The code used for these study is collected at the following link: (cell counting yellow supervised and weakly supervised approach for CTb dataset)

2.1.1 Related works

One of the pillar works concerning the segmentation in biological images is reported in the paper Ronneberger et al. [2015] which described a deep learning approach for cell segmentation. They proposed the U-Net architecture characterized by the typical U shape consisting of an encoding and decoding path respectively used to capture important features and to furnish a precise localization. This architecture becomes a state-of-the-art solution and the basic architecture for various applications.More recently, an extension of the U-Net has been proposed introducing units named residual blocks He et al. [2016] with short-range skip connections. This modification helps to prevent the vanishing gradients (Clevert et al. [2015], Hochreiter [1998]; Guan et al. [2020]; Cao et al. [2020]; Qamar et al. [2020]) effect that may occur when too many layers are included in the architecture. Moreover, these blocks also help to reach a minimum smoothly and in lesser time. Leveraging the U-Net and its modifications, a lot of other works start to address the segmentation task, especially in the biomedical field. In the work, Kolařík et al. [2019] the authors described a fully automatic method for high-resolution 3D volumetric segmentation of medical image data introducing 3D Dense-U-Net architecture that implements densely connected layers. Another work Küstner et al. [2020] focused on the 3D segmentation aim of the quantification and localization of different adipose tissue compartments from whole-body MR images. A reliable automatic segmentation of adipose tissue into subcutaneous and visceral adipose tissue is proposed using a 3D convolutional neural network, the DCNet still inspired by the U-Net structure. Instead, in the work Wang et al. [2022] the authors combine the novel attention mechanism Vaswani et al. [2017] mutated from the transformer architecture, to improve the U-Net architecture and provide more accurate segmentation.

Moving to the microscopy domain,Kraus et al. [2016] combined deep CNNs with multiple instance learning to classify and segment microscopy images using only whole image-level annotations. Vigueras-Guillén et al. [2022] use a modification of U-Net, DenseUNets to provide the binary segmentation of the corneal endothelial cells and estimate some relevant biological parameters. Other methods based on the modification of ResUnet Jha et al. [2021] proved to be effective in the detection of colorectal cancer and its precursors. Here, improvement with respect to the previous version of their ResUnet++ Jha et al. [2019] is achieved by using conditional random field and test-time augmentation.

2.1.2 Contributions

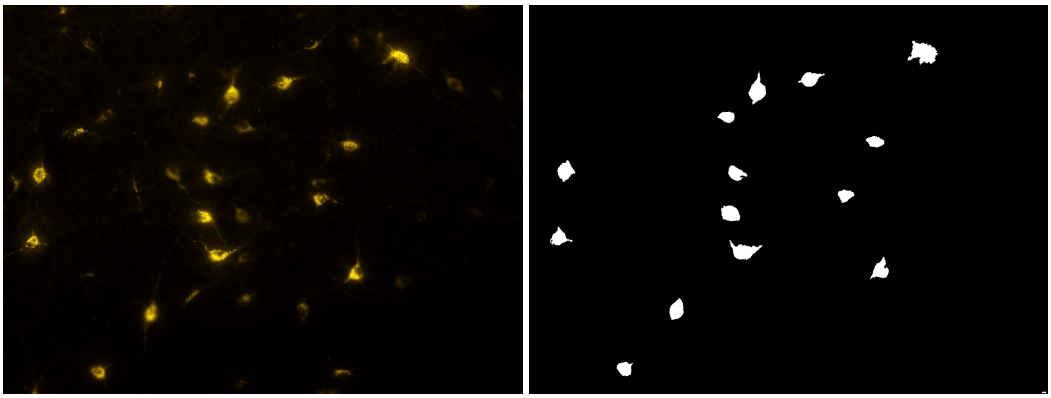
Our work focues on a supervised learning approach to count cells, specifically neurons, in fluorescence microscopy images providing a segmentation map to show which cells contributed to the total counts. We test different architectures and experiment designs through an ablation study and validate the results by using both detection and counting task metrics. The model we introduced, c-ResUnet, trained with the weighted maps results in the best model metrics. Our contributions are mainly twofold:

- Introduction of c-ResUnet, a model specifically designed for counting cells also when highly clustered.
- Description of a processing pipeline to segment and post-process the results
- Introduction of novel weighted maps that penalize errors on cell boundaries promoting accurate segmentation

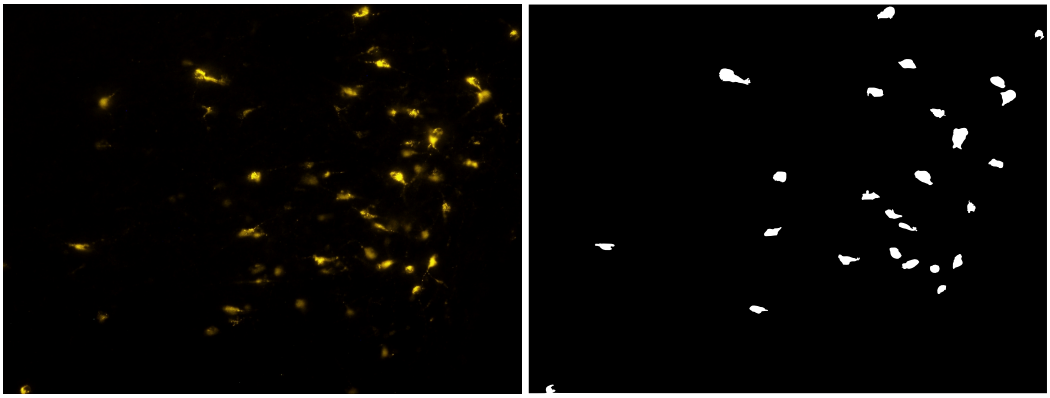
2.2 CTb Dataset

The Fluorescent Neuronal Cells dataset (Fluorescent Neuronal Cells) consists of 283 high-resolution (1600×1200) images of mice brain slices, along with their corresponding ground-truth labels. These images were obtained

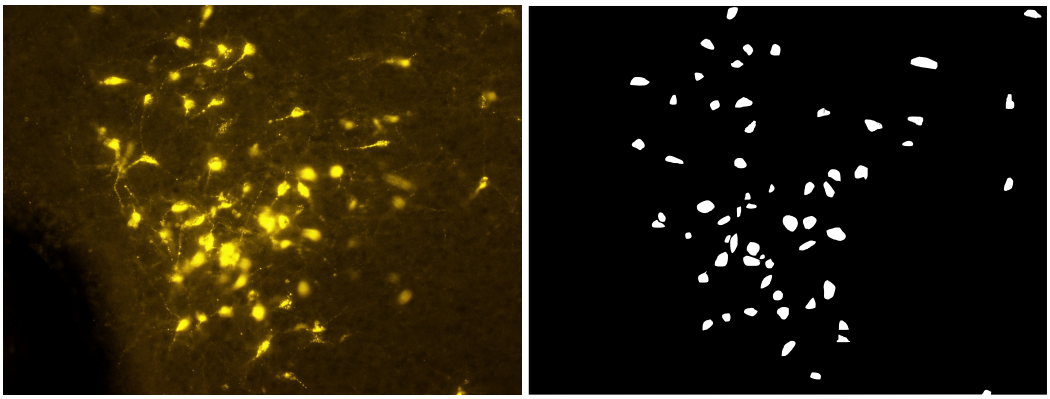
through fluorescence microscopy, using a monosynaptic retrograde tracer (CTb) injected into specific brain structures to highlight only the neurons whose signaling is linked across distinct brain areas. The resulting images show neurons of varying sizes and shapes as yellowish spots with variable brightness and saturation against a generally darker background 2.1. However, some images 2.1a exceed this rule and the background assumes values closer to those of the cells making the recognition task harder. Indeed, despite efforts to standardize the acquisition process, these images present several challenges for accurate detection due to their variability in brightness and contrast, as well as the presence of clutter and overlapping cells. The cells themselves may also exhibit varying levels of saturation due to natural fluctuations in fluorescent emission properties, and their shape may change significantly, making it difficult to distinguish them from the background. In addition, artifacts and bright biological structures, such as neurons' filaments, may interfere with recognition, as well as non-marked cells that are similar in appearance to the stained cells. These factors not only complicate the training process but also make it difficult to evaluate the model as the interpretation of such borderline cases becomes subjective.



(a) image (b) ground truth



(c) image (d) ground truth



(e) image (f) ground truth

Figure 2.1: Sample data.

2.2.1 Ground-truth labels

Supervised ground-truth labels correspond to the output that we want the model learns to reproduce during the training phase and to generalize on unseen test data. For the segmentation task, the ground truth images correspond to a set of binary masks labeled pixel-by-pixel into two distinct classes: (1) the neuronal cells and (2) the surrounding background represented by biological tissue or cells excluded by the synaptic signaling process. Indeed, the cells we want to recognize are only those that have undergone a specific biological process during the experimental assessment and are generally visible thanks to higher pixel intensity values. However, we will see that in these pictures sometimes some activated or stained cells may look very similar to

the nonactivated ones. The degree of similarity sometimes is such that also the most experienced annotators can be confused.

Obtaining these target masks can be time-consuming and required a lot of human resources. To make the labeling process more efficient, we used an automatic procedure that started with a large subset of 252 images. We first clean these images from noise by applying gaussian blurring and then thresholded using automatic histogram shape-based methods. From this operation, we obtained images containing both good cell candidates and a significant number of undesired objects like an artifact, non-activated cells, and other biological tissue whose pixel intensity get a value similar to those of the stained cells. So, these images need to be reviewed by experts to eliminate these false positives and irrelevant artifacts. Sometimes, further adjustments of the cell boundaries were required mainly for two different reasons. First, the thresholding procedure appeared to be inaccurate clustering into a unique object with many distinct cells that are close together. In this case, the annotators took care to separate the cells drawing black pixels line across the cells' boundaries. The other reason regarded the inaccuracy of the thresholding technique that often resulted in bigger cell shapes. In these cases, the object masks included significant parts of the surrounding background biasing the model. The remaining part of samples, nearly 30 pictures, were manually segmented by domain experts, including those with unique features such as artifacts, filaments, and crowded objects to ensure highly reliable masks for challenging examples. While deep learning has gained a lot of popularity in computer vision in recent years, a lack of annotated data can be a problem when dealing with non-standard tasks or images. To overcome this problem, one common approach is to fine-tune models pre-trained on large datasets of natural images like ImageNet or COCO, using as few new labels as possible for the specific task at hand. However, this strategy may have a negative effect when the source dataset is significantly different from the target one as describer in the section 1.2. By releasing our annotated dataset and pre-trained model, we hope to advance the field of biomedical imaging through the automation of manual operations and to promote research into new data analysis techniques for microscopic fluorescence and similar domains. More-

over, the next chapter of the thesis describes two possible approaches to deal with weak annotations. Specifically, we will use dot annotation which labels a way easier to obtain. The same problem, counting-by-segmentation, is considered both on the same dataset and extended to another sample of fluorescent neuronal cells characterized by a different variety of shapes and sizes. Instead, concerning the fine-tuning procedure, another chapter 4 will be focused on an extensive study to understand better the degree of attainment between two models trained on the same domain but on a different dataset.

2.2.2 Data description

In the following, we report the main of the cells represented inside the pictures, but a more detailed data exploration on such a dataset is provided in Clissa [2022]. From this latter work, we report a table summary in table 2.2 of the relevant cell features.

	red intensity	green intensity	blue intensity	signal (%)	signal ratio	area (μm^2)	Feret diameter (μm)	# cells/image
count	1,920,000	1,920,000	1,920,000	283	283	2,137	2,137	283
mean	7.32	2.83	0.20	0.50	366,681.94	119.30	17.48	27.05
standard deviation	16.81	13.30	1.43	0.61	755,628.42	97.96	8.20	21.75
min	0	0	0	0	19.57	15.94	5.86	0
10%	0	0	0	0	92.39	35.23	9.42	4
25%	2	0	0	0.09	145.35	55.51	11.95	7
50%	5	1	0	0.34	291.10	89.86	15.53	21
75%	9	2	0	0.68	1,163.29	148.02	20.86	48
90%	12	4	0	1.07	1,920,000	237.09	27.61	59
max	252	251	87	4.86	1,920,000	796.39	67.48	68

Figure 2.2: **Distribution summary** Clissa [2022]. Summary of the distributions illustrated. For each distribution are reported the mean and standard deviation; minimum, maximum and 10-th, 25-th, 50-th, 75-th and 90-th percentiles; the count of objects from which such measures are computed, i.e. pixels, images, and cells..

From the table, we can remark on some interesting features like the mean number of cells per image and their mean size reported in terms of areas and feret diameter. The first characteristic is an index of the class representation that, in this case, is unbalanced in favor of the background class. The

second characteristic is useful information to optimize the post-processing 2.3.3 step helping to filter out the spurious objects from the segmentation output. From these statistics, we can also observe the RGB distribution that highly promotes the red and green intensity while the blue channel is scarcely populated. This is an obvious consequence of the acquisition procedure that is target a narrow wavelength range. However, to handle the color representation we decided to leave the model to learn the most appropriate transformation introducing an initial layer that maps the RGB image to a 1-channel representation as described in the section 2.3.1. Another useful piece of information regards the distribution of the pixels across the two classes (neuronal cells and backgrounds) reported in Fig. 2.3 where we refer to the pixels belonging to a neuronal cells as the *signal* pixels.

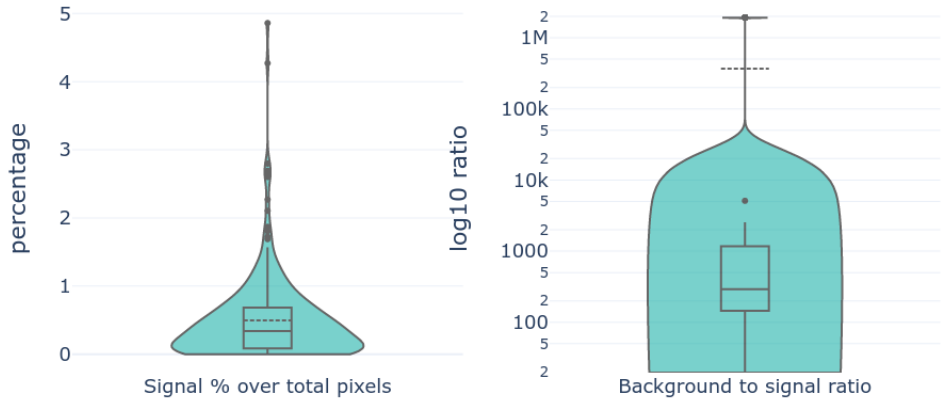


Figure 2.3: **Distribution summary** figure from Clissa [2022]. Summary of the distributions of the signal pixels (pixels representing a neuronal cells) with respect to the background pixels. For each distribution we reported the mean and standard deviation; minimum, maximum and 10-th, 25-th, 50-th, 75-th and 90-th percentiles; the count of objects from which such measures are computed, i.e. pixels, images, and cells..

By analyzing the number of pixels that belong to the background and the signal, it becomes apparent that the two classes are heavily imbalanced (see

the *signal (%)* and *signal ratio* columns in Fig. 2.2 for a numerical summary). On the left of figure 2.3,it is shown a violin plot of the percentage of signal pixels out of the total image pixels for the 283 images. This evidence is remarked by the median value of 0.34% and a 90-th percentile of 1.07%. This means that about 90% of the images contain less than 1% of pixels that belong to the signal. From the right side of the same figure 2.3 we can deduct that the background pixels are roughly 20 to 300 times the number of signal pixels in 50% of the images but this is also due to the presence of empty masks that cover more than 10% of the total masks. These findings highlight the need for specialized training strategies to address this strong class imbalance and enable the model to correctly classify image pixels. For a more detailed dataset description please refer to Clissa [2022].

2.2.3 Challenges

As anticipated in the section 2.2, the image acquisition may present some difficulties due to the fluorescence emission intrinsic properties resulting in some traits that make the detection task harder. We already provide some examples of saturation shift that sometimes make the surrounding background get intensities value similar to those of the cells to detect 2.1f. In this case, the model should learn a set of features that don't rely only on the pixel's intensity value but that should consider also other characteristics. Indeed, this is what we expect a deep-learning model to do during the training phase. However, the neuronal cells depicted in this dataset can assume a wide range of shapes and dimensions challenging the model to adapt consequentially. This feature is due to the acquisition procedure that reports a 2-D projection of 3-D biological tissues. Indeed, while some of the cells can be shot lying on a plane parallel to the acquisition one looking like an elongated neuronal shape, other neurons whose axis is oriented perpendicular to this direction, will more likely appear as a circular bright point. In this regard, sometimes the fluorophore trapped in a limited biological area can generate emissions that look very similar in shape and pixel value intensity to these smaller neurons. In Fig. 2.4 we report an example of these kinds of artifacts

(points on the right side of the images) together with an extreme case of artifact emission represented by the rectangular elongated shape. Another source of noise derived from the biological filaments depicted in Fig. 2.3.1 that spread across the right side of the image. These structures may assume shapes and intensity values pretty similar to the neurons making them harder to discard by the model. In the same image, we can notice another confounding structure represented by the elongated strip on the right side of the picture. Lastly, we report a case where the cells tend to cluster together hidden by a fluorophore agglomerate that encapsulates the cells into one big blob. Here, the model needs a good strategy to learn how to divide these distinct objects. In the method, we will describe the weighted maps we used to enforce the cell separation operated by the model.

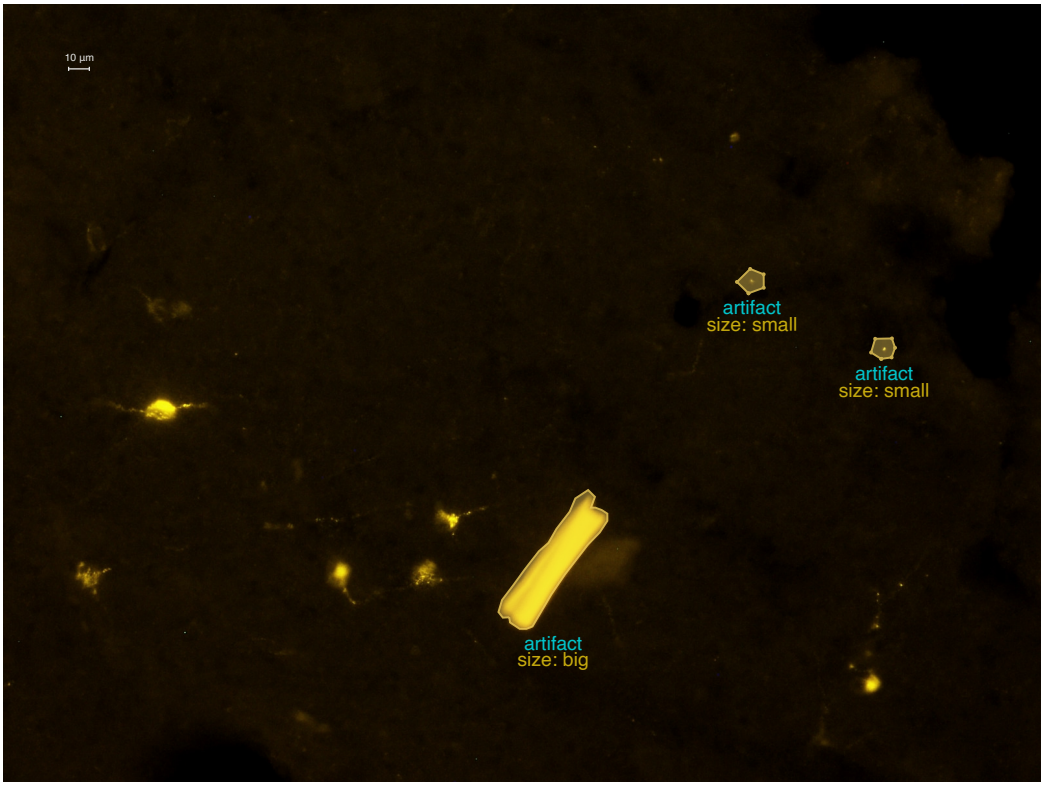


Figure 2.4: **Challenges and Artifacts** figure from Clissa [2022]. Dot shaped light emission looking like stained neurons and elongated rectangular shape artifact.

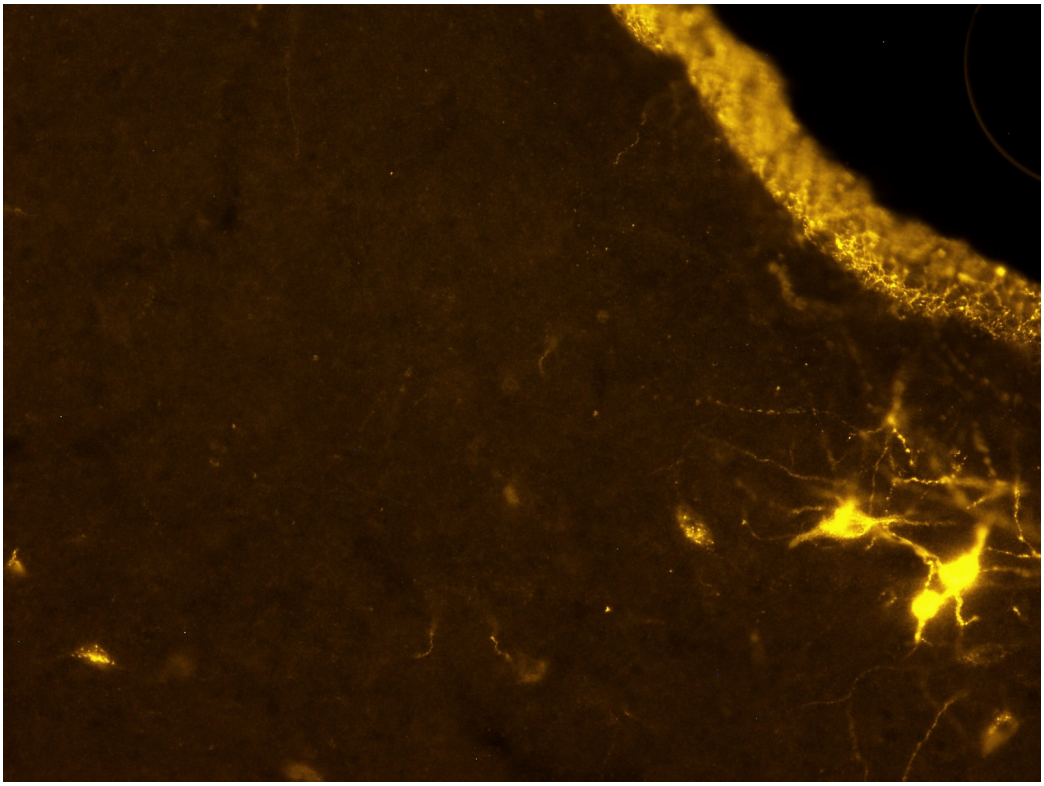


Figure 2.5: **Challenges**. Biological filaments and stripe.

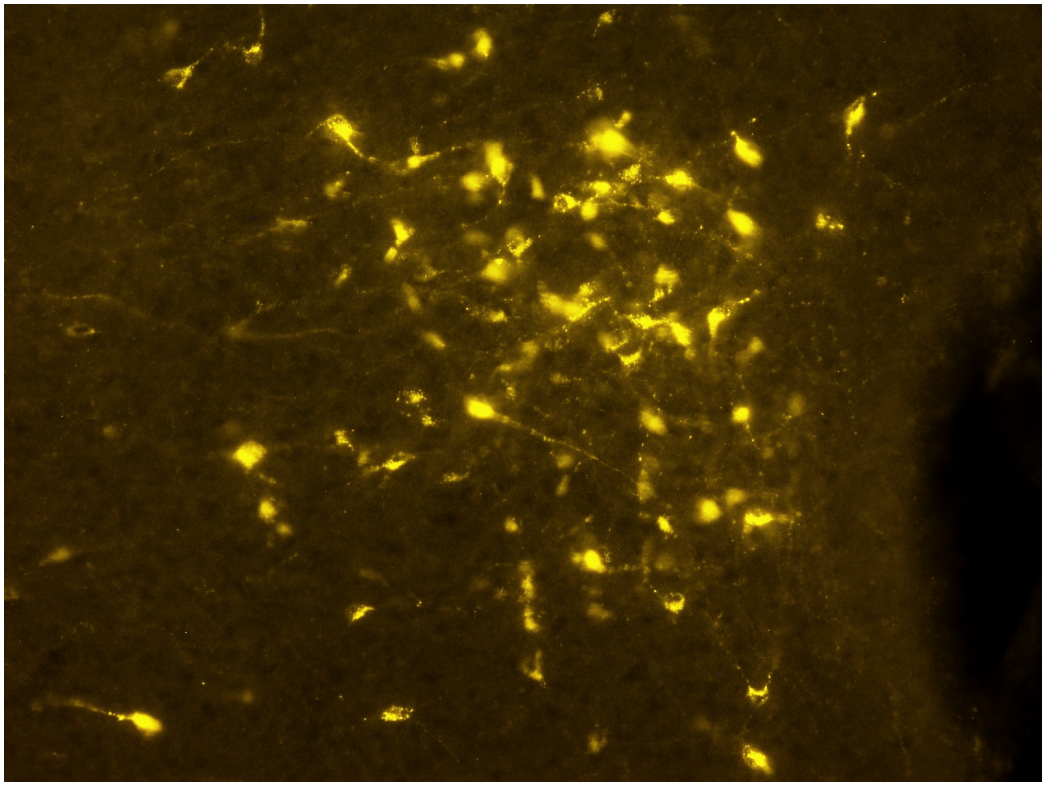


Figure 2.6: **Challenges**. a picture with several example of high density clustered cells.

2.3 Methods

In this work, we aim to segment and count cells in images using a supervised learning approach evaluating four different CNN architectures from the Unit and ResUnet families. The raw output of the models is a probability heatmap that, after a thresholding operation, reports the segmentation boundaries of the detected objects. We apply a dedicated post-processing pipeline to reduce the number of false negatives and false positives. An ablation study is performed to evaluate the models and the impact of study design choices used to limit the number of artifacts detected (false positive) and the number of missed cells (false negative) when clustered together. We will begin with the description of the model introduced in this work c-ResUnet and then will describe the different design choices adopted for the training stage. Finally,

we describe the post-processing pipeline and the evaluation methodology we use to compare the competing architectures.

2.3.1 Model Architecture

This section introduces the architecture we propose to handle the cells semgnation and detection. This model combine the innovative features of two pillars of the convolutional neural networks architecture that are the U-Net and the ResUnet models.

C-ResUnet

The architecture proposed is largely inspired by the U-Net family. It follows the typical U shape developed to enable high-resolution segmentation results. Also, during the design, we keep the long skip connection to combine low-level features with the high-level ones associated with the deeper convolutional layers. Instead, we replaced the conventional U-Net convolutional block with the ResUnet one to enable a better gradient backpropagation. Indeed, many works, established the efficacy of these blocks to face the vanishing gradient problems that affect the deeper architectures. However, many versions of the ResUnet block exist and a choice is needed to accomplish the most appropriate one. We opted for a sequence of *BatchNorm-Activation-ConvLayer* repeated twice for each block that is reported on the right in Fig. 2.7 compared to the standard unit block (on the left). From this picture, we can also notice the *identity path* needed to sum up the features maps obtained as a result of the residual block and the features maps input of the block. The resulting feature maps are then summed with the input of each block to complete the *identity block* design typical of the ResUnet architecture.

We inserted into the overall architecture some minor modifications. Specifically, we added an initial 1×1 convolution to simulate an RGB to 1-D space conversion which is learned during training. The model should have a representation that helps to address effectively the segmentation task. Moreover, we inserted an additional residual block at the end of the encoding path with 5×5 filters (instead of 3×3). These adjustments should provide

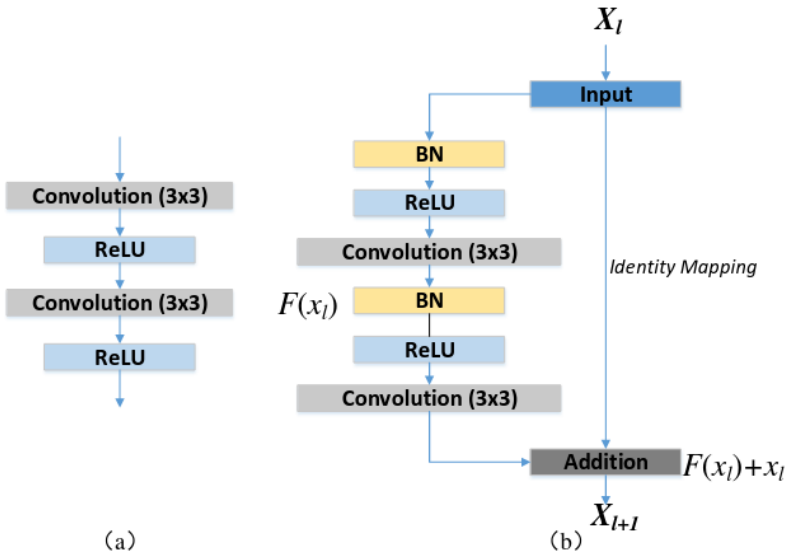


Figure 2.7: **Model architecture**. Comparison between the UNet convolutional block, on the left and the ResUnet one, on the right. The main introduction is represented by the identity mapping path.

the model with a larger field of view, thus fostering a better comprehension of the context surrounding the pixel to classify. This kind of information can be beneficial, for example, when cells clump together and pixels on their boundaries have to be segmented. Likewise, the analysis of some background structures can be improved by looking at a broader context. The resulting architecture is reported in Fig. 2.8 and it will be referred to as cell ResUnet (c-ResUnet).

Ablation studies

In this work, we compare four network architectures. However, we associate each model with an alternative training strategy relying on weighted maps that penalize more the loss on the high-density cell areas where the model may wrongly count multiple cells only once. To evaluate the impact of this strategy, we selectively switch on its contribution during the learning phase. Moreover, for the c-ResUnet model proposed here, we evaluate also a pre-processing step concerning the augmentation procedure to increase the

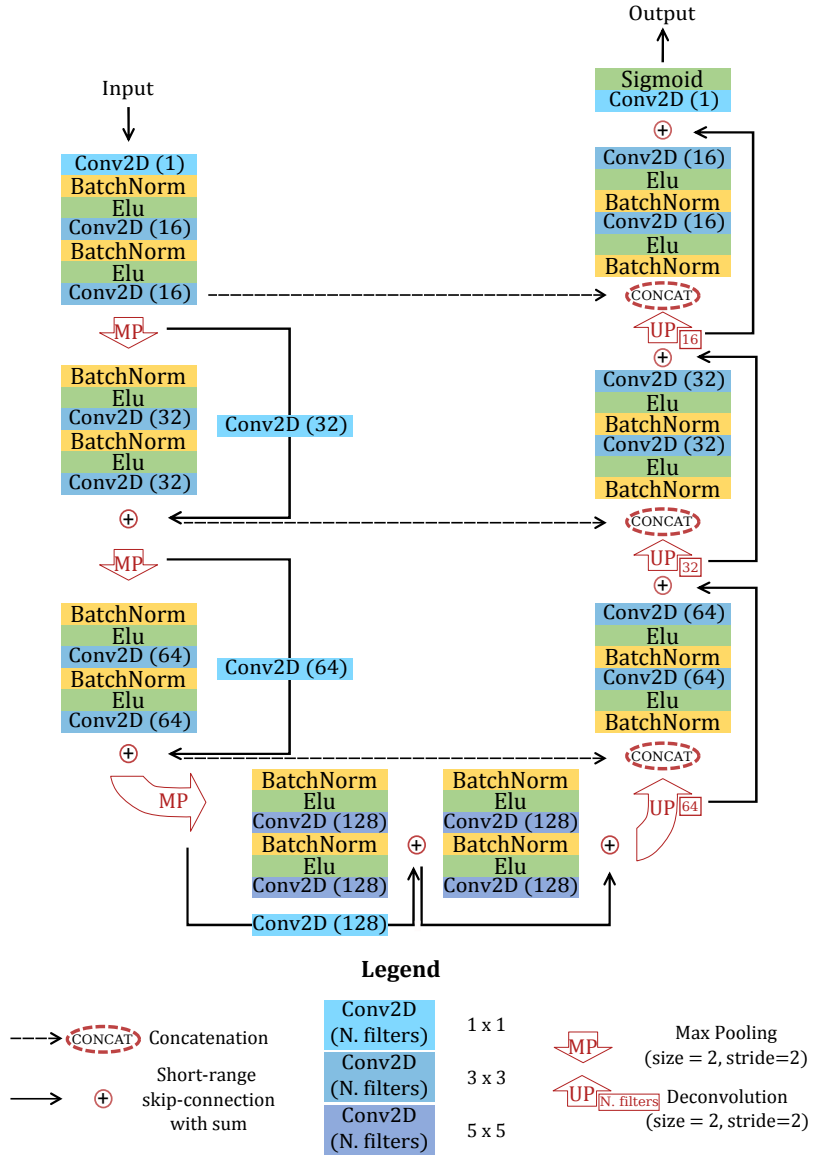


Figure 2.8: **Model architecture**. Scheme of the model. Each box reports an element of the entire architecture (individual description in the legend). The shortcut-connections along the encoding-path are supported by a 1×1 convolution to enable the final sum before the max-pooling operation Morelli et al. [2021a]

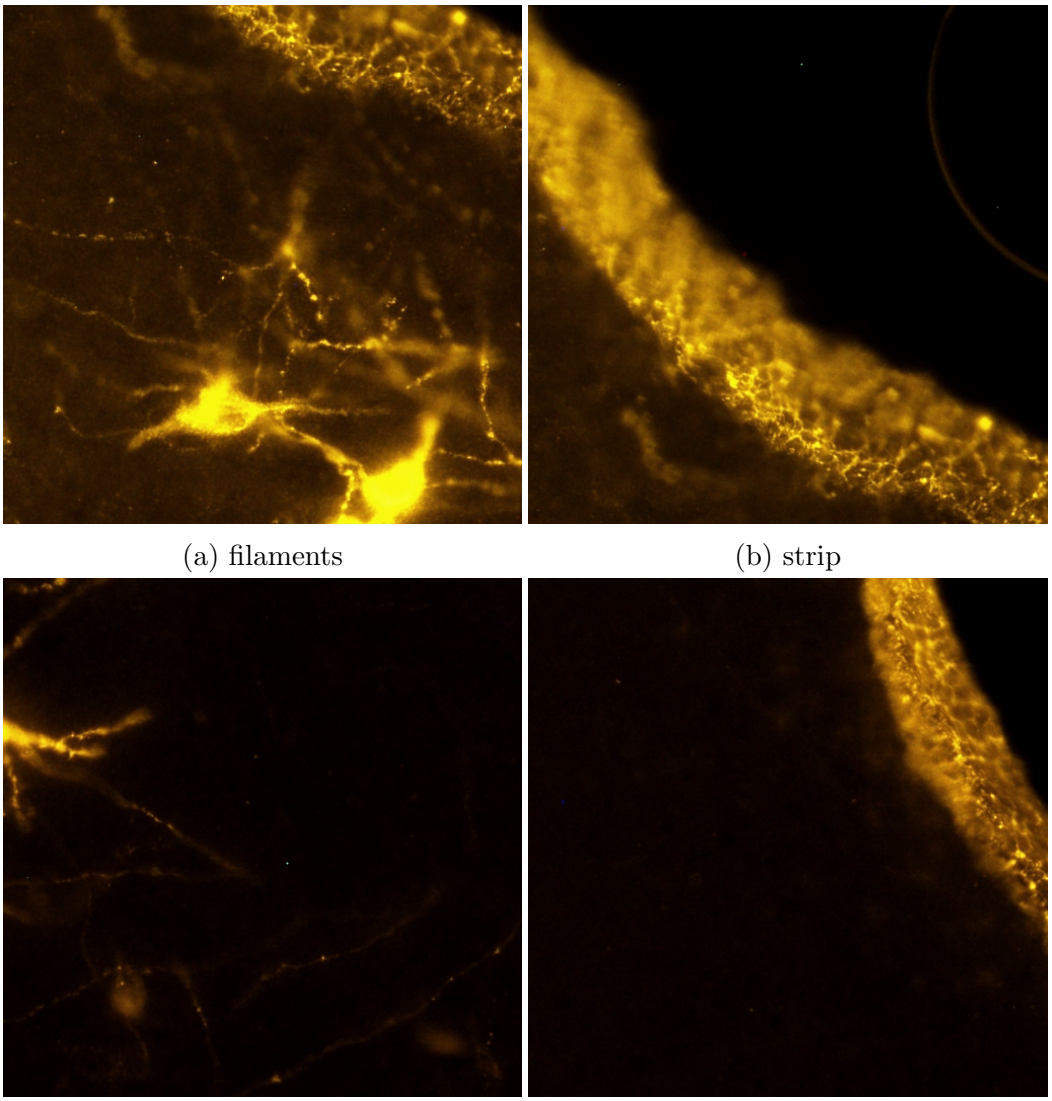
artifact recognition operated from the model (2.2.3. We consider these design choices in the ablation study which results are reported in section 2.4

Artifacts oversampling

As noticed described in the section 2.2.3, there are biological structures or artifacts depicted in the images that may cause the model to detect false positives. These structures can be difficult to recognize because they resemble cells in terms of saturation and brightness. Moreover, they are not well represented in the data limiting their correct identification. To address this issue, we increased the augmentation factor applied to these inputs in order to improve the model's learning. Specifically, we selected six crops that represented these structures and used the augmentation pipeline described in the *Model training* section to create 150 new images for each crop. It is worth noticing that these procedures are supervised by the expert of the domain since they only can confirm the ultimate nature of such components. While for some biological elements, like the stripe in Fig. , can be easier to guess if we are facing an artifact, in other cases, like the bright spots of the same image became more difficult. So, after the selection of the crop we ask for feedback to confirm the rightness of our choices.

Weighted maps

One of the main challenges when using the model for inference is accurately segmenting cells that are crowded together. If the boundaries between cells are not accurately identified, the model may connect separate objects, resulting in multiple objects being treated as a single entity. This can negatively impact model performance. To address this issue, we implemented a weight map approach inspired by the work of Ronneberger et al. [2015]. This approach penalizes errors at the borders of touching cells, improving cell separation. Our implementation involves computing the weights for each object individually and combining them in an additive manner. This generates weights that are higher at the borders of cells and decrease as we move away from them. Specifically, for each cell included inside a segmentation mask,



(a) filaments (b) strip

(c) filaments (d) strip

Figure 2.9: Crops used for the artifacts oversampling.

we compute the distance of each image pixel to the closest pixel of that cell. Then we apply a gaussian kernel on each image pixel using the distance of the pixels as the exponent value. Sigma is set equal to the average cell radius. We repeat this procedure for each binary object representing a cell inside the ground truth. In the end, we sum up all the intermediate images we obtained for each cell to get the final weighted mask. The resulting global weight map has higher values where there is a higher density of cells (as shown in Fig. 2.11). The pseudocode for this weight map is provided in Fig 2.10 and an example weight map is shown in 2.12.

Algorithm 1: weight map pseudocode for j-th mask

Initialize empty map$_j$ (mask size) `// weight map j-th mask`
for *each cell in mask* **do**
  `/* loop over i-th cell in j-th mask */`
  Initialize empty map$_i$ (mask size) `// weight map i-th cell`
  Add i-th cell to map$_i$
  Compute euclidean distance between each pixel of map$_i$ and the closest pixel of the i-th cell
  Compute each pixel's weight in map$_i$ according to a decreasing exponential function:

$$\text{weight} = \exp\left\{\frac{-d^2}{2\sigma^2}\right\} \tag{1}$$

  `/* d is the distance computed at step 5 */`
  `/* σ is a customizable parameter set to 25 (average cell radius) */`
  Sum the resulting map$_i$ to the full map$_j$

Figure 2.10: **Weighted Maps**. Pseudocode for weighted maps

2.3.2 Training

All the competing architectures are trained on the same images and hyperparameters values which are reported in the following. For our experiment, we selected a test set of 70 full-size images taking care to include at least a dozen of significant examples. We inserted pictures representing high-density cells, artifacts, and low background-cell contrast 2.2.3. We used the remaining pictures to create training and validation sets. For these latter sets, we extracted 12 $512x512$ partially overlapping crops from each image for two main reasons. The first motivation regards increasing the batch size providing examples coming from different pictures. With full-size pictures, only a few samples fit the ram before incoming into a memory issue. Smaller crops in-

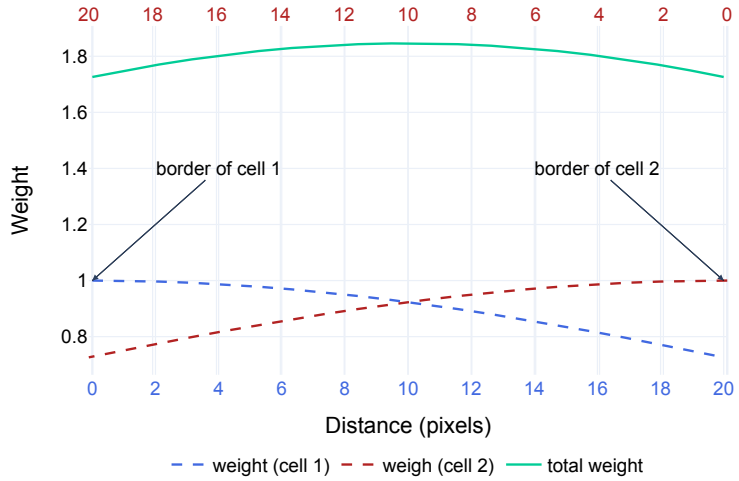


Figure 2.11: **Weighted Maps**. For each cell, are reported the weights generated as a function of the distance from their borders. The above solid green line represent the total amount obtained by adding individual cells' contributions

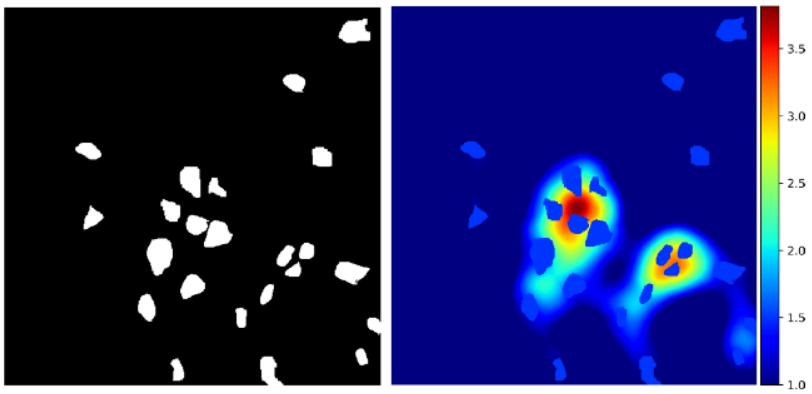


Figure 2.12: **Weighted Maps**. On the left, the mask of a crop representing a crowded area. On the right, the relative weighted maps

stead, help to combine different samples and provide more meaningful training batches. Another reason is related to the augmentation strategy. Indeed, in this way, we can select only a small crop representing the artifact we want to oversample without the need to augment an entire image. After the split and crop operation, we apply an augmentation pipeline performing various transformations, such as rotations, Gaussian noise, brightness adjustments, and elastic deformations. This latter transformation is used to augment the shape and size variability but it requires to be used cautiously since extreme parameter setting can provide cell samples resembling artifacts biasing the model. However, for manually segmented crops, we used an augmentation factor of 10 and 4 for all other images. When artifacts oversampling is applied, 2.3.1, the crops representing the challenging traits are augmented with a factor equal to 100. This resulted in a total of nearly 16,000 images, with 70% used for training and 30% for validation. To ensure a fair comparison, all of the competing models were trained from scratch under the same conditions. The Adam optimizer was used with an initial learning rate of 0.006, scheduling a decrease by 30% of its value each time the validation loss did not improve for five consecutive epochs. A weighted binary cross-entropy loss was employed to handle the imbalance between the two classes, with weights of 1 and 1.5 for cells and background, respectively. Also, for the experiment performed with weighted maps, the map of class weight is multiplied by the weighted map. The early stopping patience is set at a value of 20. Finally, the model relative to the best validation performance was considered for the successive evaluation.

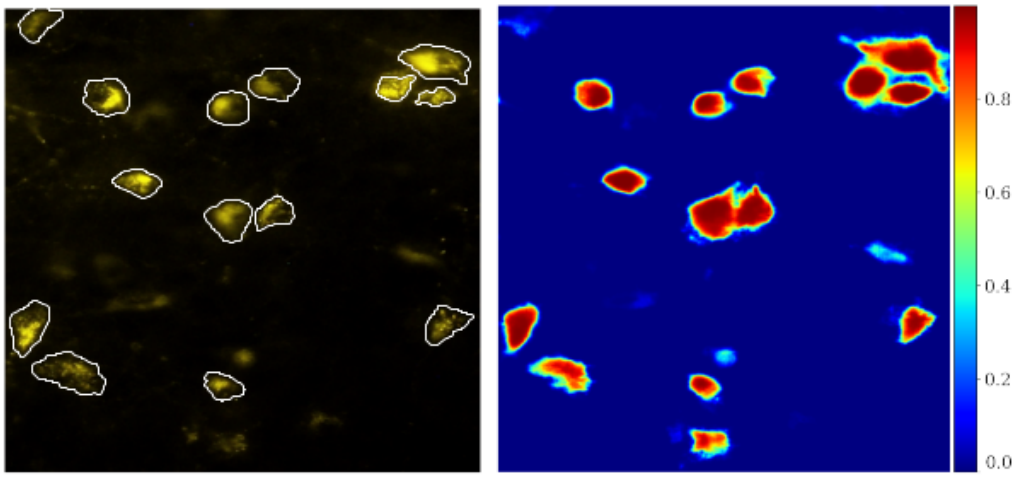
2.3.3 Post-processing

We recall here that the final output of the model is a probability map (heatmap) whose pixels value represents the probability to belong to a cell (Fig. 2.13a). By thresholding operation we then a binary mask, in which groups of white connected pixels represent detected cells. However, these images can present some spurious little connected components that contribute as noise on the final count evaluation. Indeed, during the inference, the model

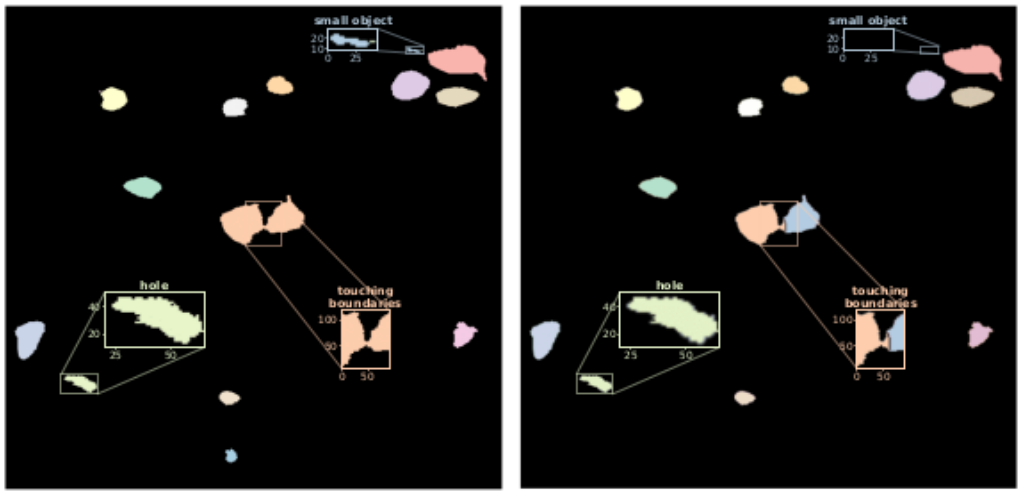
can be fooled by small sizes artifacts (Fig. 2.4) detecting them as cells. Moreover, some cells can be detected only partially, so that in the heatmap they look like halved discontinuous cells. We define a post-processing pipeline to remove these undesired effects and to promote a better count evaluation. Precisely, leveraging the cell shape information (2.2.2, the post-processing provides the first removal of small objects that, by size analysis, have a low likelihood to be fully detected cells. Then, a step to fill the holes inside the detected object is performed to avoid further cell fragmentation during the next step of post-processing. Indeed, the last phase of this processing chain provides the application of the watershed algorithm that is used to detach connected objects. If wrongly parametrized or if the cells contain large holes inside, this algorithm tends to produce over-fragmentation resulting in new undesired objects. An example of the application of this pipeline is reported in Fig. 2.13. In 2.13a we observe the heatmap outcome for the corresponding input image. In the bottom part of the figure 2.13b, we evaluate the effect of the post-processing. We can observe the removal of the small spurious object, the filling of holes, and the final separation effect obtained by using the watershed algorithm. It is worth remembering that the parameters used for the post-processing are customized for this dataset. The morphological cells' information can provide some useful hints but the ultimate test is the visual inspection. For such reason, we promote the development of a web app that, other than the model inference, provides an intermediate step to set the parameters suitable for a specific dataset. Through an interface, it is possible to evaluate in real-time the effect of parameters fine-tuning applied in multiple images.

2.3.4 Model evaluation

The Unet, small Unet, ResUnet, and c-ResUnet architectures, together with the artifacts oversampling and the weight maps design choices, were evaluated in terms of both detection and counting performance. Indeed, despite the detection concerns mainly the segmentation performance, we are not interested in quantifying as much as a model is accurate to draw accurate shapes.



(a) On the left, groud-truth. On the right, heatmap



(b) On the left, thresholded prediction. On the right, post-processed prediction

Figure 2.13: **Post-processing pipeline**. Upper-row pictures, the input image with ground-truth cells' shape overlapped (left) and the model's raw output (right); bottom row (figure from Clissa [2022]). The predicted mask after thresholding (left) and the predicted mask after post-processing (right)

The segmentation is only a mean to our ultimate task which is counting. Nevertheless, it is worth remembering that to enforce the correct number estimation, we need accurate segmentation at least in the crowded area, justifying the adoption of the weighted maps. Concerning the counting task we used the Mean Absolute Error (MAE), Median Absolute Error (MedAE), and Mean Percentage Error (MPE) to test the model performance. Precisely, letting n_{pred} to be the number of the object predicted by the model for the *i-th* image and n_{true} the actual number, then, the **absolute error** (AE) and **percentage error** (PE) are calculated as follows:

$$\text{AE} = |n_{\text{true}} - n_{\text{pred}}|; \tag{2.1}$$

$$\text{PE} = \frac{n_{\text{true}} - n_{\text{pred}}}{n_{\text{true}}}. \tag{2.2}$$

The counting metrics are then the mean and median of the AE and PE just defined. To evaluate the detection ability of the models, instead, we rely on the accuracy, precision, recall, and F_1 score whose definition follows here:

$$\text{accuracy} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} = \frac{1}{1 + \frac{1}{\text{TP}}(\text{FP} + \text{FN})}; \tag{2.3}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}; \tag{2.4}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}; \tag{2.5}$$

$$F_1\text{score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} = \frac{1}{1 + \frac{1}{2\text{TP}}(\text{FP} + \text{FN})}. \tag{2.6}$$

where TP, FP, and FN indicates true positive (cells correctly detected), false positives (cells erroneously detected), and false negatives (cells erroneously missed), respectively. It is worth observing that in the accuracy term, we do not have true negatives since the prediction of our model is at the object level, so there are no *non-cell* objects predicted by the model. However, to quantify the right number of such quantities, we designed an

algorithm able to match the true positives and quantify the number of false negatives and false positives. This algorithm compares each target cell to all objects in the corresponding predicted mask and associates it with the closest match. If the distance between the centroids of these two elements is less than a fixed threshold (60 pixels, corresponding to the average cell diameter), the predicted element is considered a true positive; otherwise, it is classified as a false negative. Elements that are detected but not associated with any target are considered false positives.

Threshold optimization

The optimal cutoff for binarization was based on the detection performance. Indeed, also if the ultimate goal is to accurately count the number of cells, the focus on detection performance helps to ensure accurate recognition and avoid balancing false positives and false negatives in a way that could affect the count. Moreover, the F_1 score is specifically indicated for a heavily unbalanced problem so we select this metric for our evaluation. Before evaluating the model performance on the test set, we need to select the threshold to binarize the heatmap produced by the models. To find this optimal value, we compute the F_1 score using all the training and validation images and varying every time the binarization threshold within a range going from 0.5 to 1. Figure 2.14 shows the optimization results for each model. On the left, we can see how each model's performance varies in the validation set as a function of the cutoff for binarization. Even though lower thresholds work best for all models, the F1 curves are rather flat after their peaks. Thus, increasing the cutoff allows focusing only on predictions whereby the model is very confident, with just a slight loss in overall performance. Also, good practices in natural science applications suggest being conservative with counts and only considering stained cells. For these reasons, we resorted to the *Kneedle* method for the selection of the optimal threshold. An example of that choice in the case of c-ResUnet is reported in Fig. 2.14 on the right plot.

The resulting threshold was then used to evaluate the model's perfor-

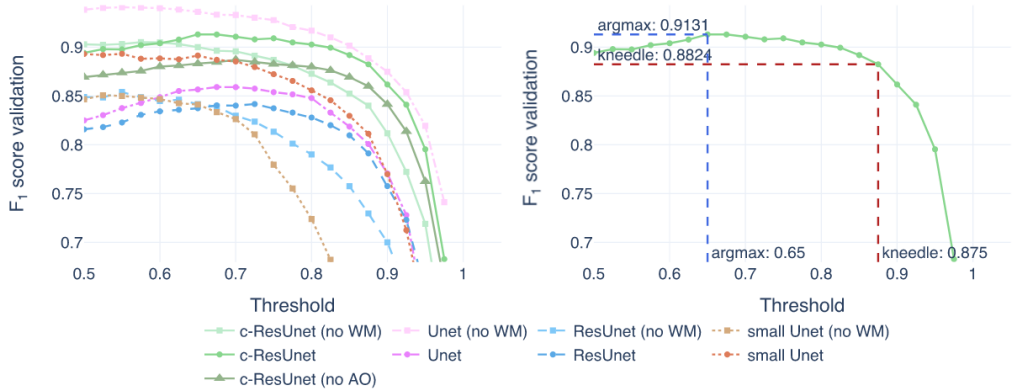


Figure 2.14: **Threshold optimization**. On the left, the F1 score is computed on validation images as a function of the cutoff for thresholding. On the right, the test F1 score of the c-ResUnet model is used to illustrate the selection of the best threshold for binarization according to the *argmax* (blue) and *kneedle* (red) methods.

mance on the test set. Full-size images, rather than crops, were used to better simulate the model's operational condition in a real-world scenario.

2.4 Results

After training, the four neural network architectures were compared in two different scenarios: using all design elements or using only artifacts oversampling (no WM). The c-ResUnet is further investigate adding the case in which no artifacts oversamplig is provided (no AO) but keeping the weighted maps design. The test set consisted of 70 full-size images, and the model's performance was evaluated in terms of both detection and counting ability. The results of this comparison are provided in Table 2.1.

Model	Threshold	F_1	Accuracy	Precision	Recall	R^2	MAE	MedAE	MPE (%)
c-ResUnet	**0.875**	**0.8149**	**0.6877**	0.9081	**0.7391**	**0.8215**	**3.0857**	**1.0**	-5.13
c-ResUnet (no AO)	0.875	0.8047	0.6732	0.9019	0.7264	0.8077	3.0857	1.5	-6.24
c-ResUnet (no WM)	0.875	0.7613	0.6147	0.9418	0.6389	0.7048	3.6857	**1.0**	-19.14
ResUnet	0.850	0.7855	0.6468	0.8865	0.7052	0.7831	3.3286	**1.0**	**-4.84**
ResUnet (no WM)	0.850	0.7513	0.6016	0.9387	0.6262	0.6955	4.0571	2.0	-24.12
Unet	0.875	0.7724	0.6291	0.9117	0.6700	0.7560	3.5143	1.5	-14.36
Unet (no WM)	0.850	0.7886	0.6510	0.8989	0.7024	0.8069	3.1571	2.0	-9.23
small Unet	0.875	0.7563	0.6081	0.9264	0.6389	0.7682	3.5714	2.0	-21.37
small Unet (no WM)	0.825	0.6697	0.5034	**0.9483**	0.5176	0.5723	4.7714	2.0	-32.01

Table 2.1: **Performance metrics.** Test set performance using the optimal *kneed* threshold. Both detection (first four columns) and the counting (latter three columsn) evaluation are reported.

2.4.1 Performance

Based on the F_1 score and MAE, the c-ResUnet architecture outperforms the other models by a consistent margin. The best ResUnet model stays three F_1 percentage points behind the best c-ResUnet results associated with the weighted-maps adoption. The best Unet design configuration also is significantly behind the c-ResUnet despite having significantly more parameters (nearly 14 million compared to 1.7). Moreover, the ResUnet model family outperforms the smaller Unet model, which has a similar number of parameters (876,000). The c-ResUnet model performs well also according to the other evaluation metrics. The only result that reflects a countertrend result concerns the precision associated with the Unet models that are higher than the c-ResUnet one. This may be due to a tendency towards *overdetection* in the Unet models that is explained by its optimal lower threshold value. This situation allows a higher number of cells to be detected at the expense of the recall value that drops below nearly 0.52. It is worth noting that the use of the optimal threshold determined by the Kneedle method results in high cutoffs and only includes detections with high confidence leading to an increase in false negatives. As a result, the accuracy suffers, as the impact of false negatives is twice as much as it is in the F_1 score. Overall, the model provides reliable predictions and satisfies the design requirement of being conservative with cell counts, as indicated by the negative values of MPE in

all experimental conditions.

2.4.2 Design evaluation

In this section, we focus on the evaluation of the two different experimental designs. After the first training round including the weighted maps in the loss term, we repeat the experiment for all the models under the same condition but switching off this loss factor. For the c-ResUnet, we also test the artifact oversampling strategy effect.

From 2.1 it seems clear that the weighted maps sort out a positive effect since all the models achieve better results both in terms of detection and counting performance with the only exception of the Unet model in its heavier version (17 million of parameters). We design these penalization maps to enforce the separation between touching boundaries helping the model to segment better the high-density cell areas. To visualize such kind of effect we report a comparison 2.15 between the heatmaps produced by the c-ResUnet trained with and without the weighted map. We can notice clear improvements in the boundaries definition when the model is trained using such maps that definitively help to separate better the touching objects.

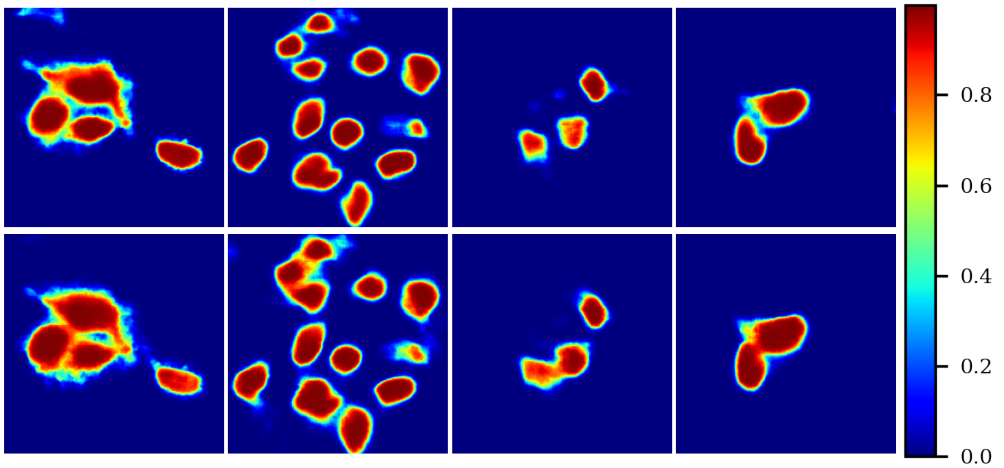


Figure 2.15: **Weighted Maps effect**. Predicted heatmaps obtained with c-ResUnet (top row) and c-ResUnet (no WM).

2.4.3 Results visualization

The visual inspection can help to define some qualitative aspects of the inference phase. Using the bounding boxes visualization we can remark on some model behaviors. We start with a picture that shows the tendency of c-ResUnet to be sometimes too conservative missing a significant number of cells. However, from the same picture, we can note that false negatives often look very similar to the other cells lying in the background that were discarded also by the human operators. It is difficult to judge these borderline cases and so we cannot make a definitive statement on the quality of this model decision. Moreover, in other cases, like in Fig. 2.16 we can observe the opposite tendency that overall balances with the previous behavior. Moreover, the false positives reported in this figure this time strongly resemble the actual stained cells present in the same picture. Again, it is difficult to judge which from the human operator and the model makes the wrong decision. Last, in the Fig. 2.16 we observe these two opposite tendencies balanced in the same images, as a result, the predicted count is close to the actual one.

2.5 Final remarks

The proposed approach, which uses Deep Learning techniques to automate the process of counting cells in fluorescent microscopy images, proved to be successful in our study. Of the four CNN architectures we evaluated, the cell ResUnet (c-ResUnet) was found to be the most effective. Despite having seven times fewer parameters than the original ResUnet, the c-ResUnet model performed better due to the inclusion of a learned colorspace transformation and a bottleneck with two blocks including 5 x 5 filters used to enlarge the field-of-view. In our ablation studies, we found that a weight map penalizing the errors on cell boundaries and crowded areas contributed to improved performance. The performance boost is due to the regularization factor effect in the loss term and the better cell separation learned from the model that implies better detection, avoiding to count multiple clustered cells once. We don't expect the artifact oversampling to result in a higher metric

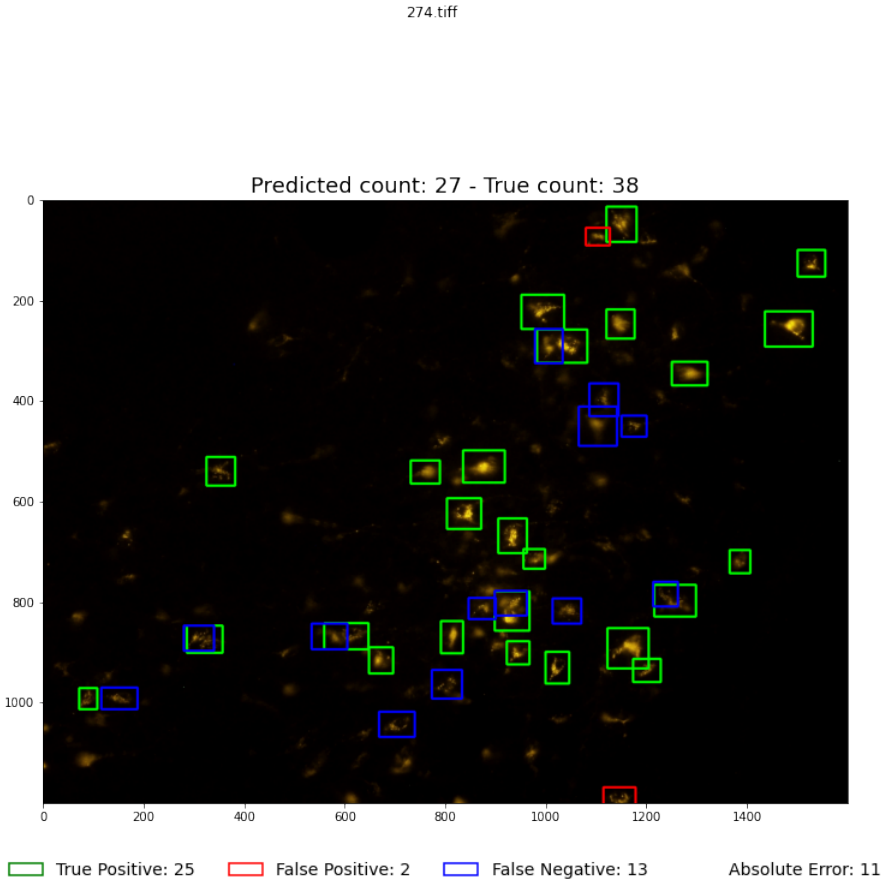


Figure 2.16: **Model Predicion**. Bounding boxes visualization of a picture where c-ResUnet produce a significative number of false positive.

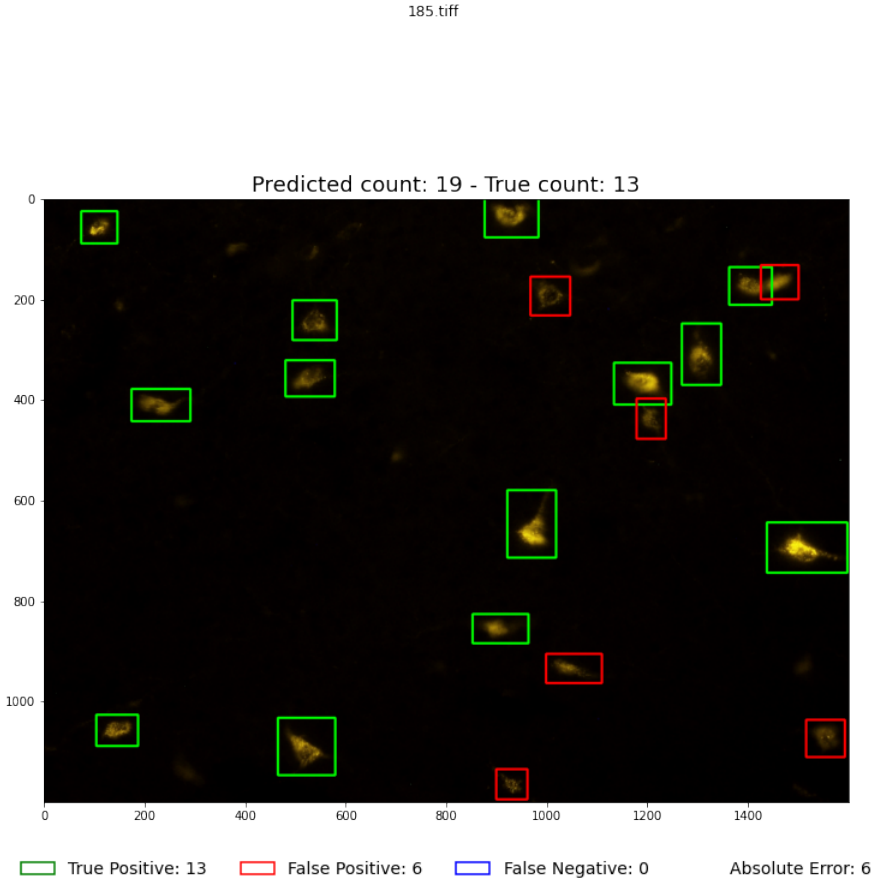


Figure 2.16: **Model Predicion**. Bounding boxes visualization of a picture where c-ResUnet produce a significative number of false negative.

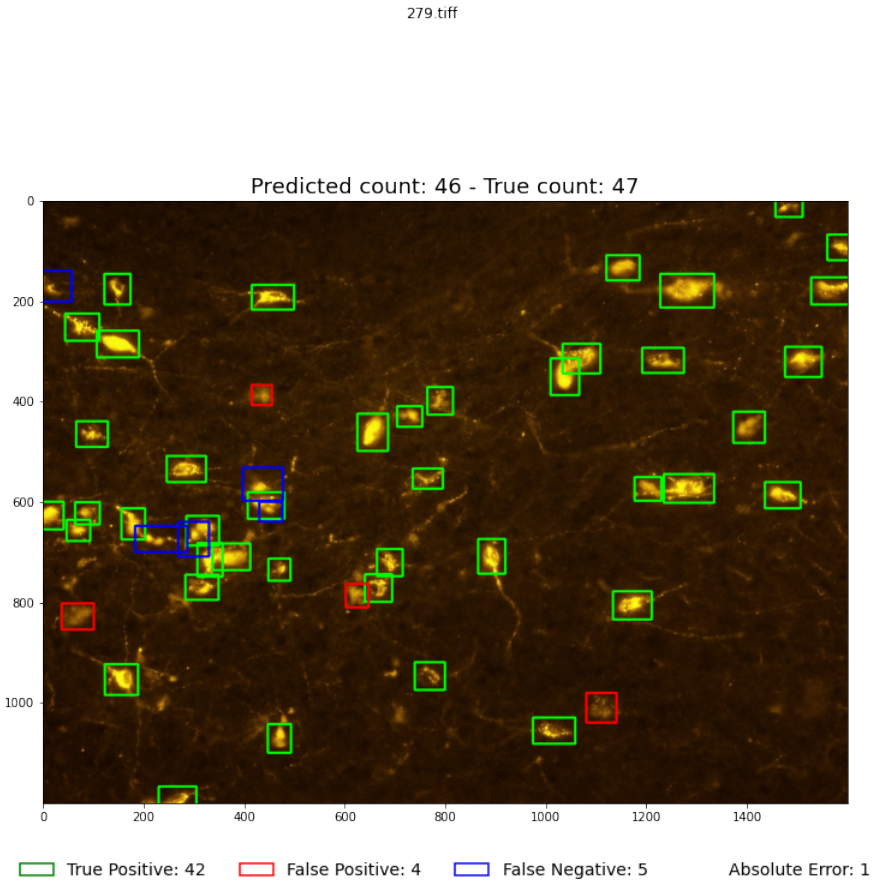


Figure 2.16: **Model Predicion**. Bounding boxes visualization of a picture where c-ResUnet balances its tendency to produce a false positive and false negative. As a result, the predicted count is close to the actual one.

score since it is only a tool to better recognize the artifacts. However, if the artifacts are already well represented, the model should learn to recognize them so making the artifact oversampling unnecessary. On the other hand, if they are underrepresented, this strategy can be useful, but, given the low number of these examples in the dataset, clear evidence in the metrics may not subsist. In terms of overall performance, our model had a mean absolute error of 3.0857 and a negative mean percentage error of -0.0513. The F1 score of 0.8149 indicated that the model was able to accurately detect cells, rather than achieving good counts through a balance of false positives and false negatives. Experts who visually inspected the predictions found that even the erroneous detections were reasonable and subject to subjective interpretation. We hope that by sharing the c-ResUnet model and annotated data, we can encourage further research and applications in this field and related areas, and bring significant advantages in terms of speed and reduced operator bias to experiments involving microscopic fluorescence.

2.6 Future works

The field of cell counting has been rapidly growing in recent years as algorithms have become more accurate and efficient, allowing researchers to focus more on the relevant aspects of their research. However, there are a few key challenges that need to be addressed to fully mature this methodology. The first challenge is the demand for supervised data. The process of producing sufficient ground truth segmentation masks can be quite time-consuming and may even be prohibitive in some cases. To address this issue, researchers are exploring self-supervised and semi-supervised methods that do not require as much labeled data, particularly in the field of segmentation. Additionally, methods that use labels that are not directly related to the segmentation output but are still useful for counting (weakly-supervision) could also be investigated. The other key challenge is the reliability and explainability of predictions. Deep learning models still struggle to provide clear explanations for their decision-making processes, which is a critical requirement in the biological field. Improving explainability would not only increase the perfor-

mance of these models but also justify their decisions, making them more trustworthy. Finally, due to the unbalanced problem, we aim to address this issue by resorting to an oversampling strategy or the adoption of a different loss, like the focal loss one.

Chapter 3

Weakly supervised approach

3.1 Introduction

Weakly supervised methods are a set of models that try to learn with weak supervision. In the case of semantic segmentation, this means that we can't rely on pixel-wise ground truth. Again, we restort the counting-by-segmentaion approach to localize the cells taken into account during the counting also in this case. Several approaches can be tried depending on the available labels to accomplish with the segmentation task and some of these are described in the following paragraphs.

Image-level caption

Image-level labels identify the categories of the objects present in a scene. They don't provide full supervision for the segmentation purpose but can be exploited to generate pseudo-labels. The first step is to train a model to associate the images with the correct labels, that is, to accomplish a classification task. During the training the model focus on the relevant areas of the scene that helps the network to understand the classification problems. These areas generally correspond to the ones in which the objects relative relative to the correct label are included. After that, it is possible to produce a sort of activation maps (class activation maps - CAM) that highlight these areas. By weighting the feature map of the last convolutional layer

of a CNN and upsampling the weighted map to the image size, the authors can begin to localize which areas of an image correspond to different classes. This is possible by introducing a global pool averaging layer to the classification network and examining the activations of the final convolutional layer in the classification CNN. From this heatmap, it is possible to produce a segmentation mask by thresholding. The outcome is a pseudo-label for the segmentation task. These images usually are not so accurate so some other steps to refine them are necessary. To enhance the generated mask segmentation, different methods have been proposed. Kolesnikov and Lampert [2016] Introduce three different loss functions to enforce the network to expand or constrain the boundary of the objects leveraging the color and the spatial structure of the images. Another frequently used method is known as **CRF** which instead exploits the local information and global context to produce a better localization of the objects Oh et al. [2017] Xu et al. [2021].

Bounding-box

Bounding boxes provide information about the objects localization. The idea is to use this information to localize the foreground extraction from within these coordinates. For example Rother et al. [2004] used an optimized version of graph-cut algorithm that use texture (colour) information, and edge (contrast) information, to segment the object of interest. Also, the same CRF algorithm can be use jointly with an expectation maximization (EM) algorithm to generate enhanced pseudo masks Papandreou et al. [2015]

In constradt to the classical methods, the autorh of Dai et al. [2015] promoted the combination of region proposal algorithm and convolutional neural network training. These two steps alternate to gradually recover segmentation masks for improving the networks, and vise versa requiring only easily obtained bounding box annotations. On the same vein, in Lee et al. [2021] is developed a new effective way for the region-proposal step introducing a bounding-box attribution map (BBAM), which identifies the target object in its bounding box.

Dot annotations

Several works exploit point annotations and this is because it offers several advantages Papadopoulos et al. [2017a]

- is substantially faster than drawing bounding boxes;
- It requires little instructions or annotator training com- pared to drawing;
- It requires no specialized hardware

Recalling the CAM method described above, a further development consists of using also the dot annotation to make a point-class-activation-maps McEver and Manjunath [2020]. Indeed, other than the classic map activation related to the classification problems, point supervision is included in a second loss term. This work promotes a better localization of the class activation maps. Then a further CNN refine the pseudo-labels obtained propagating the affinity between neighboring pixels. After this further step, a CNN can be trained in a fully supervised manner exploiting the ground-truth labels generated in this way.

Another interesting work Bearman et al. [2016] promotes the joint use of one-click supervision with the objectiveness concept, that is, the probability that a pixel belongs to an object. In this way, the lack of knowledge doesn't cover by a point annotation, that is, the extent of the object is alleviated by introducing this prior directly in the loss term.

Another concept introduced is that of extreme point Papadopoulos et al. [2017b]. Differently from a bounding box approach, the annotator is asked to click on four physical points on the object: the top, bottom, left- and right-most points. From these four points is still possible to get a bounding box and at the same time, to know the extent of the object inside it. So, more information is retrieved in less time to produce better results. This approach so can be used both for object detection and segmentation. Indeed, a gaussian is placed on top of these points and the resulting image is added to the RGB input picture. From this image, the network learns to segment the object inside the four extreme points.

Our work is based on the dot-annotations availability. In the following sections, we first introduce one of the two datasets used for this study, while the other is the same already considerede on the previous chapter. Then, we articulate two different procedure to produce pseudo-labels enabling the segmentation approach. Indeed, our approach is still based on the counting-by-segmentation approach to provide a precise localization of the object counted also if not interested to provid a fine object segmentation. For the training of the model and the successive evaluation we resume the same methodology exposed during the supervised case analysis. Result and conclusion remark the validity of the described approaches.

3.1.1 Related works

The term weak supervision related to the segmentation task includes several labels kind as bounding boxes, dot annotations, scribble, and image-level caption. Many works address the usage of these labels to perform such a task in different research domains using only this limited information. With respect to the cell counting tasks, some labels are more suitable than others. For example, drawing bounding boxes around highly packed cells is not easy with the chance to pack multiple cells or to include a relevant portion of the surrounding background inside the same box. This is especially true when the cells have an extremely variable and elongated shape like those depicted inside the fluorescent neuronal cells dataset. In some cases, this is still a feasible option to save labeling time as demonstrated in Khalid et al. [2022]. In this paper, the authors promote a method base on a multi-step annotation process. During the first step, the annotators draw the bounding boxes around the cells. Then, some points are sampled from these boxes and the human annotators select which of those points are cells or backgrounds. Other methods Zhou et al. [2018] Li et al. [2018] Faster [2015] are successfully applied but only in well-defined conditions: the objects don't touch each other and don't have complex rigid structures and, usually, this is not the case for the neuronal cells.

On the other hand, point annotations represent a good choice in terms

of preserving information and saving annotation effort. Several works focus specifically on the dot annotation Nishimura et al. [2019] Qu et al. [2020] Khalid et al. [2022] trying to design a general framework to apply across different datasets going from histopathology to microscopy images. In Qu et al. [2020] a pipeline is developed to gradually refine the segmentation result of a CNN trained on partial dot annotations. The first step provides a detector whose aim is to find the center of the cells in the images. In these pictures, only a part of the cells is annotated. On top of each cell is placed a gaussian probability distribution. While the pixels under this distribution belong to the cell, the others immediately outside are labeled as background. The remaining pixels are excluded from the training. After the first round of training, based on the output of the detector, the authors select more background examples. The masks are so refined step-by-step to provide even more accurate segmentation masks. After these training steps, the Voronoi graph and k-mean clustering are used to separate the touching cells. A new model is then trained from scratch on the resulting masks. A different approach is described in Nishimura et al. [2019]. Here, the dot annotation is used to build a cell centroid likelihood map similar to the work described before Qu et al. [2020]. The detector learns to localize the center of the cell first, then, a backpropagation path is applied to localize the pixels around the center associated with the detection of the cell. Graph-cut iterative procedure is used to refine the segmentation results. Instead, in the work Khalid et al. [2022] an approach using a mix of dot annotations and bounding boxes is described. Here, through a region-proposal method, detection and segmentation of the cells are accomplished jointly. Also, Chamanzar and Nie [2020] develop an algorithm to perform both single-cell detection and segmentation using only point labels. This is achieved through the combination of different task-orientated point-label encoding methods and a multi-task scheduler for training.

3.1.2 Contribution

This work describes two different approaches to dealing with point annotations for to perform count-by-segmentation task. Since the segmentation is a means to provide the localization of the cells included into the counting process, to evaluate these methods we use metrics concerning counting/detection performance. Our contribution is twofold:

- We provide a method for fast and easy training suitable when the cells have a regular and homogeneous shape
- We furthet develop this method to perform the detection-by-segmentation task also when the shape of the cells are more complex presenting a large variety of shapes and sizes.

In both cases the contribution is focused on the binary masks generation used for the model training. in the former method we levereage on the features map obtained from an autoconder trained on a recontruction pretex-task. Then, we apply the weakly dot annotations information to refine the previous results and remove the undesidered objects. Where possible we compare the weakly trained model with its supervised couterpart to validate the method proposed. The code used for both the use case presented in this chapeter are reported to the following links:

- CTb dataset: (cell counting yellow supervised and weakly supervised approach for CTb dataset)
- c-Fos dataset: (weakly supervised cell segmentation c-Fos dataset)

3.2 c-FOS Dataset

The dataset is composed of 251 images with a resolution of 1200x1600. The pictures are acquired using a similar methodology exposed in the previous chapter but exploiting a different fluorochrome emission. Indeed, the microscopy configuration adopted exploits absorption (excitation) of a wider

emission wavelength range, going from 495 to 570 nanometers. As a result, the images have a greenish appearance in contrast with the yellowish coloration of the fluorescent cells dataset. To acquire these images, the same monosynaptic retrograde tracer (b-subunit of Cholera Toxin, CTb) is injected into the raphe pallidum which is thought to be involved in the regulation of the torpor onset. Also, the brain regions involved in the experiment are the same, involving the dorsomedial hypothalamic nucleus (DM), the lateral hypothalamic area (LH), and the ventrolateral part of the periaqueductal gray matter (VLPAG). From this area, the activity of the neurons is observed through tracer signals to understand the functional connections between brain regions Hitrec et al. [2019]. After some time, about a week, the brain is cut into slices and analyzed using the fluorescence microscope.

3.2.1 Data description

c-fos

The appearence of the images is drastically different from the images belonging to the yellow dataset altought deriving from the same experiment. Firstly, the hue is higly shifted on the green channel how it is visibile from figures 3.1. The stained cells look like bright circular shape surrounded by homogeneus greenish biological tissue. In figure is reported a zoom detail highlighint such a stained cells 3.2 with bounding boxes to help their visualization. To summarize, the most relevant characteristcs of this dataset are:

- High resolution images: 1200x1600 pixels
- High value on green channel both for background and cells pixels
- Low contrast between the background the the cells
- Reduced impact of artifacts respect to the fluocells dataset
- Quite homogeneus size and shape

cTB

A description of this dataset is remainded to the previous part. Here we only summarize the relevant features in comparison with the c-fos dataset.

- High resolution images: 1200x1600 pixels
- High value on green and red channel
- Considerable contrast between the background and the cells in most part of the images
- Presence of characteristic artifacts
- Large variability in size and shape

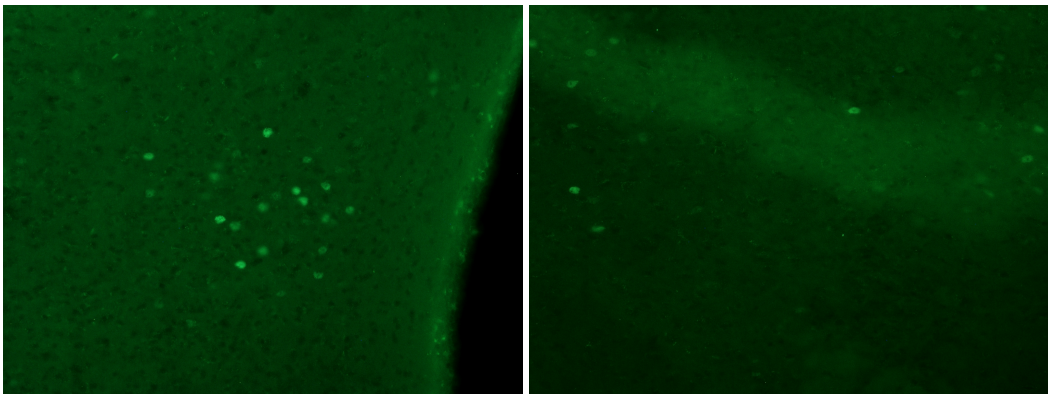


Figure 3.1: **Sample data.** Raw image. Some peculiar samples. In the top left corner a typical sample image. On the top right an image with an artifact related to the fluorescence emission process: the green light agglomerate is not a stained cell to count. In the left bottom corner, image with stripe. The small bright spots can fool the model during the count. On the bottom right, an example of image with darker background.

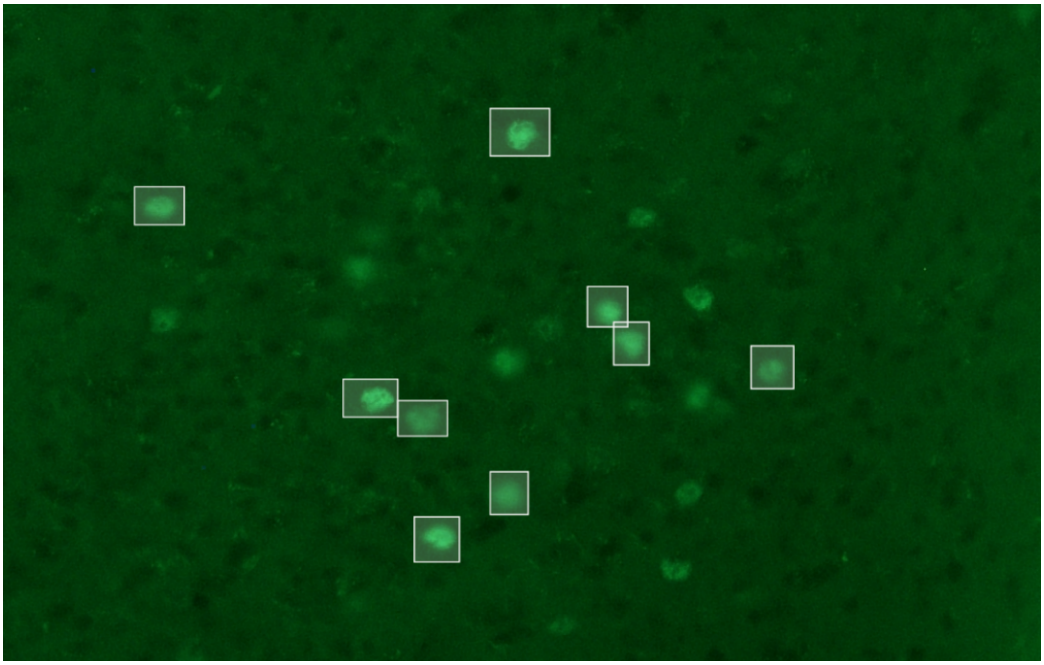


Figure 3.2: Stained cells. The cells appear like circular bright spots. In this picture some examples are highlighted under the white bounding boxes.

3.2.2 Challenges

In this section, we describe the critical traits of the images that mainly influence the counting task process. Indeed, similarly to the yellow fluorescent cells, some peculiar aspects of the images can fool the network during the training phase affecting its resulting generalization performance.

Altought the impact of these traits is reduced considering the yellow dataset, there are still some critical aspects that are worth highlighting. One first salient feature is the reduced contrast between the background and the cells that makes the detection task harder (Fig. 3.3). Indeed, the activated cells usually appear as distinct green bright circular spots but sometimes they are almost indistinguishible from the sourrounding background. This circumnstance happens when the cells are out of the microscope focus and, also if stained, they show a very weak intensity. In figure 3.4 we provide a zoom detail togheter with bounding boxes to help the visualization of such kind of cells distinguishing between in-focus and out-of-focus cells. This

situation complicates the annotator works that can be fooled and make wrong assumption annotating also fake objects (see section 3.3.1).

Another trait that sometimes generate confusion arise from some small bright spots localized in some specific area of the brain tissue. An example of these fake activated cells is depicted in 3.5 around the edge close to the black background area. The bright spots localized in these areas look like the other stained cells but they don't provide any hint about the practical biological aspects and they are not to be confused with the actual activated cells. Luckily, these spots, appear in some specific areas of the biological tissue surrounded by completely black areas 3.5. This peculiar feature is an hint that the model can use to avoid counting them. Lastly, a remark on some artifacts that appear as a side-effect of the fluorescence process and also can be recognized as the other stained cells 3.5.

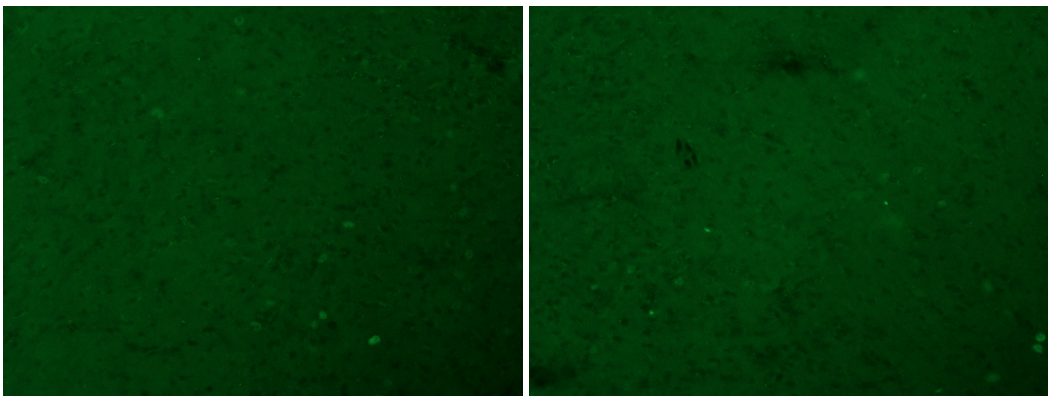


Figure 3.3: **Sample data**. Low contrast image. Here the difference in intensity between some stained cells and the backround is very limited.

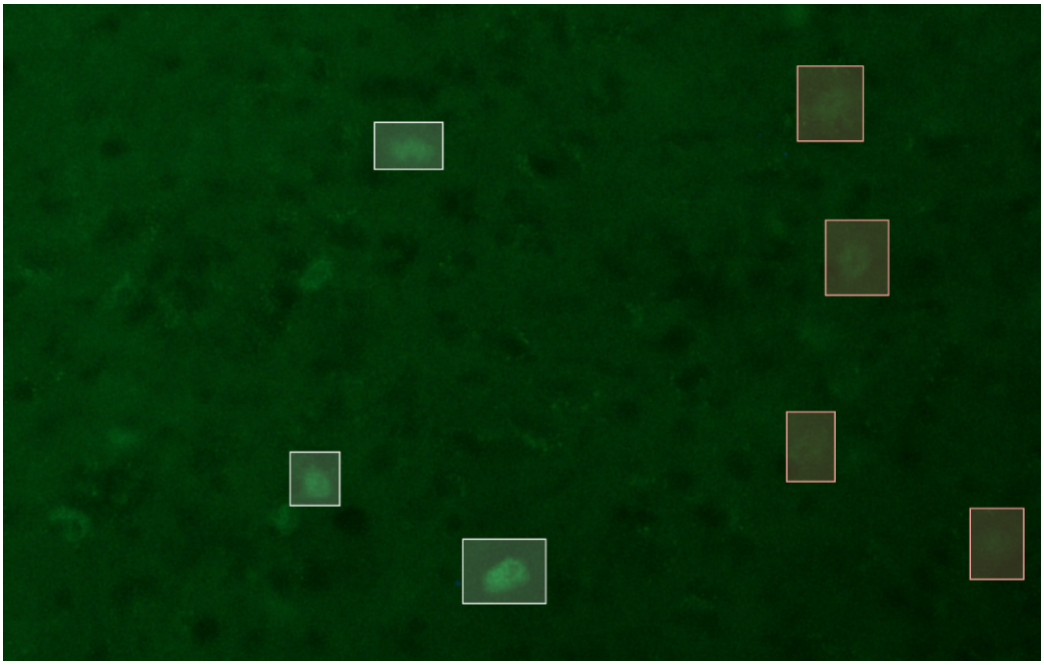


Figure 3.4: Stained cells. The cells usually appear like circular bright spots but some of them have a very weak intensity. In this picture in focus cells and out of focus cells are surrounded respectively by withe boxes and a red boxes.

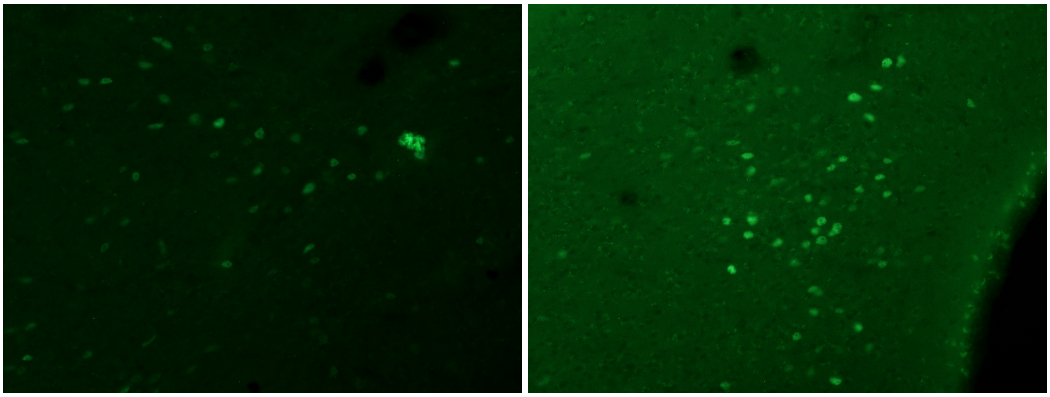


Figure 3.5: **Sample data.** Some challenging traits. On the left an artifact example. The big bright spot look like an agglomerate of cells. On the right, a strip with a large number of bright spot. These elements also if look similar to the cells are not relevant to the analysis.

3.2.3 Dot annotations

The dot annotations represent a set of coordinates that identify the cells activated during the onset of the torpor mechanism Hitrec et al. [2021] Cerri et al. [2021] Tinganelli et al. [2019] Dentico et al. [2009]. Like in the yellow dataset case, the cell activation is traceable to the intensity level of the fluorochrome emission that makes some cells look more stained than others. While the c-Fos images are provided directly with dot annotations, the CTb pictures have their ground-truth mask associated. So, to achieve the dot annotations, we recover the center point of each segmented object represented inside each ground-truth image. Then, the dot annotations are the information used for both the pseudo label generation methods used in the next section (3.3.1).

c-Fos

The dot annotations provided report only the spatial coordinates of the centers of the cells which undergone activation during the experiments. However, also if less stressful with respect to drawing a binary mask around each cell, also this annotation task can be affected by a different source of errors:

- Fatigue of the Operators
- Subjectivity of the operators' choices
- Low accuracy in the coordinates identification

The first two sources of errors can imply a wrong count result and, consequentially, an incorrect evaluation of experimental evidence. The third source of uncertainties refers to the selection of a point outside the cell area. This error has no consequences for the counting task but can affect the *pseudo-labels* generation 3.3.1 since the boolean mask is placed on the wrong picture points. To avoid such a bias affecting the model during the training it is better to define a strategy to adjust these coordinates. However, by a visual inspection, the number of such cases is not relevant and we decide to the left to the model to handle this source of uncertainties.

cTB

The cTB or fluorescent neuronal cells dataset is described already in the first part of the thesis (Chapter 2.2) where we remind for a detailed explanation of the original ground truth provided. We only recall that those binary masks are carried out using both a semi-automatic and a manual labor-intensive approach. So, the dot annotations we recover from these target masks are considered quite reliable since provided by domain experts that image-by-image regard all the cells candidate excluding those wrongly annotated during a first semi-automatic round. On the other hand, the hand-labeled process also represents a task during which the annotator stays constantly focused, thus limiting the propensity to click also on the false positive objects like happened for some of the cFos images.

3.3 Methods

3.3.1 Pseudo labels

We recall that the cells represented in these two datasets have significant dissimilarities, from the color to the shape. The c-Fos pictures depict cells characterized by a quite homogenous shape Section 3.2.1 resembling a circular structure. On the other side, the cells depicted in the CTb dataset show a large variety of shapes and dimensions. From these observations, we decide to pursue two different approaches during the pseudo-labels generation process, one addressing the c-Fos simplified case, and the other facing a more general and complex situation. In the former case, we have a fast and easy approach while in the latter we resume a pretext task as a preliminary step of the pseudo-labels generation. The pseudo-labels generation represents the critical part of the methodology described in the following. The remaining part of the processing pipeline largely resumes the methodology exposed in the previous chapter (Chapter 2)

C-fos

The first step of pseudo-labels generation is the shape and dimension definition of the boolean structure to place on top of the dot annotations. After a visual analysis, we use a circular shape with a radius of 12 pixels. With such a structure, we assess that most of the pixels marked as cells are contained inside the cell boundaries. The steps of mask generation consist first of the creation of a completely black image (target). Then, the pre-defined boolean structure is placed on top of each cell's annotation. Some example results are reported in fig 3.6b.

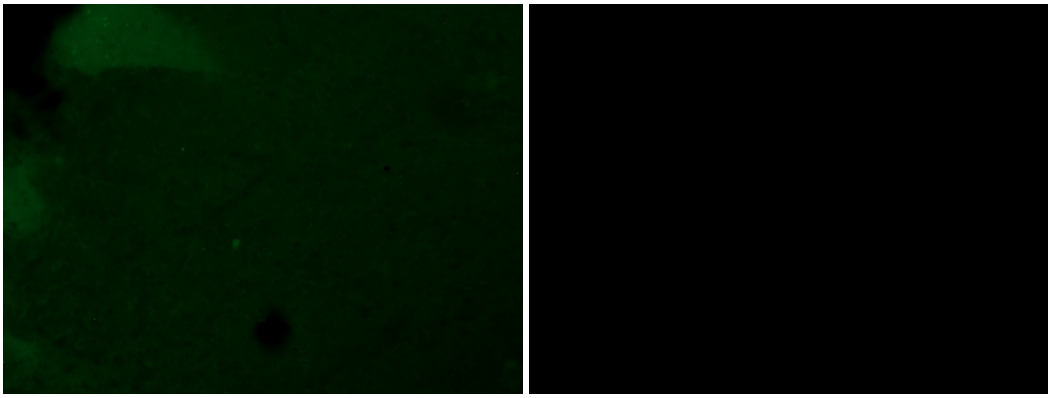
Challenges

After the mask generation, an assessment of the goodness of the mask was qualitatively performed by visual analysis. From this observation we noticed, supported by the domain expert, some labels inconsistency. Especially, the following problems were detected:

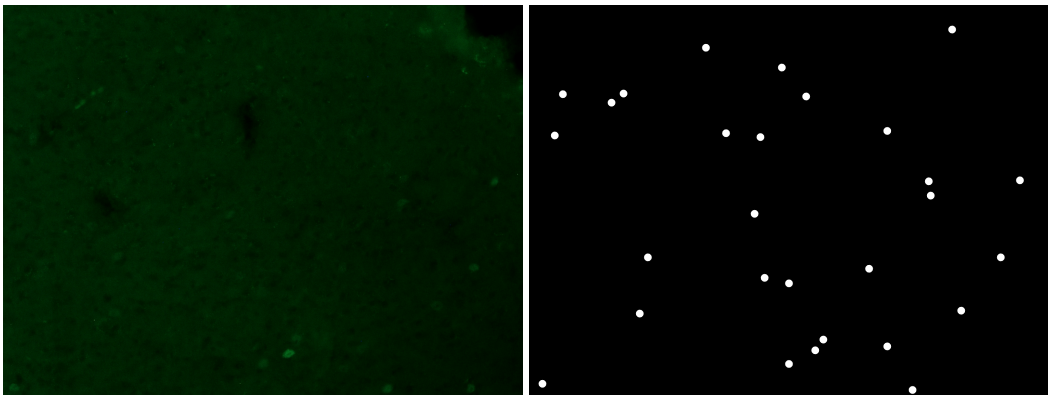
1. Artifacts labeled as cells
2. Wrong annotations

An example of the first issue is provided in figure 3.7 where each cell is overlapped with the contour of the corresponding boolean structure inside the pseudo-labels. From the overlap of the dot annotated red circles, we can observe two things. First, some annotations are slightly far away from the cells. Second, a bigger blob on the bottom of the images is wrongly labeled as a cell. Also if the impact of such an error is reduced, this figure resalts how human error can easily affect the counting task. Deserve more attention to the second type of error from the above list whose examples are reported in figure 3.8. This misjudgment can occur because of the subjectivity or the fatigue of the annotator that selects, wrongly, fake cells. For example, in left picture of thi figure, fake cells are selected from the strip on the side of the image. The red overlapping circles are enlarged to enable better visualization of the underlying cells. Also, in figure (3.9), the operator selected some cells

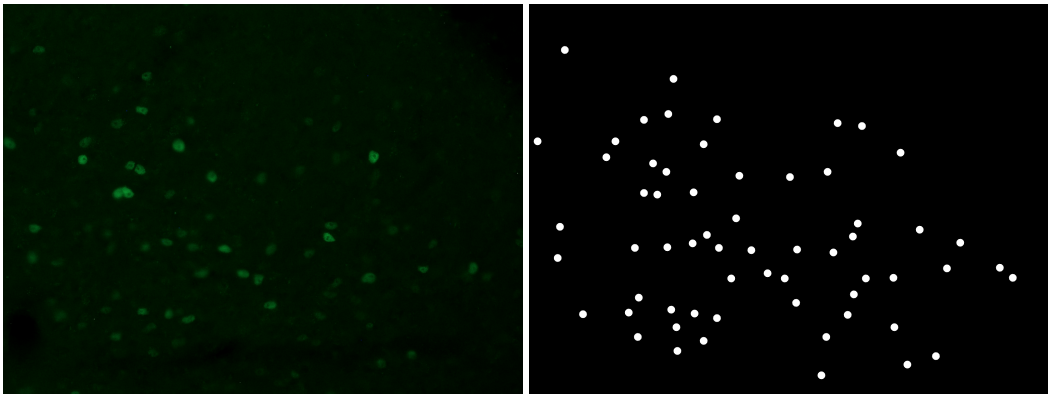
whose fluorescence intensity is very low and that from a posteriori analysis were defined as no-activated cells. These images report the examples already analyzed in figure 3.4 concerning the in-focus and out-of-focus cells. These latter objects, also characterized by weak fluorescence, can be considered as activated cells. However, we include these images for the model train to test the model robustness in a real-case scenario.



(a) image (b) generated mask



(c) image (d) generated mask



(e) image (f) generated mask

Figure 3.6: Sample data.

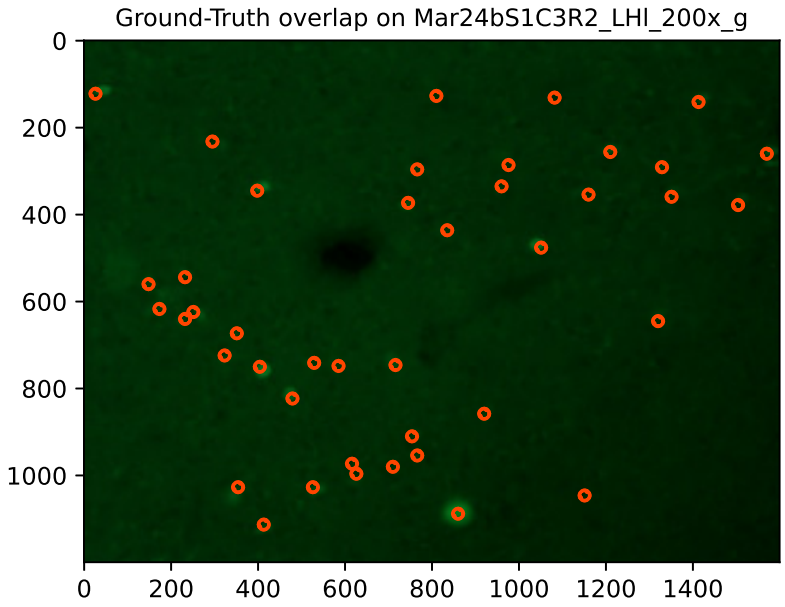


Figure 3.7: Big spot on the bottom of the image is wrongly annotated.

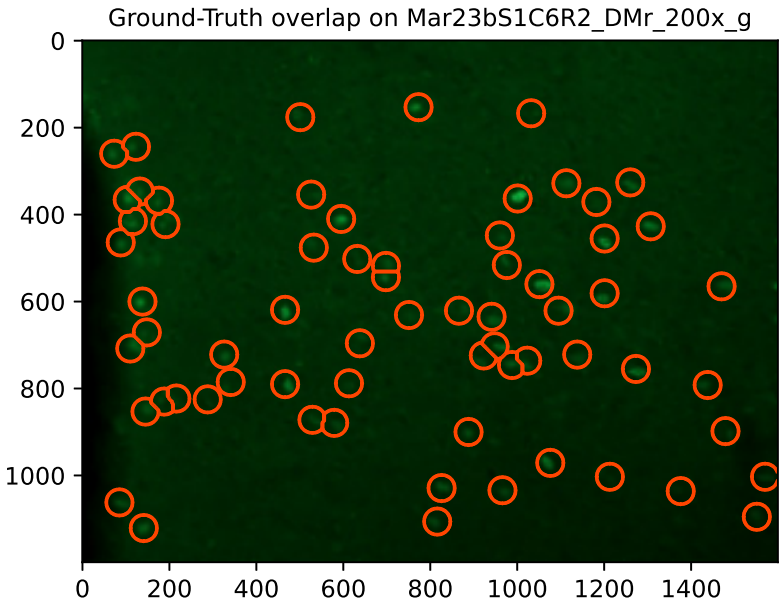


Figure 3.8: **Groud-truth overlap**. Wrongly annotated data. On the left side of the image over the edge some points are wrongly annotated. The circles size are exhagerrated to enable a better visualization of the underlying cells.

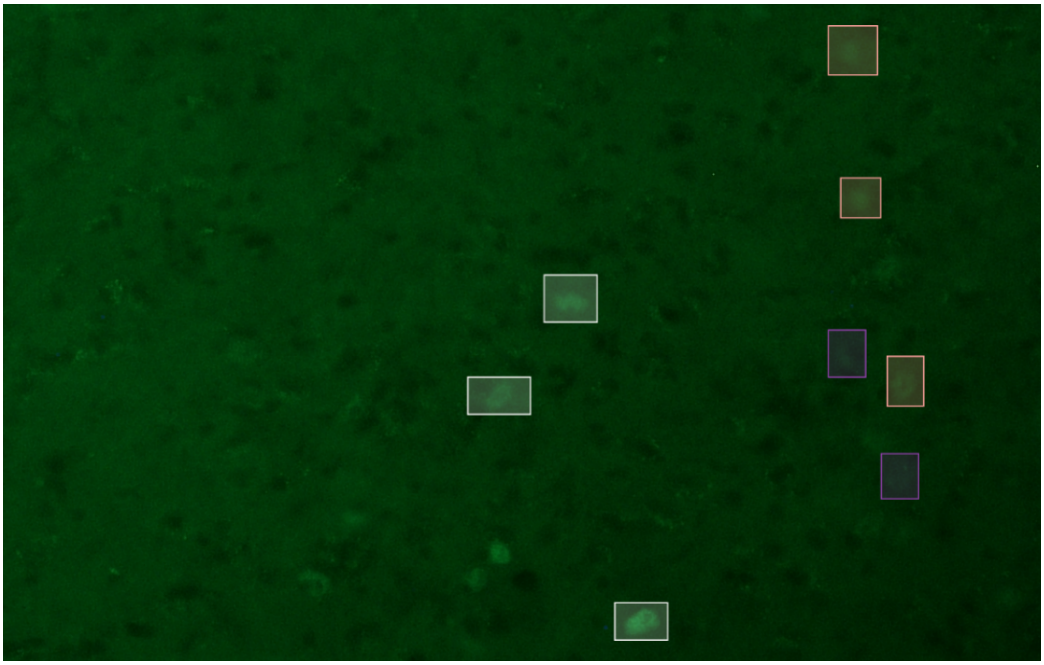


Figure 3.9: **Wrog annotated cells**.The operator wrongly annotated some very low intensity spots hinghighted with purple boxes in the image. For comparison are reported also in-focuse cells and out-of-focus cells.

CTb

Fluorescent microscopy images can depict many biological structures, and the cells shown in these images can have different characteristics depending on the experimental conditions and the biological process being studied. The c-fos dataset analyzed nuclei of neuronal cells that are relatively uniform in shape and size, while the cTB dataset required a more robust method for generating pseudo-labels due to the varied shapes of the cells being detected. The method involves several steps. First, we train an autoencoder on a pretext task like image reconstruction. Then, we use the features maps of the trained autoencoder to obtain a raw segmentation mask, based on the assumption that a few of the features maps can roughly separate cell objects from the background. A threshold is applied to create raw binary pseudo-labels. However, these images contained many artifacts and false positives, so we used dot annotations that only mark actual cells to clean the images. We

created an auxiliary mask with a circular boolean shape centered on each dot annotation and then multiplied the first raw image with the boolean mask to keep only objects that intersected at least partially with the boolean objects of the auxiliary mask. This gave us clean pseudo-labels. The steps can be summarized as follows:

1. Training of the convolutional autoencoder on the reconstruction image pretext-task
2. Selection of a discriminative feature map
3. Application of all model filters until the convolutional layer selected to the previous selection
4. Normalization and thresholding of the output obtained at step 3
5. Generation of a boolean mask
6. Multiplication of the raw mask obtained at step 4 and boolean mask created at step 5

The most critical step is the selection of the feature maps to discriminate between background and cells that, so far, remain a step to be defined manually.

Image reconstruction task

The first phase to consider is training a convolutional autoencoder for image reconstruction. The architecture is similar to the c-ResUnet, but the last layer is replaced with a 3-channel layer to enable RGB image reconstruction. We assume that the network, by learning how to reconstruct images from the dataset, will define a set of filters that can highlight certain characteristics of the cells against the background pixels. Alternatively, a standard thresholding technique could be used instead of deep learning training in this phase, but these techniques require manual tuning of many parameters and mainly use pixel intensity as a feature to discriminate signals from the

background, while a set of features could be more powerful by also capturing other discriminative features. Another method worth exploring is training a regression network that aims to infer the total number of cells. Such a network would promote a set of features to enhance the appearance of cells in the images. However, we chose the easier approach of delivering raw pseudo-labels through autoencoder training. For the reconstruction task we analyzed the same images used for the segmentation problem, but this time we ignored the ground truth labels. We trained the model until early stopping occurred after around 60 epochs. However, highly accurate network training is not necessary for this phase because the model usually learns the required set of filters after a few epochs. The model's architecture starts with a block containing two convolutional layers with 16 filters, which sequentially increases until the bottleneck. It then decreases again through the decoding path, ending with the last layer having only 3 channels, which are necessary to reconstruct the original RGB image.

Discriminative features map

To visualize the feature maps we pass an input image through all the convolutional filters of the model. Each block gives a different number of feature maps that we sum up to get a unique grayscale image that we define as *agglomerate feature map* or *a-fm*. The total number of these grayscale agglomerate filters is 18 all reported from figure 3.11 to 3.15.

To define which a-FM to select, we take a look at the images produced passing an input image 3.10 through the model. Some of these filters enhance the difference between the background value and the cells. Using a false color map we can analyze which of these filters is the most suitable for our purpose. For example, the filters relative to the 15-th convolutional layer seem to accomplish better this scope as visible from the bottom right of figure 3.14. In figure 3.16a the same filter is reported after a normalization. After this visual analysis, we select this a-FM and apply it to all the training-validation sets of images.

After that, we apply thresholding to get a binary mask. First, we need

to define an appropriate value to apply to all the a-FM. For this reason, we design a threshold sweep evaluating the resulting F_1 score. Since the output of the model is not a probability map, the threshold value corresponds to a variable level of the intensity percentile. Ten values starting from $90 - th$ to the $99 - th$ are tested. To evaluate the F_1 score, the same algorithm described in 2.3.4 is used. Each predicted cell's center is associated with the closest ground-truth dot-annotation. If the coordinates are close within a pre-fixed distance (1.5 the diameter of the mean cell), a true positive match occurs. Finally, we select the 99-percentile intensity value as the best value. An example of the resulting pseudo-labels is reported in Fig. 3.16b together with a comparison with the actual ground truth. These pseudo-labels do not contain yet the weak supervision provided by the dot annotation thatis included with the step described in the folowing section.

Weak supervision

Now is the moment to leverage the weak supervision by using the dot annotation. We know that these points correspond to the actual cells that have to be counted. To use this information we prepare some pictures with a boolean circular structure similar to what we saw for the c-Fos dataset. We want to use these images as a boolean mask to define which object from the raw pseudo labels should be kept in the final pseudo-labels. So we multiply the raw pseudo labels with the corresponding dot-annotated image. If the cell from the raw output corresponds to a center in the boolean mask, then the cells are promoted into the final ground-truth mask. These steps are repeated for each image present in the raw pseudo labels. The final image is a ground truth cleaned from spurious objects. An example of these steps application is reported in figure 3.17. On the left we have the raw binary mask, in the center the resulting image after the cleaning process, on the right the actual ground truth is reported for a comparison with our pseudo-label. Some other examples results are reported from figure 3.18a to figure to3.18e. From this comparison, we observe that the size of the cells inside the pseudo-labels is systematically smaller than the ground-truth counter-

parts. We should ignore these observations simulating a real case where the ground truth is not available and so we don't perform further adjustments or post-processing. Remarkably, we observe that only a little fraction of the actual cells are not replicated in the pseudo-labels. Moreover, most parts of the objects keep separated also inside the pseudo-labels without observing the generation of white clustered blobs that would invalidate the training process. Aside from this, we remind that the pseudo-labels can have to the utmost the same objects as the pseudo-labels since the weak supervision exclude the other irrelevant object. In the future work section 3.7 we propose some alternatives to the current strategy to reduce this bias.

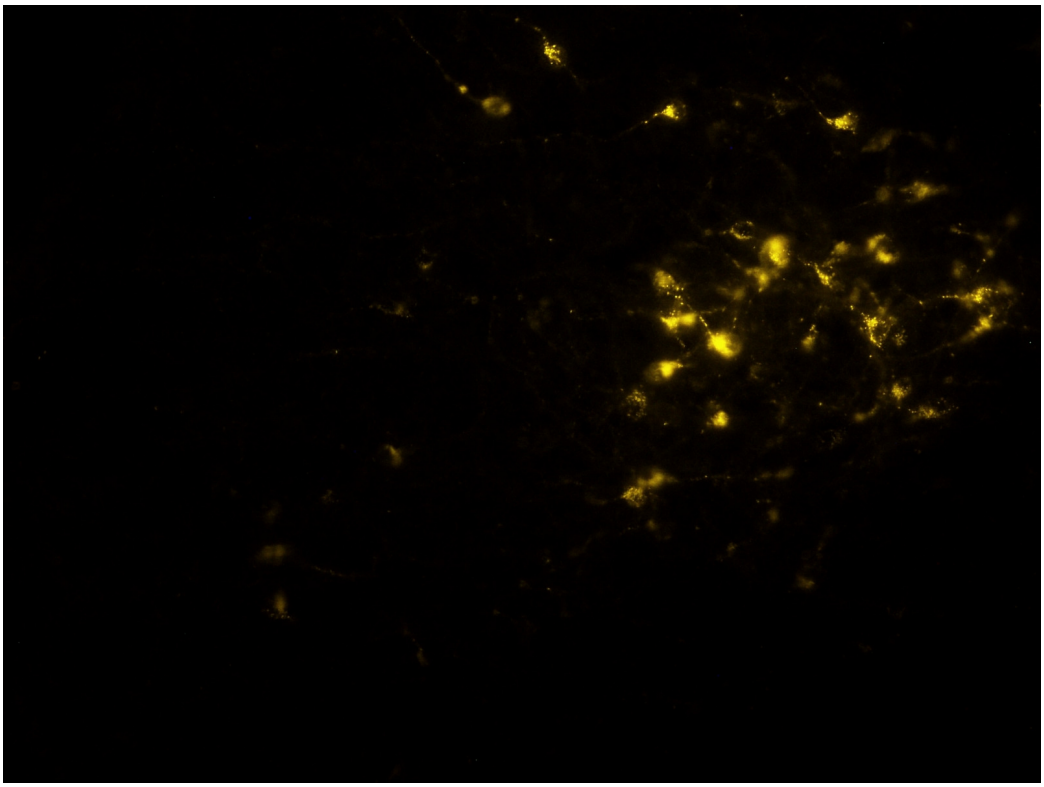


Figure 3.10: Input image used to visualize the agglomerate features maps of the autoencoder corresponding to each convolutional layer.

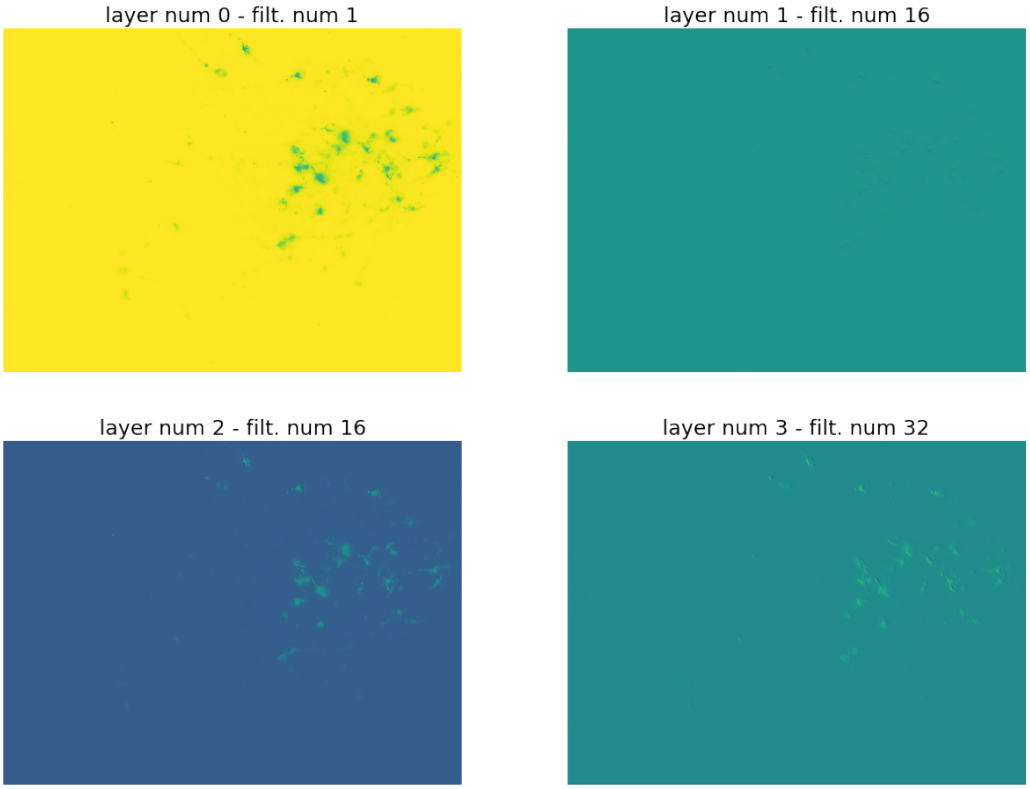


Figure 3.11: Agglomerate features maps from the first to the fourth convolutional layers

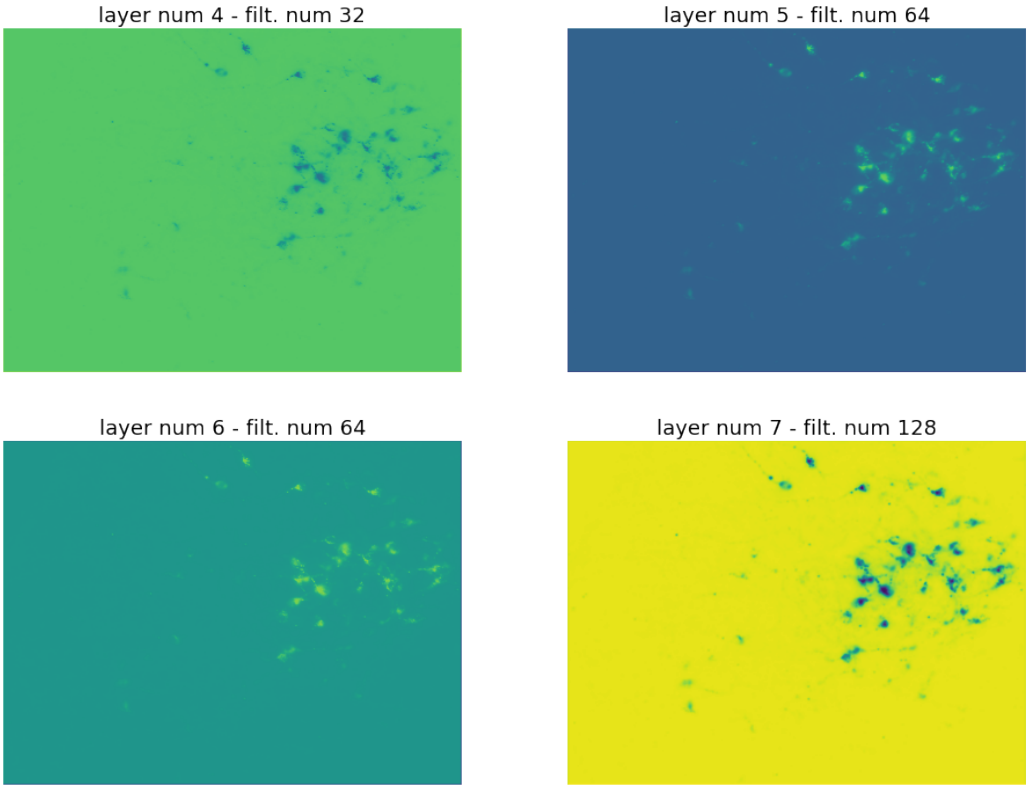


Figure 3.12: Agglomerate features maps from the fourth to the eighth convolutional layers

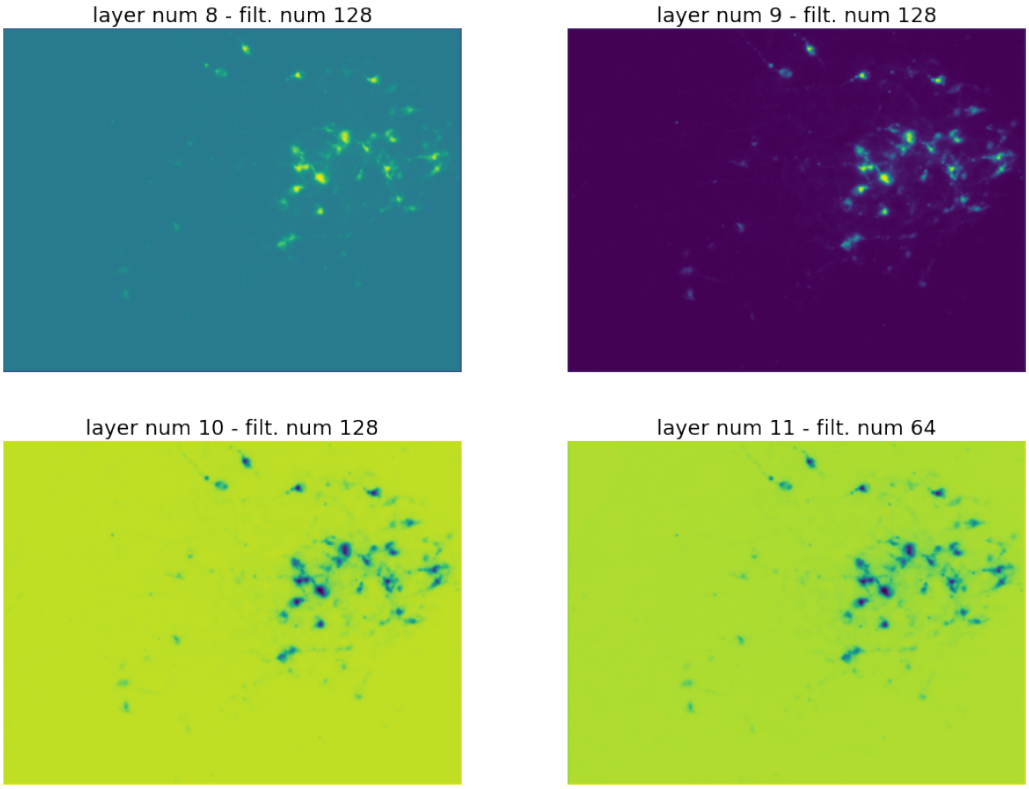


Figure 3.13: Agglomerate features maps from the eighth to the twelveth convolutional layers

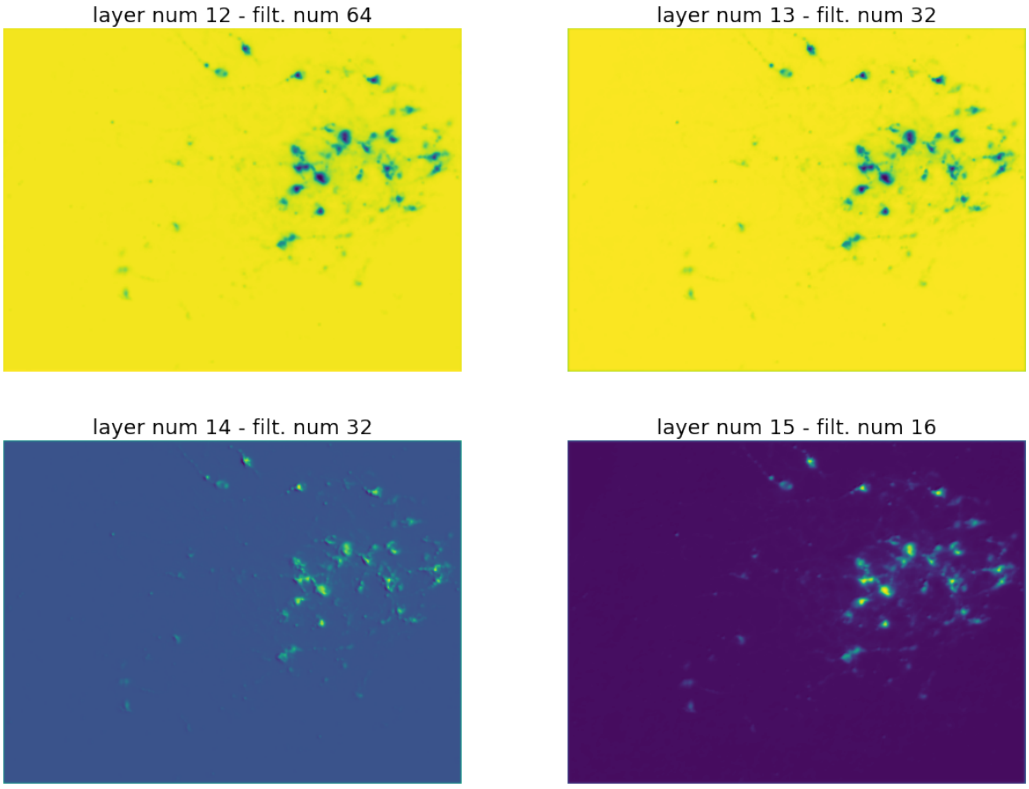


Figure 3.14: Agglomerate features maps from the twelveth to the sixteenth convolutional layers

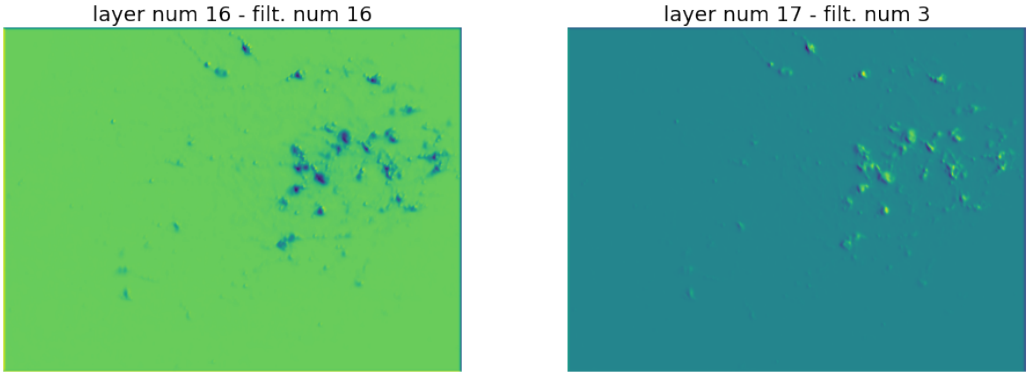
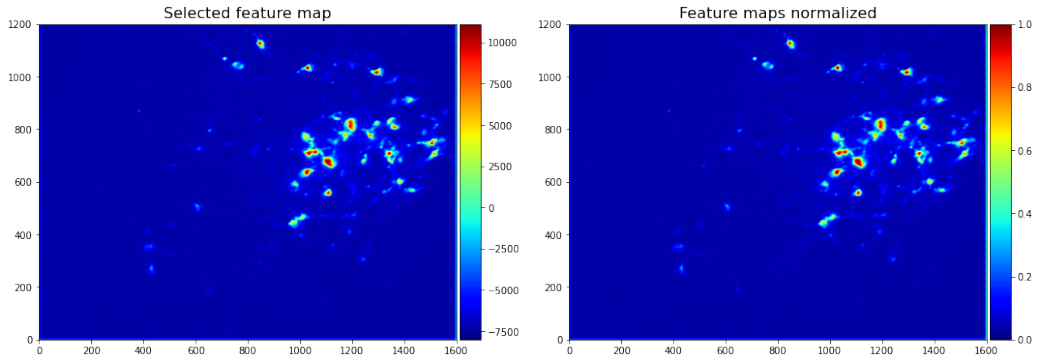
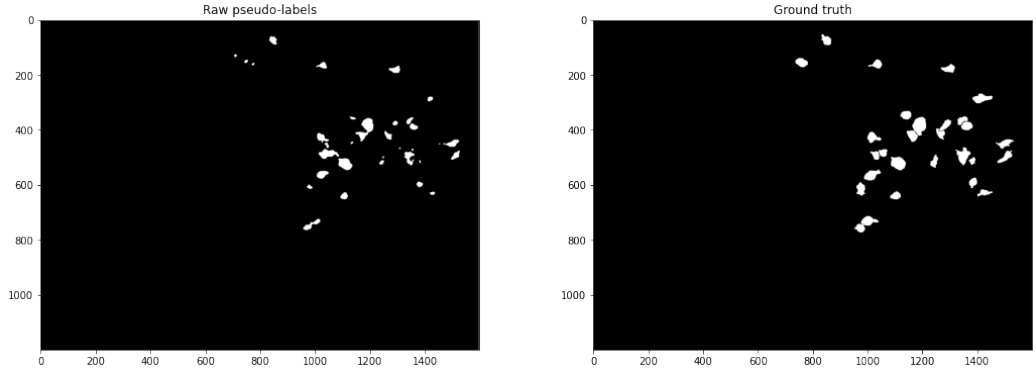


Figure 3.15: Agglomerate features maps from the sixteenth to eighteenth convolutional layers



(a) Feature map obtained from the selected set of filters added togheter to form a one-channel image.



(b) Left. Pseudo-labels obtained with thresholding. On the right the relative ground-truth

Figure 3.16: Selection of feature maps used to generate the pseudo-labels. Top raw. On the left, the original feature map value Right, the same feature map is normalized between 0 and 1. On bottom raw. Left, Binary mask obtained tresholding with the ninty-nine percentile value of the feature map value. The same value relative to each image is applied for the raw pseudo-labels generation process. Right, ground truth mask.

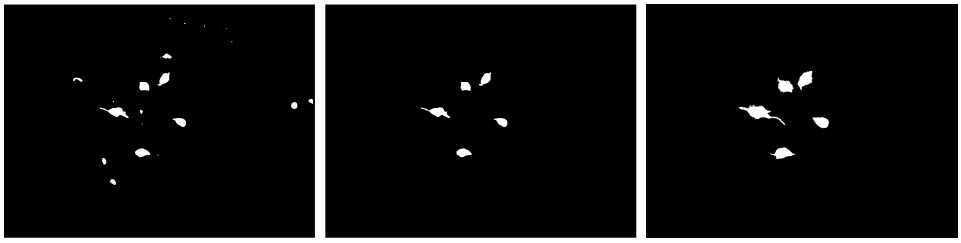
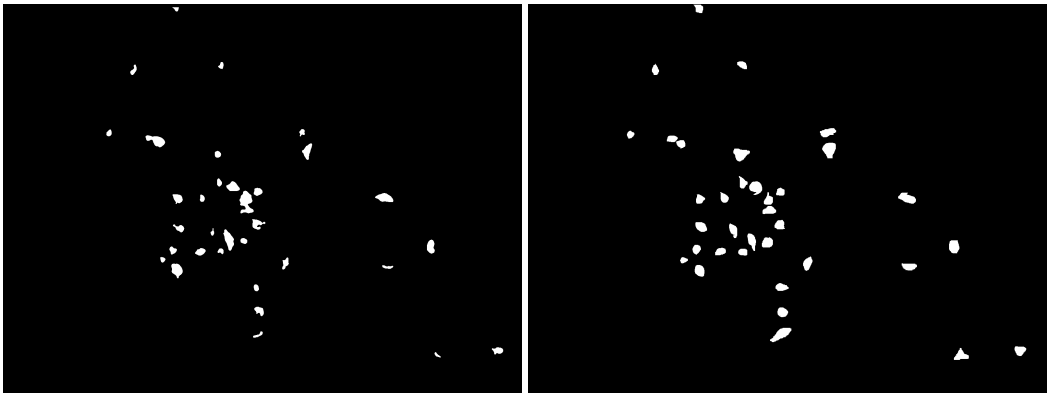
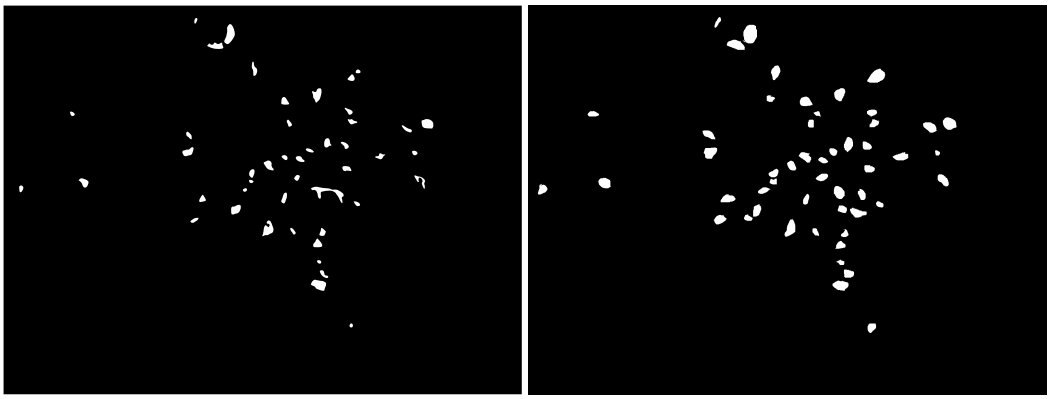


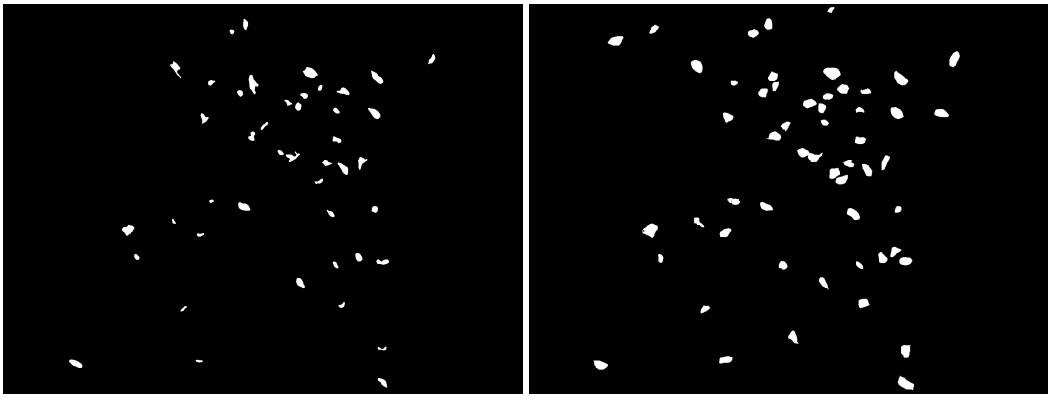
Figure 3.17: From the left. First pseudo-labels obtained with autoencoder feature map and thresholding. Center, application of weakly-supervised dot-annotation filtering. Right, corresponding ground-truth label.



(a) pseudo-label (b) ground-truth



(c) pseudo-label (d) ground-truth



(e) pseudo-label (f) ground-truth

Figure 3.18: Comparison between pseudo-labels generated after the entire process and the corresponding hand-crafted ground-truth. The generation of pseudo labels consist in two main steps: Application of autoencoder filter to get discriminative feature map, and filtering using the weakly-supervised dot-annotation.

3.3.2 Model architecture

C-ResUnet is selected to be trained on both the dataset: cTB and c-Fos. This model is the outcame of ablation study performed on the yellow (cTB) dataset with a fully supervision on the ground-truth masks 2.3.1. For any details on the model architecture we remaind to the specific section of this thesis

3.3.3 Training

In the following, we will refer to **c-ResUnet-y** and **c-ResUnet-g** to the models trained respectively on the c-TB and c-fos datasets.

c-ResUnet-y

We distinguish three different learning procedures for the c-ResUnet-y family

- **c-ResUnet-y** - supervised mask generation. The ground truth is produced both by hand from domain experts and using a semi-automatic thresholding technique.

- **c-ResUnet-y-u** - unsupervised mask generation. The binary masks are generated using the convolutional autoencoder's feature map without using dot-annotation filtering.

- **c-ResUnet-y-ws** - weakly-supervised masks generation. The unsupervised masks are corrected using dot-annotation information.

We apply for each model the same dataset split. From the initial 283 images we first selected randomly 70 pictures for the test set while we further split the remaining part following the standard 70 : 30 proportion, respectively for the train and validation set. During the split, we used the same seed to assess the replicability of the experiment and promote a fair comparison among all the competing methodologies. After this stage, we cropped each image into 12 partially overlapping (120 pixels on the width dimension) patch of size 512×512. We generate weighted maps using the same procedure described in the section 2.3.1 to penalize the loss of the pixels surrounding the higher density cells area. Then, an augmentation was applied to increase the number of examples with the same criterion exposed in the first part of this thesis. The augmentation performs both standard transformations like rotation, flip, blurring, and shift of hue, saturation, and value, and also elastic deformation to get a wider variety of cell shapes. For the images whose ground truth was generated by hand, the augmentation factor increase from 5 to 10 to provide more accurate examples to the model. For the same reason, for a subset of patches depicting relevant artifacts, the factor was increased to 100. After the augmentation process, we end up with nearly 16000 cropped images. We also set each model to the values of the same hyperparameters. We used a batch size of 8 together with an early-stopping patience of 5 and a learning rate of 0.0001. We use a learning rate scheduler to decrease the learning rate value to the 80% of its previous value each time the model does not improve for more than three consecutive epochs. Concerning the loss, we combine a weighted binary cross-entropy loss with the weighted maps setting the class weights to 1.5 and 1 for cells and background, respectively.

It is worth observing that the training, also for the supervised method, is performed again from scratch without reusing the results obtained in the

first part of this work. The main reason is that we want to retrain all the architecture from scratch using the same seed for the train and validation splits. However, since we start from different ground-truth labels, we have to take again the augmentation process for each method introducing a non-reproducible pattern. The code used for the training and subsequent evaluation is written using Pytorch (cell counting yellow Pytorch) rather than Keras (cell counting yellow repository) which was used for the first version of the fluorescent cell supervised training.

c-ResUnet-g

The c-Fos dataset consists of 251 images whose 39 are included in the test set. This time we train only a model, **c-ResUnet-g**, to test the validity of the weakly-supervised approach. In this case, we use a train-validation split following respectively a 85 : 15 division. We cropped the images into 12 partially overlapping patches also in this case but we did not use weighted maps because of the lower density of the cells. The augmentation pipeline is the same used for the cTB dataset except for the elastic deformation that can generate cells examples different from the actual ones, since, in this case, the cells are characterized by a quite regular shape. Moreover, the augmentation factor is set equal to 2 but with some variation depending on the crop features to augment. To promote more significative samples, the split factor decreases to one if in the patch there are no cells while it increases to 4 if there are more than 5 cells. This strategy acts similarly to an oversampling used to alleviate the unbalanced problem between cells and background pixels. Finally, we ended up with nearly 9000 images for the training and validation phases. Once the splits are generated we started the training using a batch size of 8, an early-stopping patience of 5, and a learning rate of 0.001. This last value was handled during the training using a scheduler that decreases its value to the 80% of the previous value, each time the model does not improve for more than five consecutive epochs. We adopt also in this case weighted binary cross-entropy loss with weights equal to 1.5 and 1 for cells and background, respectively.

Parameters	**CTb**	**c-Fos**
remove obj. size	200	4
holes size	600	6
max distance	30	3
foot	40	4

Table 3.1: **Post processing parameters.**

3.3.4 Post-processing

The post-processing pipeline follows the same steps described in the first part of the thesis. Since the cells represented in the two datasets have different sizes, a little fine-tuning of the parameters is required to get a clean result. We remind that the outcome of the model consists of a probability map of the same size as the original input, whereby each pixel value can be interpreted as the probability of belonging to a cell. Using a threshold it is possible to get a binary mask. This segmentation mask represents all the objects detected as cells. However, some spurious objects can pop up and affect the counting. The post-processing step address removing these small objects or filling holes within the cells detected. Anyway, depending on the size of the cells, the parameters used to process these results can change. In the following, we summarize the parameters used for the two different datasets.

The first two parameters are strictly related to the cell's size and regard the area of the object to remove and the area of the holes to fill. The last two parameters, instead, affect the watershed algorithm Soille and Ansoult [1990] that acts to separate the touching cells. It is worth observing how drastically changed depending on the size of the cells to detect. For these parameters, it is useful to have at least some shallow knowledge about the cells' dimensions. These parameters can also strongly affect the results and so accurate tuning is recommended. To make this step easier to handle and to get fast feedback with respect to the parameters sweep, we developed a web app that enables a qualitative and quantitative inspection of the results. We used this interface to test some parameters combination and get the parameters that work better. We will described in detail the web-app features and their scope in the last part of this thesis (5.2.3)

3.3.5 Evaluation

To validate the weakly-supervised methods presented here, we used the same metrics already seen for the c-ResUnet evaluation concerning the detection and counting task problems.

To better understand the validity of the proposed method we compare it with other training procedures. In the first place, we retrain the model in a supervised manner using the ground truth describe in the 2.2.1. We use the supervised models' performance as the benchmark. On the other hand, we also set as a baseline the model trained on the masks generated using only the autoencoder features map without the intervention of weak supervision. We refer to such a model as the **c-ResUnet-y-u** where u stays unsupervised also it could be also intended as a self-supervised learning.

For the c-fos dataset, we only evaluate the model trained following the weakly-supervised approach. To define the best model we use the F_1 score relative to the test set. This value is computed with the same algorithm described in the chapter (REF). The only difference is the distance used to define a true positive match between the predicted mask and the ground truth. Indeed, for the c-Fos, these distance decrease in absolute value but is still equal to 1.5 times the mean cell radius associated with that dataset.

Threshold optimization

Before getting the F_1 on the test dataset,it is needed to find the best threshold to use during the binarization of the heatmaps. To do that, we proceed with threshold optimization upon the train and validation images. We investigate different threshold values ranging from 0.30 to 0.90. Within this interval falls the optimal threshold relative to the maximum value of the F_1 score. Once this value is found, it is used for the test set evaluation.

In the following, we report the curves in Fig. 3.19 representing the F_1 scores of all the trained models. For the fluorescent-cells dataset , we test three different learning approaches while for the c-fos we only have one available approach. Anyway, from these curves we can observe the dependence of the F_1 score from the threshold. These lines look quite smooth demon-

strating the robustness of the results. Indeed, these trends show that the results are not drastically affected by the threshold selection within a quite wide range.

Looking at the result relative to the yellow fluorescent cells (Fig. 3.19 on the left), as expected, the supervised method outperforms the other approaches but the distance with respect to the weakly-supervised learning method is limited to a few percentage points. This is quite surprising thinking that, among the supervised dataset, we have 31 images accurately selected and segmented by human annotators that required great time and effort. For the c-Fos dataset, we have only the curve relative to the model trained with the weakly-supervised approach. The same observation regarding the robustness of the result applies here, but the F_1 score result is a bit lower with respect to the previous case. Anyway, from these curves we get the optimal threshold to asses the model's performances on the test set.

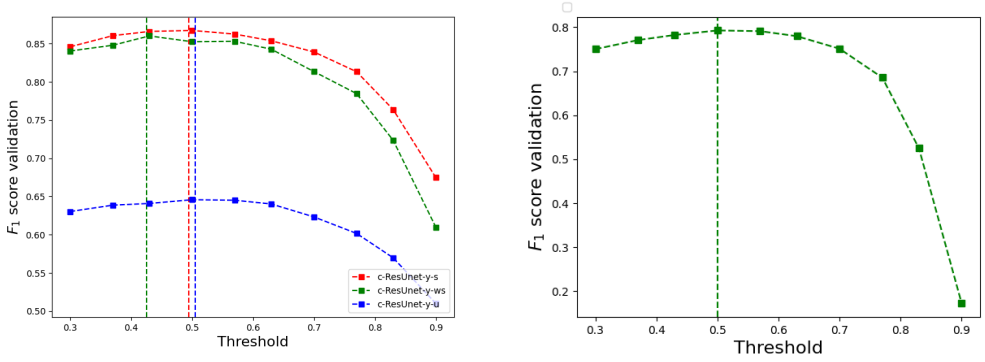


Figure 3.19: **Threshold optimization**. **Threshold optimization**. Results relative to c-ResUnet-y-s, c-ResUnet-y-ws and c-ResUnet-y-u on the left and to c-ResUnet-g on the right

3.4 Results

The test sets are composed of 39 and 70 images respectively for the CTb and c-Fos datasets. The images are provided full-size to simulate closer a real-case scenario where the images don't need to be cropped to optimize the training process. To compute the f_1 all the images of the test set are used. 3.2

reports the performances in terms of both detection and counting ability for the three different models trained with different ground-truth labels. Instead, 3.3 regard the performance metric of the model trained on the green dataset using the weakly supervised approach.

From 3.2 we confirm the superiority of the supervised approach accordingly to what we already saw during the threshold optimization. Also, the small distance in terms of both detection and counting performance between the weakly supervised and the supervised methods is replicated on the test set proving the validity of the method developed in this work. To remark on the impact of the weak supervision we can observe the performance score of the c-ResUnet-y-u and the c-ResUnet-y-us whit this latter model outperforming the former by a large margin. We still remember that the difference between these two approaches resides in the dot-annotations information that is used to remove fake objects from the pseudo-labels obtained from the convolutional autoencoder 3.3.1. We keep in mind that the three architectures are completely equivalent in terms of the number of parameters and the number of images (nearly 16000) used for the training.

Moving to the results relative to the green dataset, looking at the table 3.3, we observe a little decrease in performances with respect to the yellow dataset. From the quantitative scores indeed, we observe a satisfactory detection performance (f_1 score) but a limited counting performance. The MAE and MedAE remark some significant disagreements with the count obtained manually. This is also related to the mean value of cells per image which increases respect for the fluorescent cells dataset. However, we already highlighted that the subjectivity issue for the c-Fos dataset has a big impact (see section 3.3.1) playing a crucial role in the final metric evaluation. Although this evidence, the visual inspection of the results, assisted by a domain expert, confirms the goodness of the results. A way to alleviate these annotations issues would promote an approach performing a mean of different annotator evaluations.

Model	Threshold	F_1	Accuracy	Precision	Recall	**MAE**	MedAE
c-ResUnet-y-s	**0.5**	**0.84**	**0.72**	**0.85**	**0.82**	**2.4**	**1.0**
c-ResUnet-y-ws	0.43	0.82	0.70	0.83	0.81	2.7	1.5
c-ResUnet-y-u	0.5	0.64	0.47	0.53	0.80	7.4	5.5

Table 3.2: **Performance metrics.** Test set performance using the optimal threshold. The first four columns report the detection metrics, while the latter ones evaluate counting performance.

Model	Threshold	F_1	Accuracy	Precision	Recall	**MAE**	MedAE
c-ResUnet-g-ws	0.5	0.80	0.66	0.73	0.86	11	8

Table 3.3: **Performance metrics.** Test set performance using the optimal threshold. The first four columns report the detection metrics, while the latter ones evaluate counting performance.

3.5 Results visualization

From the visual inspection of the results, we can make some observations. We start considering the problem from a counting task/object detection point of view using bounding boxes to highlight the number of true positives, false positives, and false negatives. We resort to the same methodology exposed in the first part of the work. For each object in the output image provided by the model, we search for the corresponding closest object to the ground truth. If the distance between the center of these two objects is lesser than 1.5 times the medium radius of the fluorescent neuronal cells, a true positive is counted and a green bounding box is displayed in the output image around the shape of the segmented object. Red and blue boxes are drawn respectively for false positive and false negative objects. We report only the significant images useful for a critical evaluation of the results. In Fig. 3.20, for example, we observe a case where many false positives are detected. However, these fake detected objects are very similar to the actual ones and sometimes look indistinguishable also for a human operator. In Fig. 3.20 instead, we report an opposite example where the c-ResUnet-y-ws is more conservative with respect to the operator who annotated the image, and many false negatives are reported. Considering again the metric score (3.2) the overall tendency is to

be more conservative. This is remarked by the recall value that is lower than the precision both for the supervised and the weakly-supervised model also if, for this latter model, the difference is very small. However, in most cases, these two trends tend to balance how visible in the image 3.20. As a result, the predicted count approach the target one also is attributed to different objects with respect to those considered by the operator. We also report the raw output 3.24 of the model that consists of a heatmap whose pixel values are related to the probability to belong to a cell or not. From this picture, we can verify the capability of the model to keep the touching cell separated. This outcome is also an effect of the weighted maps used during the training as demonstrated in the first part of this work. Moreover, from the same image, we can observe that the segmentation heatmap of the model results in shapes that look very similar to the ground-truth one. We remind that for these images we have the ground-truth label but we didn't use it during the training. Also, a comparison between the heatmpas produced from the c-ResUnet-y-ws and c-ResUnet-y-s model is reported in 3.22. Although the latter model is trained on a conspicuous part of a handcrafted segmentation mask, the maps look very similar. An effect that we can ascertain is the smaller size of the object produced by the weakly supervised model but this is expected since the pseudo-labels objects are smaller than the ground-truth counterparts like visible in the Fig. 3.18. This effect is related to the mask generation and the high threshold we use to produce the pseudo labels. However, the counting performance is not drastically affected by this situation also if it may require further fine-tuning of the post-processing parameters.

In the same way, we approach the analysis coming from the c-ResUnet-g trained in a weakly-supervised manner. Starting from the bounding boxes result, we report similar situations to those found for the the c-ResUnet-y-ws model. First a case with a significant number of false positives and then an opposite example with a larger number of false negatives. Last a case where these terms are balanced. This time, the tendency is opposed with respect to the c-ResUnet-y models since the recall is greater than the precision value 3.3 indicating a tendency to output more false positives than false negatives. In other words, the model is lesser conservative than the

human annotator. Following the same observation valid for the fluorescent dataset, also here the true positive can easily be confused with a true object and vice-versa. Indeed, this disagreement is more related to the noisy dot annotation provided to the model. Last, 3.24 some heatmaps are reported together with the images overlapped by the ground-truth labels. The white circle only indicates the mask we used for the train but, in this case, doesn't represent the accurate hand-labeled segmentation masks. From this image, however, we see that the shape produced by the model are not simple circles but seems to resemble the actual shapes of the cells. We also take these two images to remark on the low reliability of some masks. Indeed, in the first image, we have dot-annotation still on the edge on the left side of the images but from a posterior assessment, we know that is a mistake made by the operator. However, during the performance evaluation, this *not-error* are attributed as mistakes of the model. The second image, instead, reports an example of misalignment between the coordinates of the dot annotation and those of the actual cells.

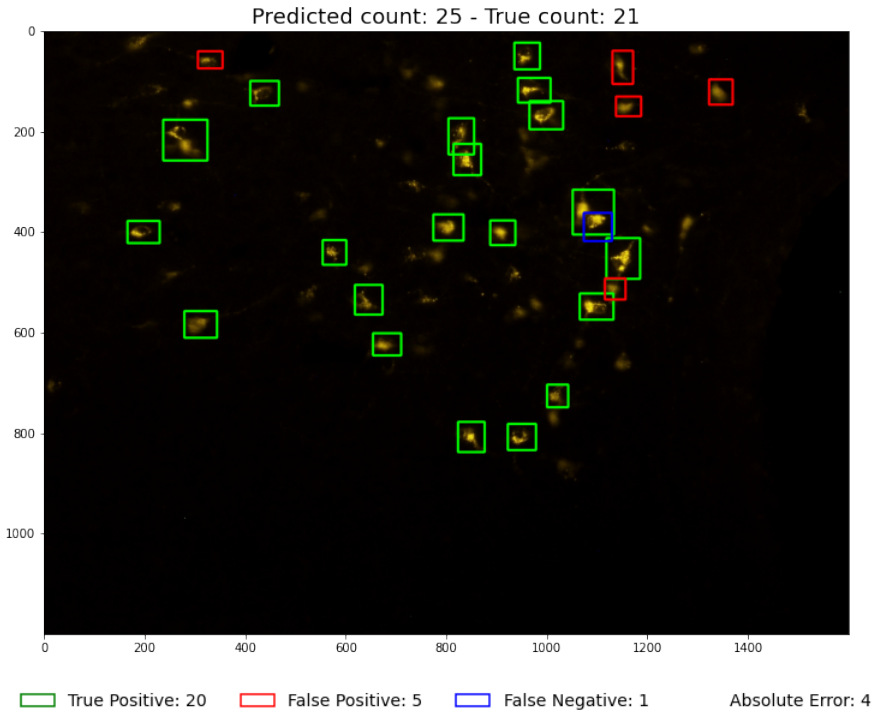


Figure 3.20: **Results on test images (1)**. In this picture, we can observe a relevant number of false positives.

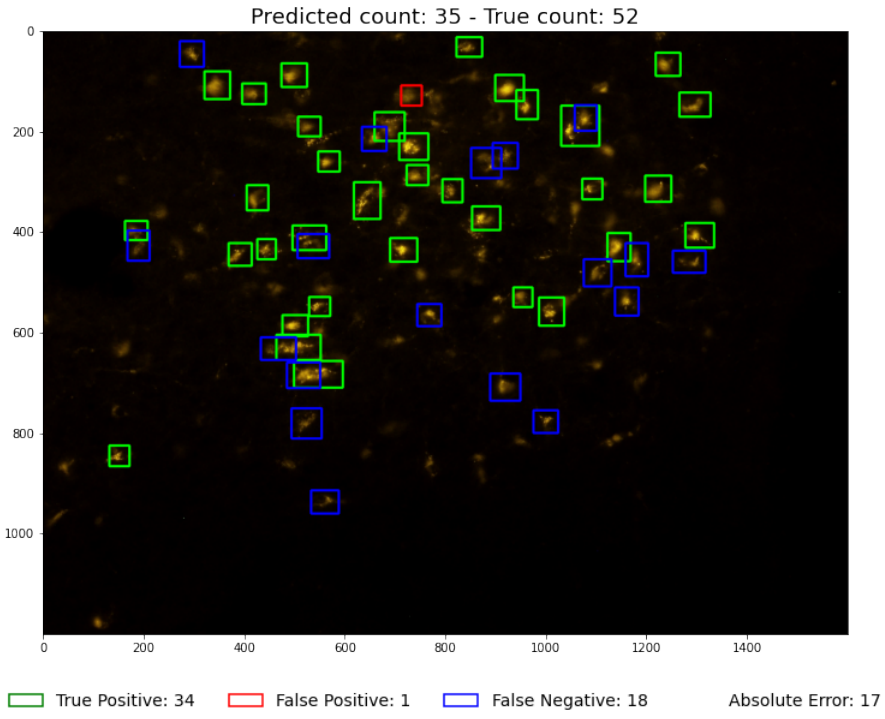


Figure 3.20: **Results on test images (2)**. In this picture, we can observe a relevant number of false positives.

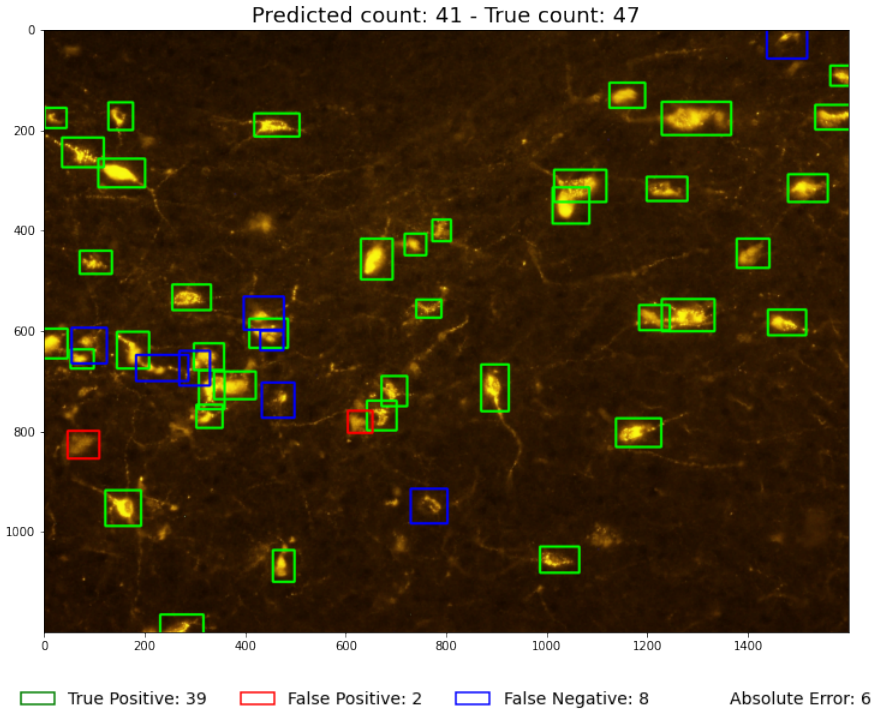


Figure 3.20: **Results on test images (3)**. In this picture the number of false positives and false negatives it balanced. As a result, the predicted number is close to the target one.

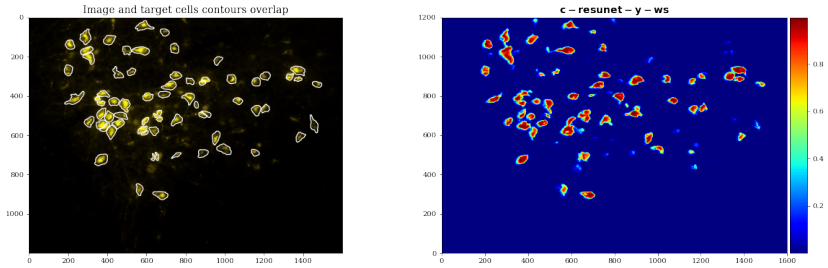


Figure 3.21: **Heatmap**. On the left, the image with overlapping cells' boundaries. On the right the corresponding heatmap produced by the model. The touching cells are well-separated resulting as distinct objects.

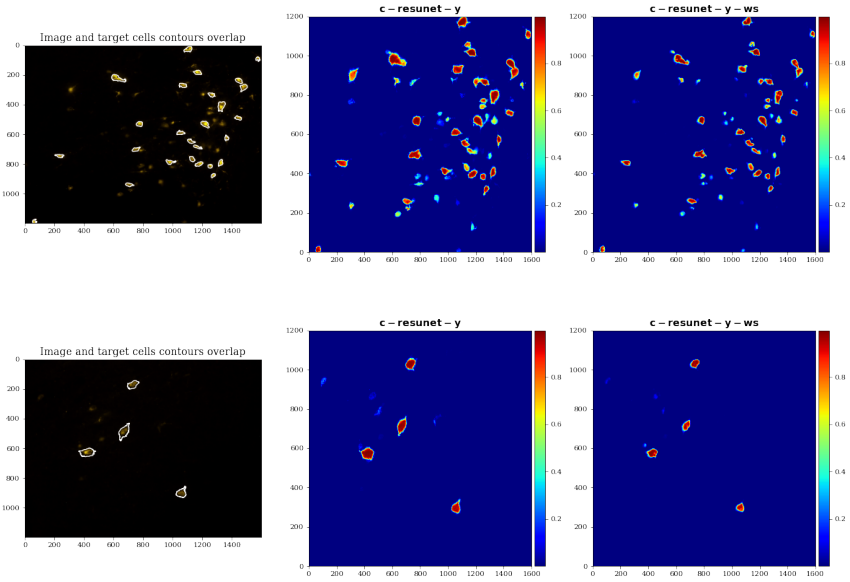


Figure 3.22: **Heatmap comparison**. On the left, the image with overlapping ground-truth cells boundaries. On the center and on the right respectively the hetmap produced by the supervised and weakly supervised c-ResUnet. The size of the cells detected from the c-ResUnet-y-ws look smaller than those identified by the c-ResUnet-y. This effect is related to the pseudo-labels generated that tend to represent a limited inner area of the neurons.

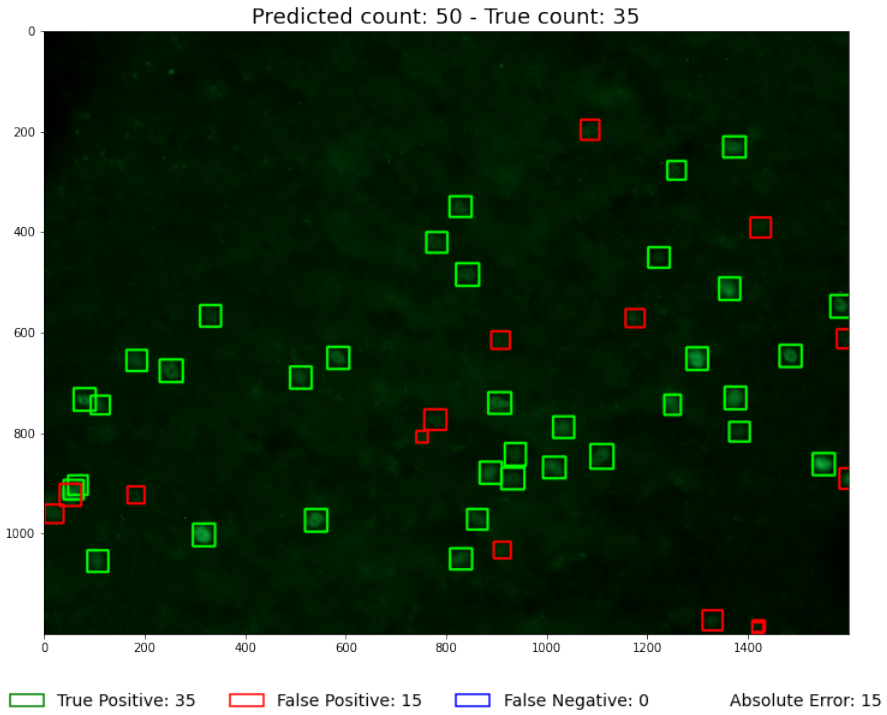


Figure 3.23: **Results on test images (1)**. In this picture, we can observe a relevant number of false positives.

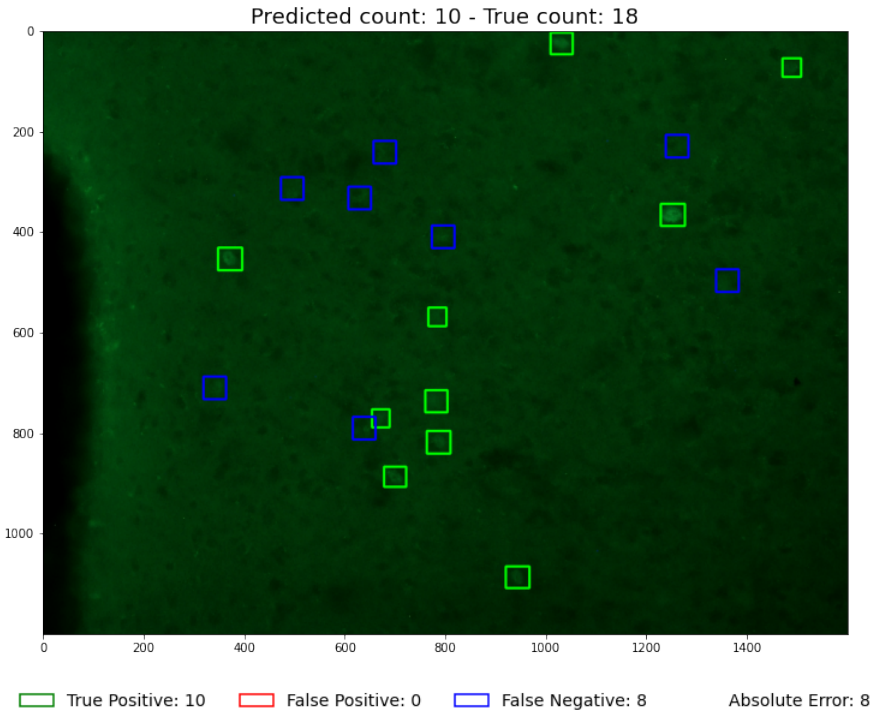


Figure 3.23: **Results on test images (2)**. In this picture, we can observe a relevant number of false negatives.

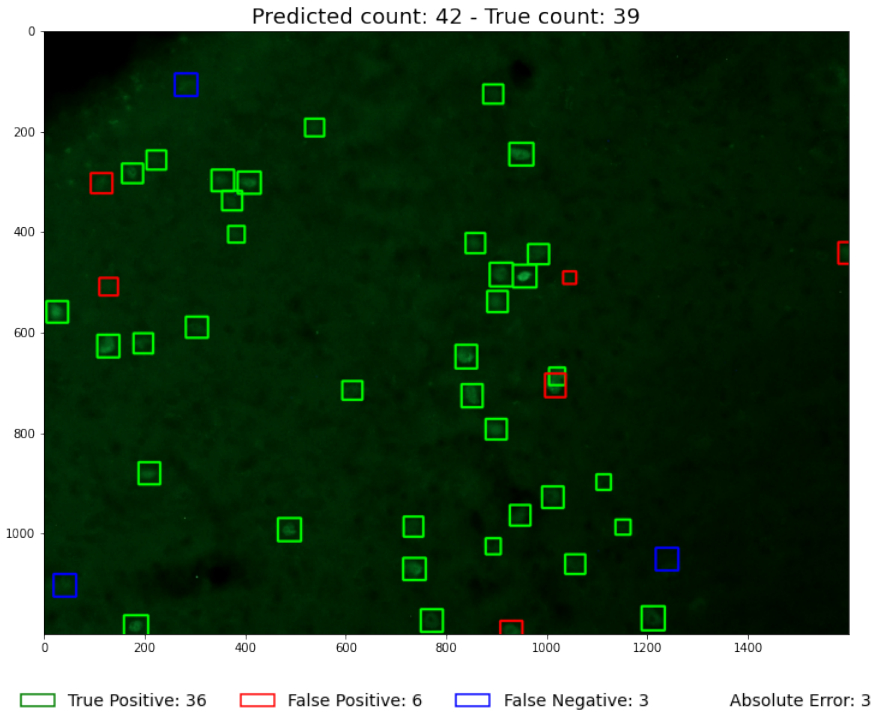


Figure 3.23: **Results on test images (3)**. In this picture the number of false positives and false negatives it balanced. As a result, the predicted number is close to the target one.

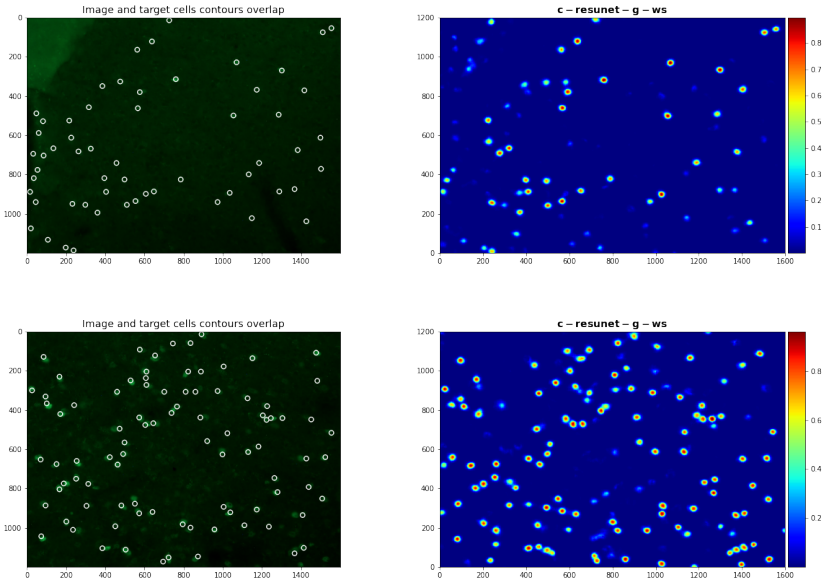


Figure 3.24: **Segmentation Results**. On the left, the input image with overlapping pseudo-labels. On the right, the heatmap produced by the models. From the heatmap we can notice that the shapes segmented by the model resamble the actual cells appearence also if the ground-truth labels are circular boolean structures.

3.6 Final Remarks

In this part of the work, we tackled the absence of supervised ground-truth segmentation masks that usually occur in the microscopy domain. We try to develop an easy approach to address the count-by-segmentation task when dot annotations are available. We distinguish two use cases. The first is relative to a simplified scenario where the instances to detect have homogeneous sizes and shapes and can be approximatively fitted by a geometrical structure like a circle. In this case, we generated some pseudo-labels by placing a boolean circle mask on the center of the dot annotations and using these masks to train the model in a supervised manner. This method represents a very easy and fast approach but is constrained by the shape of the detected object. The results relative to the model trained in such a way demonstrate the validity of the methods also if the counting task metrics are still limited.

The main reason is attributed to the noisy dot annotations provided by different human annotators with different degrees of expertise. Indeed, from a posteriori visual inspection, the most experienced annotators confirmed the goodness of the results provided by the models.

The second use case, relative to the fluorescent cells dataset, pointed out a more general use case. The instances to detect reflect a large variety of sizes and shapes preventing the application of the method developed for the c-Fos pictures and requiring a different pseudo-labels generation strategy. Leveraging an image reconstruction pretext task and the dot-annotation information we provide a viable method. By combining autoencoder features maps and the coordinates of the objects to detect represented by the dot-annotations, we end up with a set of pseudo-labels used to train the c-ResUnet model. The performance comparison between the model trained in such a manner and the supervised one demonstrates the validity of the described method. We found only a small margin between the counting and detection metrics score reached by the the two different approach.

In conclusion, the proposed approaches proved to be solid candidates to provide count-by-segmentation problem when only dot annotations are avaiable. We suggest addressing the problem following the first method when the shape fof the objects to detect fulfilled the required constraint. However, the second approach provided is more general and can be applied also in the case of more involved cell shapes but requires more step to be performed.

3.7 Future works

One of the critical points of the work is the generation of pseudo-labels. Here we exploit the features maps of an autoencoder trained on a reconstruction task. After that, the selection of one set of agglomerate feature maps is needed to produce a first round of raw pseudo-labels. However, this step is subject to a certain degree of arbitrariness and a wrong choice can imply a bad quality mask generation. This manual selection could be replaced by an automatic procedure exploiting the histogram value of the features map generated by the autoencoder. Through these values should be possible to

understand which convolutional layers developed filters that better separate the background from the foreground objects' histogram peaks.

However, the real step forward in this sense would be the a-priori definition of which model's layer should be specialized to learn such kinds of discriminative filters. In this case, no one should make any decision in this regard and select directly that specific set of features. A good candidate would be the last layer of a model trained on a different pretext task. In recent years the self-supervised-learning research field is focused on such types of tasks to generate usefull pseudo-labels. For example, in place of the autoencoder, a regression network trained to detect only the center of the cells or to infer the number of cells inside a picture could be a better choice. A comparison between this alternative strategy to optimize and automatize all the training would be a definitive step forward in this research domain.

Attaining to the work described here we saw that the pseudo-labels generated always have a smaller, and in the best case the same, number of cells with respect to the dot annotations. To adress this bias we think about a possible strategy. For example, for a fraction of the images we can preserve a little percentage of *spurious* objects and/or remove some objects also if validated from the weak supervision. This strategy is based on the assumption that also in the supervised ground truth, we can observe some wrong labels. Moreover, by repeating the pseudo-labels generation we can achieve many different datasets to perform a model ensembling that resembles the presence of several human annotators.

Chapter 4

Transfer Learning

In this chapter, we adopt a transfer learning approach using as the source and target domain respectively the cTB and the c-Fos datasets. During the procedure, we investigate relevant aspects like the optimal pre-training epochs or which groups of layers are best to unfreeze. Then, we systematically investigate the relationship between the performance score and the number of images available in the target domain. We lead this study by comparing the metrics of scratches and fine-tuned model families. The code used to replicate these experiments is collected in the following repository (transfer learning from CTb to c-Fos dataset)

4.1 Introduction

Data availability is a major constraint for that concerns model training. Scarcity of data, and, especially of annotated data, may prevent the application of a data-driven model. However, to alleviate this problem usually transfer learning approach can be used Pan and Yang [2010], Tan et al. [2018]. However, transfer learning can have also a negative effect when the corresponding domains present features that are too far apart Weiss et al. [2016]. Nevertheless, when the positive effect is achieved the model performance increases.

In our case, the datasets considered belong to the same fluorescent mi-

croscopy domain but represent quite different neuronal cells. Indeed, a model trained on one of these two datasets systematically fails to recognize the cells of the other. However, in this chapter, we want to analyze if some information transferability is possible Yosinski et al. [2014]. To prove that, we extensively compared pre-trained models after a fine-tuning procedure with models trained from scratch only on the target domain. We respectively set the cTB and the c-Fos datasets as the source and the target datasets. It is worth noting that while on the target domain, we only have dot label annotations for the source dataset we also have the supervised binary masks. Although several definitions exist to identify the different degrees of knowledge moving from a domain to another Pan and Yang [2010] Tan et al. [2018], the case concerning a switch from supervised to weakly supervised is not documented yet to the best of our knowledge.

We started investigating the optimal number of pre-trained epochs and which layers are better left free to be trained (unfreeze layers) during the fine-tuned procedure. This latter topic has already been explored for many problems like image classification Lee et al. [2022] Guo et al. [2019] and here we want to contribute to what concern the counting-by-segmentation task. Once we fulfilled these questions, we systematically compared fine-tuned models with models trained from scratch varying the number of images available from the target domain. Indeed, we want to remark on the fine-tuning benefits, especially in a situation of scarce data availability. We assume that, if the target domain has enough data, the advantages related to a fine-tuning procedure can be swept away since the information is already contained in this latter dataset. So, we quantify the amount of data needed in the two cases (scratch versus fine-tuning) to reach a given score.

4.2 Dataset

We used the dataset already introduced in the previous chapter. Specifically, we use as a source and target dataset those presented respectively in 2.2 and 3.2, so we refer to these sections for a detailed description.

4.3 Methods

We present the ablation study performed to achieve a better pre-trained model architecture in terms of two key aspects:

- Number of epochs for the pre-training stage;
- Selection of layers to unfreeze during the fine-tuning

The architecture used for these experiments is the c-ResUnet described in the section 2.3.1. We trained such a model with the same number of parameters used in the previous chapters (see chapters 2 and 3). Once the model is pre-trained on the cTB dataset, it has been fine-tuned on the target c-Fos dataset following different strategies illustrated in the next ablation studies (see section 4.3.1).

4.3.1 Ablation studies

Optimal pre-trained epochs

The ablation studies are aimed to investigate two different aspects. The first one is related to the number of epochs used to pre-train the c-ResUnet on the cTB dataset. The scope is to understand if a model trained until it reaches the convergence for the source task is better than a model that only starts to acquire information about the source dataset. The outcomes are to be considered valid only for this particular application. To do that, we saved the c-ResUnet model trained on the yellow dataset, c-ResUnet-y, not only at the end of the training but also at the intermediate steps. We took three different models:

- An early stage pre-trained model: **c-ResUnet-y-1-g**;
- An intermediate stage pre-trained model: **c-ResUnet-y-5-g**;
- A fully pre-trained model: **c-ResUnet-y-12-g**

Here, the number used at the end of the names represented the number of epochs spent since the training started while the letter y and g represent the transfer learning direction: from yellow dataset pre-training to green dataset fine-tuning. The first model is stored after the first very epoch. The second is relative to an improvement on the validation loss that happened at the fifth epoch and the last is saved as the last improvement occurred before early-stopping finished the train.

Unfreezed layers

The second aspect concern the layers we want to unfreeze during the fine-tuning phase. Usually, for the classification task, the only very last layer is replaced to accommodate a new fully-connected layer able to classify different classes concerning the source domain. The segmentation aim is more frequently used to unfreeze the decoder path of the network since the encode is deputed to learn low-level features that are, usually, transferable moving from one domain to another. Together with this combination, we also test the other two configurations consisting of a network fully unfreezed and a network with only the last layer unfreezed. However, inspired by recent work Lee et al. [2022], we also test other configurations. Specifically, we jointly unfreezed one or more blocks linked by a long-skip connection. The three architectures added to these experiments are reported in Fig. 4.1. Overall we tested several combinations whose names are summarized here and resort for results reported in the next section (4.4).

- **encoder**: encoder path and bottleneck are unfreezed;
- **decoder**: decoder path and bottleneck are unfreezed;
- **1-last-layer**: first convolutional-block and the corresponding long-range connections up-convolutional block are unfrozen (first architecture on Fig. 4.1);
- **1-bottle-last-layer**: as the previous but adding the bottleneck to the unfreezed blocks (second architecture on Fig. 4.1);

- **1-2-bottle-lat-layer**: as the previous but adding the second convolutional-block and the corresponding long-range connections up-convolutional to the unfreezed blocks (third architecture on Fig. 4.1).

- **full**: all the layers are unfreezed.

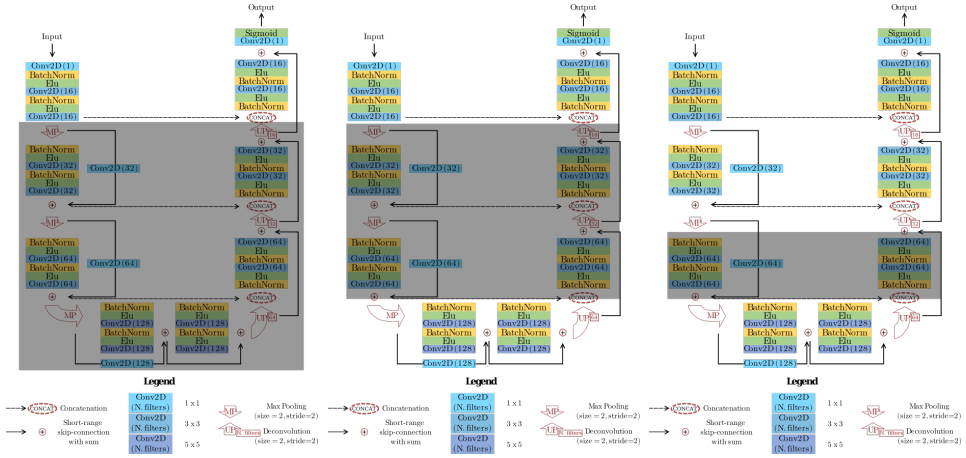


Figure 4.1: **Fine-tuning**. Fine-tuned architecture. Shaded areas indicate the layers freezed during the fine-tuning.

Resampling method

After the selection of the best architecture in terms of the number of pre-training epochs and layers trainability, further investigations are performed to directly compare fine-tuned with scratch models. We carried out these experiments by varying the number of target dataset pictures used. From the total of 240 images, we randomly sampled several times a different subset of n on images. Specifically, for each pre-fixed number n, we sampled four different samples to simulate several scenarios and to test the robustness of the algorithms to the data fluctuations. Indeed, we saw from 3.2 that several images can be noisy annotated, generating some issues for the model training. We want to asses if one of the competing strategy models can handle differently these hard-learning situations. We select 7 different$n$ for a total of 46 subsets. We extracted only three subsets for the 150-image case

since with this amount of images the subset starts to share the major part of the images and the training requires a large quantity of time. Indeed, the subsamples are only partially disjointed since the sampling is random. So, increasing the number of n also rise the likelihood to have many common images between the different samples. Contrary, for low n we are confident to have quite different subsets. It is worth reminding that the resampling act on the pictures is already cropped. So, when an image is extracted among the 240 available, actually all the related crops together with their augmented version are used to form the subset.

4.3.2 Training

For the pre-training part, we first train the c-ResUnet on the cTB dataset saving the checkpoints at different epochs: the first, the fifth, and the twelfth being also the last time the model improved its validation loss. Being a training from scratch, we used the same hyperparameters training selected in the 2.3.2. We started from the same pictures of the supervised case, but we make the augmentation again using the artifacts oversampling strategy and obtaining nearly 16000 cropped pictures. For the fine-tuning phase, we trained the models on the c-Fos dataset. For the first two ablation studies 4.3.1, we selected the same images used for the weakly supervised training 3.3.3 using all the pictures available, nearly 9000. Contrary to the scratch case, for the fine-tuning purpose, we switched the learning rate from 0.001 to 0.0001. The first round of fine-tuning defined the optimal number of pre-training epochs.

Then, using the best pre-trained architecture defined in the previous steps, we performed another round of fine-tuning to select which layers to unfreeze moving from the source to the target dataset. Also in this case, we use all the images available (nearly 9000) and a learning rate of 0.0001. The other hyperparameters keep the same as that exposed in 3.3.3. The study consists of training both scratch models from scratch and pre-trained ones on each of the subsets defined following the procedure described in the previous section 4.3.1. The only difference is the learning rate that for the scratch

case is 0.001 and for the fine-tuned model is 0.0001. For the model trained from scratch, we also tried to use the same learning rate of the fine-tuning case but obtained worsen results, so we decided to use 0.001.

4.3.3 Post-processing

We applied the post-processing using the same parameters used for the c-Fos dataset (see table 3.1 in section 3.3.4).

4.3.4 Model evaluation

For the evaluation purpose, we resume the same methodology exposed in chapter 3.3.5. For each model, we first find the optimal threshold value, and then we used this parameter to get the metric scores also on the test set. The only difference is that for the model trained on the resampled subsets, the train and validation sets are smaller than the full pictures dataset, while the test set keeps the same being the model evaluated on all the images available. In the next section we illustrated the result obtained for each round of experimentation.

4.4 Results

Here, we analyze the results of each study performed, which are aimed to obtain the following answers:

- Best pre-training epochs number;
- Wich layers to train during the fine-tuning procedure;
- Quantification of detection performance boost under the transfer-learning approach.

We first evaluated the outcomes related to the first two points. This evidence defines the model that is later used for the comparison between the from-scratch and fine-tuning approaches. This last comparison is the bulk of this chapter and is analyzed in the last results section.

4.4.1 Design evaluation

Pre-training epoch number

The results reported here concern the comparison of three pre-trained architectures which undergone a different number of pre-training epochs. We used the nomenclature introduced in the section 4.3.1 to refer to these three different models. To give more consistency to the results, we don't limit the comparison concerning the fine-tuning of all the architecture layers, but we also compare the results obtaining unfreezing only the decoder part (bottleneck included). After the first round, we decided to further investigate only the 5 and 12 epochs model since the 1-epoch model demonstrated to stay significantly behind in terms of performance. From the table 4.2 we observe a clear trend that identifies the network trained until the end (12-epochs model) as the best one. So we decide to select the 12-epochs pre-trained models to proceed with further investigations (4.4.1, 4.4.3).

Unfreeze part	c-ResUnet-y-1-g $\boldsymbol{F_1}$	c-ResUnet-y-5-g $\boldsymbol{F_1}$	c-ResUnet-y-12-g $\boldsymbol{F_1}$
decoder	0.766	0.790	0.801
full	Na	0.779	0.796

Table 4.1: F_1 **score results.** Results for different training stages. Each model underwent a distinct number of pre-training epochs.

Unfreezed layers

These successive rounds of experiments aimed to discover which unfreezing layers strategy performs better. We tested different fine-tuning combinations 4.3.1 reporting the results here. This ranking is led by the decoder and the full unfreeze architectures that reach the same performance. However, we decide to pick the former architecture since it requires unfreezing a smaller number of parameters to proceed with the final investigation reported in the next section (4.4.3).

Model name	F_1 score
encoder	0.732
decoder	0.801
1-last-layer	0.537
1-bottle-last-layer	0.782
1-2-bottle-last-layer	0.779
full	0.796

Table 4.2: F_1 **score results.** Results from different unfreezing layers strategy.

No. images	Mean (Median) ± std $\boldsymbol{F_1}$ Fine Tuned	Mean (Median) $\boldsymbol{F_1}$ ± std Scratch
10	0.74 (0.74)± 0.02	0.66 (0.68) ± 0.04
20	0.76 (0.75)± 0.007	0.68 (0.68) ± 0.019
30	0.76 (0.76)± 0.017	0.73 (0.73)± 0.014
40	0.78 (0.78)± 0.010	0.72 (0.72)± 0.02
70	0.78 (0.78)± 0.012	0.76 (0.76) ± 0.007
100	0.77 (0.77)± 0.02	0.71 (0.73)± 0.07
150	0.79 (0.79)± 0.005	0.78 (0.78)± 0.003

Table 4.3: **Performance metrics.** The test set performance using the optimal threshold. We report the comparison between the fine-tuned models and the model trained from scratch. Mean, Median values are reported with the standard deviation uncertainty interval.

4.4.2 Fine-tuning

Here we analyze the results of the fine-tuned models and the models trained from scratch. These models are trained on the subsets of images resampled from the starting dataset following the strategy illustrated in 4.3.1. The results are reported in table 4.3 in terms of mean and median F_1 score together with the number of images included in the training subset.

4.4.3 Performance

The results reported in table 4.3 show a clear superiority of the fine-tuned models that stand constantly above the models trained from scratch. This evidence lingers until a sufficient number of images are used allowing all the models to reach a performance plateau (150-image results). There are several

aspects that it is worth highlighting. First, we note that the gap between the two classes of models is larger when only a few percentages of images are included in the training. Under this *few-shot* regime, the scratch models struggle to be competitive reaching a quite poor F_1 score while the fine-tune models, already with only 10 images, reach a competitive mean score value. These latter models, indeed, start to approach the best scratch performances obtained in the previous chapter 3.4 using all the 240 images, already with 40 images that are six times lesser than the entire dataset size. With the same amount of images, scratch models stay around an F_1 score of 0.72 which is still lower than the lowest fine-tuned model score results reached with 10 images (0.74). These gaps are drastically reduced to around 70 images when the performances start to approach their plateau. Comparing these results with the previous chapter we also observe that to reach the same results we only take 150 images instead of the 240 images representing the full-size dataset. This means that the information contained inside the images became redundant after this amount of pictures. Moreover, applying a fine-tuning procedure, we halved this number of images using only 70 instead of 150 images to reach the best F_1 score. Indeed, two of the fine-tuned model trained with 70-images subsets reaches the limit of 0.79

No. images	Mean $\boldsymbol{F_1}$ Fine Tuned
subset 1	0.79
subset 2	0.79
subset 3	0.78
subset 4	0.76

Table 4.4: **70 Subset results**. A focus on the results relative to the four 70 subsets.

However, around 100 images a countertrend result appears. Indeed, for this amount of images, we expect the fine-tuned models definitively approach the maximum performance value attested between 0.79 and 0.80. Instead, a slight drop in the results sweeps the score to one point behind the previous step (70 images). This evidence is even more visible for the scratch models that collect mean results of 0.71, lower than the 30 image results.

Perhaps, among the subsets resampled with 100 images, one or more collections are introducing noisy annotated examples like those remarked in the section 3.3.1. These subsets lead to poor model training and so to a lower score. We report the median value to highlight this evidence but, with only 4 measures, this metric gives the third-best result and so is not so helpful. So, to investigate better this point, we report in table 4.5 the results of every single model trained on the distinct subsets comparing scratch and fine-tuning results. There are two main aspects to highlight. The first is related to the fine-tuned class that, in one of the subset, reach the best metric value of 0.80. The second concerns the scratch class, which exhibits a sharp drop in the performance in the correspondence of the third subset. From this latter observation, we can state that the scratch models are more sensitive to the noise included in the dataset which may significantly affect the training stage.

To give more evidence to this latter observation, we can also take a look at the standard deviation values that are generally higher for the scratch model family indicating larger results fluctuations. Indeed, for the 100-image case, the standard deviation is more than three times larger than the fine-tuned counterpart. On the other side, with only 4 measures, we take cautiously these results that, however still represent a useful insight for the practical aspects where we usually deal only with subsamples of the entire dataset. From these last results, we can assess that this subset selection can impact drastically a model trained from scratch. Another remark concern the discrepancy of the results between the subset resampled. Indeed, we should expect that the subset that induces a poor test performance for one class of model should negatively affect also the other class but the results depict a different situation.

To focus further on the variability issue, we report also a set of violin plots that better remark the variability related to the subset resampling. In the first plot (Fig. 4.2) the F_1 score results for each model are reported. As already highlighted, the fine-tuned models class is above the scratch counterpart but this gap progressively reduces increasing the number of images and it vanishes at the last collected point (150-images case). From these

Subset Number	Mean F_1 Fine Tuned	Mean F_1 Scratch
subset 1	0.77	0.78
subset 2	0.80	0.74
subset 3	0.78	0.60
subset 4	0.74	0.73

Table 4.5: **100 Subset results**. A focus on the results relative to the four 100 subsets.

plots we also observe the degree of dispersion that imply a more stretched distribution, confirming a greater variability results for the scratch class. A one-to-one comparison is reported in figure 4.3. Here, together with the median value, the mean is also represented as a red spot overlapping the violin plot.

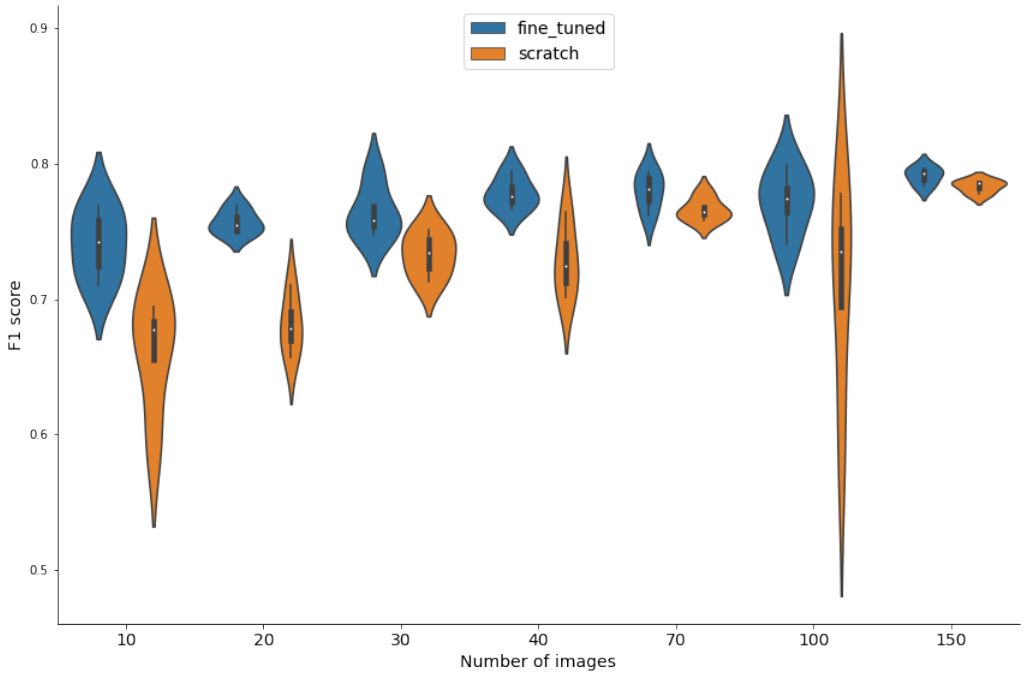


Figure 4.2: **Violin plots comparison**.The F_1 score for fine-tuned and scratch models. When a little fraction of images is available, fine-tuned models perform significantly better. Also, the score distribution of fine-tuned models is narrower than the scratch ones being more robust concerning the dataset fluctuations.

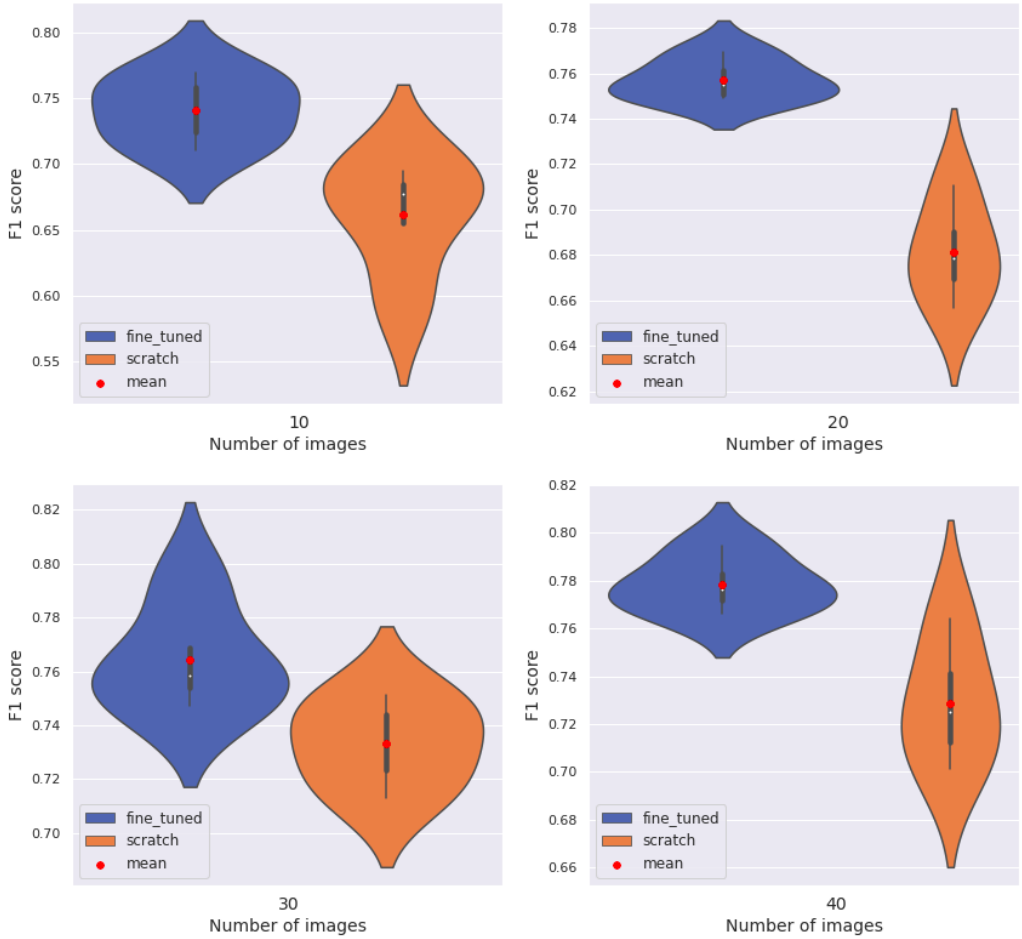


Figure 4.3: **Violin plots comparison**.The F_1 score is reported both for the fine-tuned and scratch-trained models varying the number of images. The red spot identifies the mean value that overlaps the median value if the distribution is symmetric. The orange plots (training from scratch) are often asymmetric due to the higher results variance.

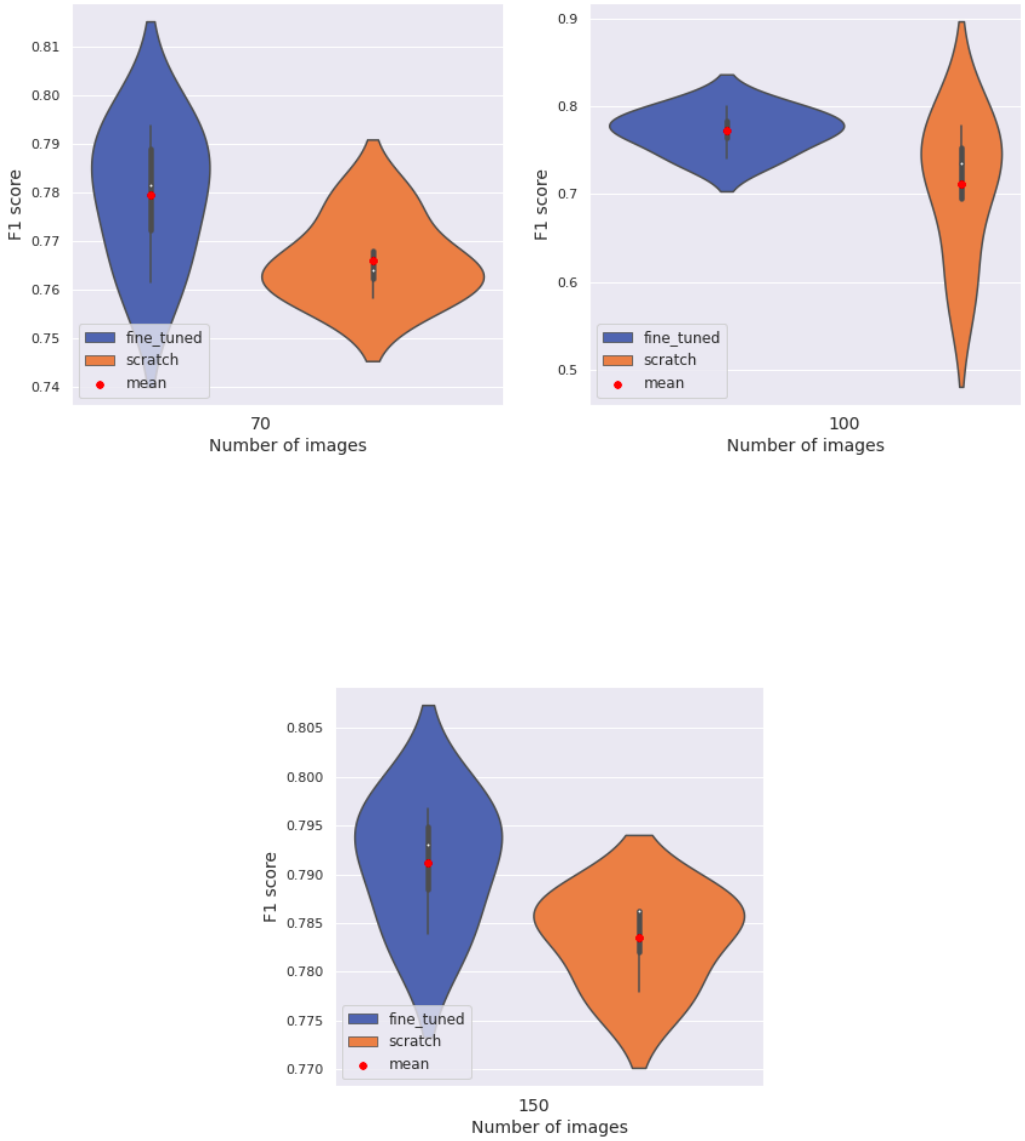


Figure 4.3: **Violin plots comparison**.The F_1 score is reported both for the fine-tuned and scratch-trained models varying the number of images. The red spot identifies the mean value that overlaps the median value if the distribution is symmetric. The orange plots (training from scratch) are often asymmetric due to the higher results variance.

4.5 Final Remarks

In this chapter, we applied a transfer learning procedure between two datasets belonging to the same fluorescent microscopy domain. We set as source and target datasets respectively the cTB and the c-Fos datasets. First, we quantify the optimal number of pre-trained epochs that, in this specific use case, is equal to the maximum value reached at the model pre-training end. This result stands for both the architectures tested that differ one from the other for the entity of the layers unfreezed during the fine-tuning phase. Then, we evaluate different fine-tuning strategies consisting of unfreezing distinct groups of layers in distinct training trials. From these experiments, we observed that it is preferable to unfreeze only the decoder path (including the bottleneck) during the fine-tuning stage.

Once we acquired this knowledge, we moved to compare systematically model trained from scratch and fine-tuned models in the function of the number of images available from the target dataset. For each number of selected images, we resampled four distinct subsets and trained the model on each of these subsamples. From these investigations, we observe that, especially under the few-shot regime, the fine-tuning procedure shows a significant performance boost. This gap gradually reduces increasing the number of images used, since the performance difference between these two model families goes from 8 points to 2 points in terms of F_1 score moving from the 10-images subset to the 70-images one. The gap definitively vanishes above 150 images. Comparing these results with the previous chapter we also observe that to reach the same results we only take 150 images instead of the 240 representing the full-size dataset. This means that the information contained inside the images became redundant after this number of pictures. Moreover, applying a fine-tuning procedure, we halved this number of images using only 70 images instead of 150 to reach the best F_1 score. Another relevant result is the robustness shown by the fine-tuned model concerning the fluctuations of the training dataset. While a model trained from scratch may be significantly affected by an *unlucky* subset, the fine-tuned face better this negative circumstance.

Chapter 5

Model deployment

5.1 Introduction

Deploying a deep learning model through a web application is a powerful way to make the model accessible and interactive for a wide range of users. Additionally, web applications provide for interactive interfaces that can display output and receive input from users, making it easier to demonstrate the capabilities of the model and receive feedback. To capitalize on the work described in the previous chapter, we dedicated the last part of this dissertation to a web app deployment. We aim to share our counting processing pipeline with the pre-trained models on both the datasets analysed: CTb and c-Fos. This introductory work is to mean as the basis of a future development that integrates the tool inside the cloud environment. So far, we build a demonstrator running via the web but not yet exposed via public IP address. We show what is already implemented and what we aim to implement for future work. All the code used for the implementation is reported at the following link: cell counting web app

5.2 Web-app

In the following section, we illustrate the web application functionalities together with some of the implementation details. The code is public and

available together with the pre-trained models at the link (cell counting app repository)

5.2.1 Web-app features

Functional requirements

In this section, are described the main functional and non-functional requirements of the web application. The application is designed to provide the following functionalities that represent the pre-processing, inference and post-processing steps already implemented in the previous part of this work to analyze the model performance:

- Uploading of one or more biological images at times
- Images normalization step
- Cell counting inference:
  - Heatmaps prediction
  - Output binarization by thresholding
  - Post-processing step
  - Neuronal cells counting image by image

All these steps are wrapped into a web-application dedicated to the analysis of fluorescent images. The expected end-user are all the researchers with little or no experience in the deep learning field that can easily upload their images to the application to run a counting retrieval of the neuronal cells contained in their custom data set. At this time, to test this application it is required to clone the repository and run the application as a local web service. In the next development, the cloud environment will be tested to expose the model via a public IP

Non-functional requirements

The non-functional requirements are listed in the following while other features to implement are reported as future step developments.

- Reachable on a local server
- Containerized environment
- Fast and easy deployment
- Responsive and easy to use

The next steps will be focused on the following steps:

- The application should be expose via public address on a cloud environment
- The application should be Managed via containers orchestrate such docker swarm or kubernetes.
- The application should be responsive and easy to use
- The application should be secure and protect user data
- The application should be able to handle multiple users simultaneously

5.2.2 Workflow

We designed the web application to perform three main steps throughout the interaction with a graphical interface. The end user can essentially provide the images for the analysis, load the pre-training model, set the post-processing parameters, and get the counting results. The entire workflow is reported in Fig. 5.0.

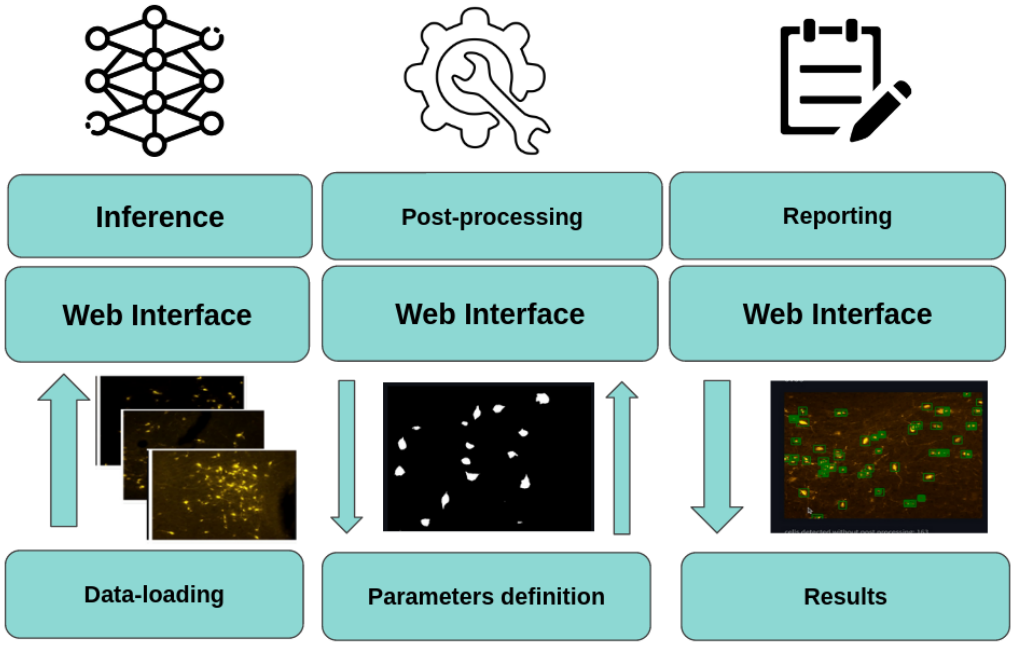


Figure 5.0: **Web app scheme**. After the images loading, the user get the preliminary results. A post-processing step che be performed after a real-time visual evaluation of the processing effects. Once defined, the visual (bounding-box) and counting results for each image are collected and send to the end-user.

Step-by-step demo

Going through each step, the first interaction provides the loading of the images as described in Fig. 5.1. The user can select one or multiple files. The files are then visualized on the app interface. It is possible to visualize also only a part of these images by selecting the number on the left side of the layout (*items to display* box). After that, using the box *select the model to load* we are going to load the pre-trained model to make the inference. At the current state, we have both the best model pre-trained on the yellow and green datasets (c-ResUnet-y and c-ResUnet-g).

Once we select the model, pressing the *make prediction button* (Fig. 5.1) the prediction stage starts. Each image is analyzed and the binary mask results are displayed (on the right of Fig. 5.2) along with a bounding box prediction (on the left of Fig. 5.2). However, as described in the previous

chapters, the threshold value used for binarization can affect the counting process. It is possible to change this parameter using the slide bar *confidence threshold* (Fig. 5.2) in real-time making a recomputation of the results. However, the prediction results (heatmaps) are already cached and only the last step of binarization is performed to save time and computational resources.

Then, in the last step, the user can apply a post-processing operation that resembles the same pipeline described in the previous chapters. In this case, we can observe from Fig. 5.3 how this processing can drastically affect the results and the importance to set appropriate values. Thanks to the user interface we can immediately visualize the results modifications due to a change in the parameter's values. From these feedbacks we can operate a fast fine-tuning procedure to get the optimal post-processing values. We suggest operating these first post-processing fine-tune on a subset of the images to speed up the operations. After this preliminary analysis, it is possible to load all the dataset to analyze and get a report that associate with each image the relative count.

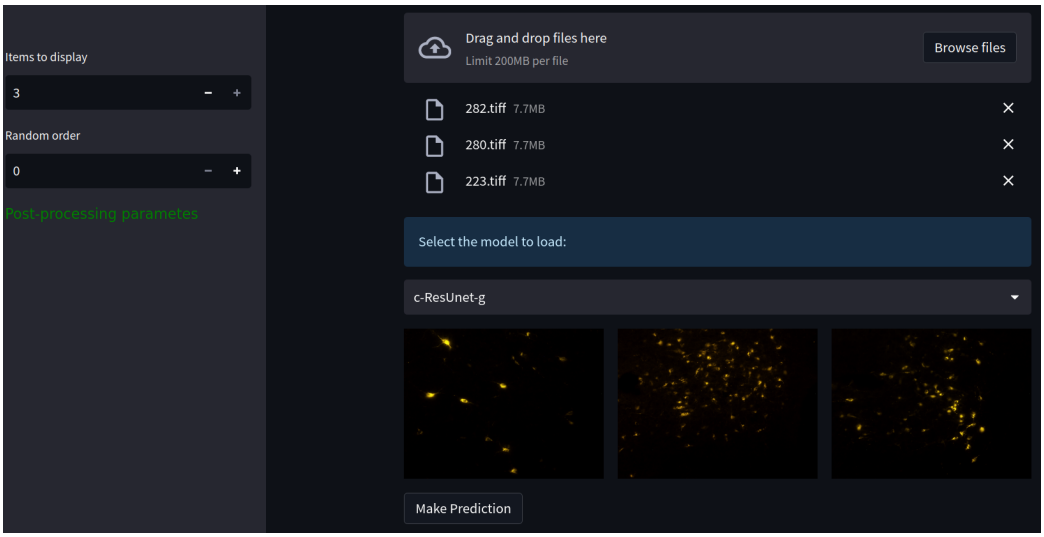


Figure 5.1: **Image loading**. The user can load one or multiple images.

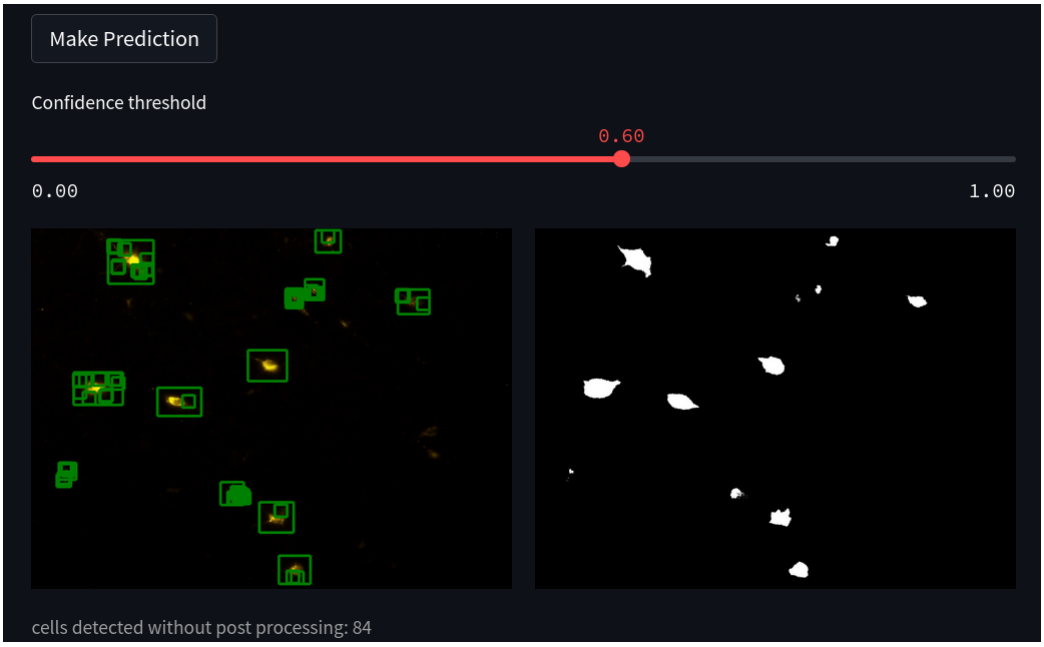


Figure 5.2: **Prediction**. Model inference. On the left the bounding box visualization. On the right the binary segmentation mask.

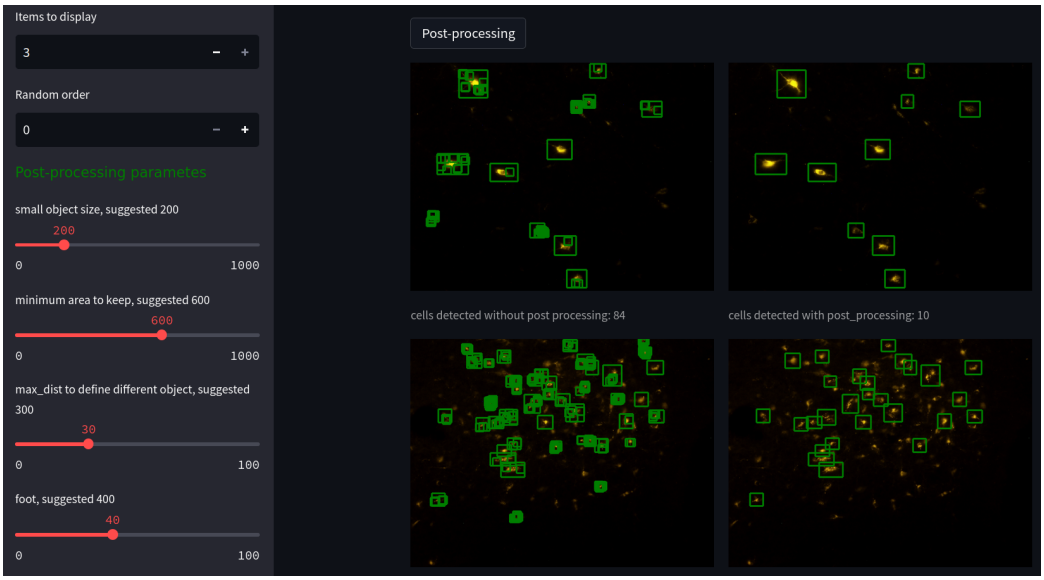


Figure 5.3: **Post-processing**. On the left, the images without post-processing. On the right the images after post-processing operation.

5.2.3 Implementation

We investigated some alternative among the libraries frequently used to develop web-application:

- Flask: a lightweight web framework that is easy to use and allows for the development of small to medium-sized web applications.
- Django: a high-level web framework that is powerful and flexible, and is well-suited for the development of large, complex web applications.
- Pyramid: a web framework that is similar to Flask and Django and is often used for large, complex web applications.
- FastAPI: a modern, fast, web framework for building APIs with Python 3.6+ based on standard Python type hints.
- Tornado: a web framework and asynchronous networking library, which can handle large numbers of simultaneous connections.
- CherryPy: a minimalistic web framework that is easy to use and allows for the development of small to medium-sized web applications.
- Streamlit is a popular open-source library that is specifically designed for creating interactive web-based applications for machine learning and data science. It's easy to use and allows developers to build web applications quickly without the need for any JavaScript or HTML/CSS knowledge. It's a great tool for data visualization, exploratory data analysis, and building simple machine learning models without the need for a web developer.

Finally, we opt for Streamlit which is a relatively new library but it's gaining popularity fast. Its ease of use makes it a good fit for our proof-of-concept. To promote the usability of the application, we provide in the code repository the instruction to deploy it using container technology. For this purpose, we use a Docker container that relies on the Dockerfile recipe for the automatization of the build and start-up of the web application container.

The power of the container is especially related to the increased reproducibility across different platforms that include also the cloud environment.

5.2.4 Model deployment phases

Deploying a deep learning model involves several steps. The first step regards the training of the model. This process involves choosing an appropriate algorithm, determining the parameters, and exposing it to pre-processed and cleansed data. For this step, we exploit the model trained in the previous chapters.

After training and validation, we need to integrate the model into a process that is our counting pipeline and make this service available to the end user. This can include, for example, making it accessible from an end user's laptop using an API or integrating it into software currently being used by the end user. Indeed, there exist different approach to make the model and the processing pipeline available:

- Web Application: deploying a deep learning model as a web application allows for easy access and interaction with the model, and can be integrated with other systems and services;

- Mobile Application: deep learning models can also be deployed as mobile applications, allowing users to access the model from their mobile devices;

- Cloud Services: Cloud services like AWS, Google Cloud, and Microsoft Azure provide tools and infrastructure for deploying deep learning models in the cloud. This is a popular option because it allows for easy scaling and management of the model;

- On-Premises: The model can be deployed on-premises, meaning it runs on a server or computers within the organization's own facilities, this can be useful for organizations that have sensitive data and have strict security requirements.

- Edge devices: deep learning models can be deployed on edge devices, such as IoT devices and embedded systems, which are devices that are located at the edge of the network, closest to the source of data, this allows for real-time processing and low-latency.
- API: A deep learning model can be deployed as an API (Application Programming Interface) which allows other applications to access the model's functionality through a simple interface.

For this proof-of-concept, we target the implementation of a front-end deep-learning web application. We developed such a tool that utilizes deep learning models on the front-end, or client side, meaning that the deep learning models and computations are performed within the user's web browser, rather than on a remote server. This approach allows for faster and more responsive user experiences, as the model does not have to communicate with a remote server to make predictions but is constrained by the end-user computation hardware.

Nevertheless, it's worth remembering that web-app deployment in a cloud environment is always possible. To implement this step is essentially required to have a public IP available to make the application service public reachable. However, we don't address this task in this work but we facilitate this future step in developing the web app using container technology such as Docker. Indeed, in this manner, the web app may potentially be shipped across the different platforms for the deployment step, including cloud infrastructure. Then, to optimize the orchestration of the application, the docker-swarm utility may be used to make the application compliant with the user-request demand. Also, this latter scope is included as a future-work development.

5.3 Final remarks

We developed a proof-of-concept web application to serve the pre-trained models trained during the investigations described in this thesis work. We decided to deploy a model accessible from the client's side web browser to design a proof-of-concept demonostrator. The next step not addressed in

this thesis, will be aimed to deploy the application on the cloud environment exploiting the container technology. From the functionalities point of view, the next step aims to enable the training and transfer learning funcionalities. For the former option, the user is allowed to load its own dataset and start training on those images. For the latter scope, we target to allow for a fine-tuning on custom dataset provided by the users starting from a pre-trained model. Moreover, dataset management system should be provided with systematic cleaning and storing procedures. All these steps will be faced in the next steps of the web app development.

Chapter 6

Conclusion

In this thesis, we developed a pipeline for cell counting in fluorescent microscopy images. We followed the counting-by-segmentation approach that requires ground-truth segmentation masks under the supervised learning framework. We combined the main features of two state-of-the-art architectures, UNet and ResUnet, to train a model with improved counting performances. We improved the segmentation output on the high-density cell area increasing the field of view of the model. We also designed a novel weighted maps generation strategy to penalize the loss as a function of the cell's crowding. These modifications prove their effectiveness through ablation studies. Then, we framed out the counting-by-segmentation problem under a weakly-supervised learning approach. We provided two distinct use cases adapting the pseudo-labels generation to the dataset characteristics. We compared supervised and weakly-supervised performances on the fluorescent neuronal cells dataset obtaining similar results and validating the weakly-supervised approach. Finally, we investigated the transfer learning approach exploiting the two datasets and showing the efficacy of fine-tuning method to improve significantly the model performances under a few-shot learning regime. In the end, we provided an example of a web app deployed to share the pre-training model and the entire counting processing pipeline.

Bibliography

Mohammad Mahmudul Alam and Mohammad Tariqul Islam. Machine learning approach of automatic identification and counting of blood cells. *Healthcare technology letters*, 6(4):103–108, 2019.

Laith Alzubaidi, Muthana Al-Amidie, Ahmed Al-Asadi, Amjad J Humaidi, Omran Al-Shamma, Mohammed A Fadhel, Jinglan Zhang, J Santamaría, and Ye Duan. Novel transfer learning approach for medical imaging with limited labeled data. *Cancers*, 13(7):1590, 2021.

Carlos Arteta, Victor Lempitsky, and Andrew Zisserman. Counting in the wild. volume 9911, pages 483–498, 10 2016. ISBN 978-3-319-46477-0. doi: 10.1007/978-3-319-46478-7_30.

Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. What's the point: Semantic segmentation with point supervision. In *European conference on computer vision*, pages 549–565. Springer, 2016.

Ronald Bellamy, Peter Safar, Samuel A Tisherman, Robert Basford, Stephen P Bruttig, Antonio Capone, Michael A Dubick, Lars Ernster, G Brack Jr, Peter Hochachka, et al. Suspended animation for delayed resuscitation. *Critical care medicine*, 24(2):24S–47S, 1996.

Hjalmar R Bouma, Esther M Verhaag, Jessica P Otis, Gerhard Heldmaier, Steven J Swoap, Arjen M Strijkstra, Robert H Henning, and Hannah V Carey. Induction of torpor: mimicking natural metabolic suppression for biomedical applications. *Journal of cellular physiology*, 227(4):1285–1290, 2012.

Yue Cao, Shigang Liu, Yali Peng, and Jun Li. Denseunet: densely connected unet for electron microscopy image segmentation. *IET Image Processing*, 14(12):2682–2689, 2020.

Matteo Cerri, Walter Tinganelli, Matteo Negrini, Alexander Helm, Emanuele Scifoni, Francesco Tommasino, Maximiliano Sioli, Antonio Zoccoli, and Marco Durante. Hibernation for space travel: Impact on radioprotection. *Life Sciences in Space Research*, 11:1–9, 2016. ISSN 2214-5524. doi: https://doi.org/10.1016/j.lssr.2016.09.001. URL https://www.sciencedirect.com/science/article/pii/S2214552416300542.

Matteo Cerri, Timna Hitrec, Marco Luppi, and Roberto Amici. Be cool to be far: Exploiting hibernation for space exploration. *Neuroscience & Biobehavioral Reviews*, 128:218–232, 2021. ISSN 0149-7634. doi: https://doi.org/10.1016/j.neubiorev.2021.03.037. URL https://www.sciencedirect.com/science/article/pii/S0149763421002645.

Alireza Chamanzar and Yao Nie. Weakly supervised multi-task learning for cell detection and segmentation. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pages 513–516. IEEE, 2020.

Joseph Y. Cheng, Feiyu Chen, M. Alley, J. Pauly, and S. Vasanawala. Highly scalable image reconstruction using deep neural networks with bandpass filtering. *ArXiv*, abs/1805.03300, 2018.

Dan Ciresan, Alessandro Giusti, Luca Maria Gambardella, and Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. *Proceedings of Neural Information Processing Systems*, 25, 01 2012.

Dan Ciresan, Alessandro Giusti, Luca Maria Gambardella, and Jurgen Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. volume 16, pages 411–8, 09 2013. ISBN 978-3-642-40762-8. doi: 10.1007/978-3-642-40763-5_51.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Luca Clissa. *Supporting Scientific Research Through Machine and Deep Learning: Fluorescence Microscopy and Operational Intelligence Use Cases*. PhD thesis, alma, Giugno 2022. URL http://amsdottorato.unibo.it/10016/.

Ellen Paula Santos da Conceição, Shaun F Morrison, Georgina Cano, Pierfrancesco Chiavetta, and Domenico Tupone. Median preoptic area neurons are required for the cooling and febrile activations of brown adipose tissue thermogenesis in rat. *Scientific reports*, 10(1):1–16, 2020.

Jifeng Dai, Kaiming He, and Jian Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1635–1643, 2015.

Daniela Dentico, Roberto Amici, Francesca Baracchi, Matteo Cerri, Elide Del Sindaco, Marco Luppi, Davide Martelli, Emanuele Perez, and Giovanni Zamboni. C-fos expression in preoptic nuclei as a marker of sleep rebound in the rat. *European Journal of Neuroscience*, 30(4):651–661, 2009. ISSN 0953816X. doi: 10.1111/j.1460-9568.2009.06848.x.

RCNN Faster. Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 9199(10.5555): 2969239–2969250, 2015.

G. M. Faustino, M. Gattass, S. Rehen, and C. J. P. de Lucena. Automatic embryonic stem cells detection and counting method in fluorescence microscopy images. In *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 799–802, 2009. doi: 10.1109/ISBI.2009.5193170.

Richard Gillis, Gary Adams, David Besong, Eva Machova, Anna Ebringerova, Stephen Harding, and Trushar Patel. Phosphorylated tau

protein in the myenteric plexus of the ileum and colon of normothermic rats and during synthetic torpor. *European Biophysics Journal*, 2016. ISSN 0175-7571.

H. Greenspan, B. van Ginneken, and R. M. Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159, 2016. doi: 10.1109/TMI.2016.2553401.

Steven Guan, Amir A. Khan, Siddhartha Sikdar, and Parag V. Chitnis. Fully dense unet for 2-d sparse photoacoustic tomography artifact removal. *IEEE Journal of Biomedical and Health Informatics*, 24(2):568–576, 2020. doi: 10.1109/JBHI.2019.2912935.

Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4805–4814, 2019.

Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain tumor segmentation with deep neural networks. *Medical Image Analysis*, 35:18–31, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 06 2016. doi: 10.1109/CVPR.2016.90.

Shenghua He, Kyaw Thu Minn, Lilianna Solnica-Krezel, Mark A Anastasio, and Hua Li. Deeply-supervised density regression for automatic cell counting in microscopy images. *Medical Image Analysis*, 68:101892, 2021.

Carlos X Hernández, Mohammad M Sultan, and Vijay S Pande. Using deep learning for segmentation and counting within microscopy data. *arXiv preprint arXiv:1802.10548*, 2018.

Timna Hitrec, Marco Luppi, Stefano Bastianini, Fabio Squarcio, Chiara Berteotti, Viviana Lo Martire, Davide Martelli, Alessandra Occhinegro, Domenico Tupone, Giovanna Zoccoli, Roberto Amici, and Matteo Cerri. Neural control of fasting-induced torpor in mice. *Scientific Reports*, 9(1), oct 2019. doi: 10.1038/s41598-019-51841-2.

Timna Hitrec, Fabio Squarcio, Matteo Cerri, Davide Martelli, Alessandra Occhinegro, Emiliana Piscitiello, Domenico Tupone, Roberto Amici, and Marco Luppi. Reversible tau phosphorylation induced by synthetic torpor in the spinal cord of the rat. *Frontiers in neuroanatomy*, 15:3, 2021.

Michael Hobley and Victor Prisacariu. Learning to count anything: Reference-less class-agnostic counting with weak supervision. *arXiv preprint arXiv:2205.10203*, 2022.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

Jeroen Hoekendijk, Benjamin Kellenberger, Geert Aarts, Sophie Brasseur, Suzanne SH Poiesz, and Devis Tuia. Counting using deep learning regression gives value to ecological surveys. *Scientific reports*, 11(1):1–12, 2021.

Haigen Hu, Yixing Zheng, Qianwei Zhou, Jie Xiao, Shengyong Chen, and Qiu Guan. Mc-unet: Multi-scale convolution unet for bladder cancer cell segmentation in phase-contrast microscopy images. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1197–1199. IEEE, 2019.

Pipei Huang, Gang Wang, and Shiyin Qin. Boosting for transfer learning from multiple data sources. *Pattern Recognition Letters*, 33(5):568–579, 2012.

Debesh Jha, Pia H Smedsrud, Michael A Riegler, Dag Johansen, Thomas De Lange, Pål Halvorsen, and Håvard D Johansen. Resunet++: An ad-

vanced architecture for medical image segmentation. In *2019 IEEE International Symposium on Multimedia (ISM)*, pages 225–2255. IEEE, 2019.

Debesh Jha, Pia H Smedsrud, Dag Johansen, Thomas de Lange, Håvard D Johansen, Pål Halvorsen, and Michael A Riegler. A comprehensive study on colorectal polyp segmentation with resunet++, conditional random field and test-time augmentation. *IEEE journal of biomedical and health informatics*, 25(6):2029–2040, 2021.

Hongyang Jiang, He Ma, Wei Qian, Mengdi Gao, and Yan Li. An automatic detection system of lung nodule based on multigroup patch-based deep learning network. *IEEE Journal of Biomedical and Health Informatics*, 22 (4):1227–1237, 2018. doi: 10.1109/JBHI.2017.2725903.

Oscar Jimenez-del Toro, Juan Otálora Montenegro, Mats Andersson, Kristian Euren, Martin Hedlund, Mikael Rousson, Henning Müller, and Manfredo Atzori. *Analysis of Histopathology Images*, pages 281–314. 01 2017. ISBN 9780128121337. doi: 10.1016/B978-0-12-812133-7.00010-7.

Nabeel Khalid, Fabian Schmeisser, Mohammadmahdi Koochali, Mohsin Munir, Christoffer Edlund, Timothy R Jackson, Johan Trygg, Rickard Sjögren, Andreas Dengel, and Sheraz Ahmed. Point2mask: A weakly supervised approach for cell segmentation using point annotation. In *Annual Conference on Medical Image Understanding and Analysis*, pages 139–153. Springer, 2022.

Martin Kolařík, Radim Burget, Václav Uher, Kamil Říha, and Malay Kishore Dutta. Optimized high resolution 3d dense-u-net network for brain and spine segmentation. *Applied Sciences*, 9(3):404, 2019.

Alexander Kolesnikov and Christoph H Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *European conference on computer vision*, pages 695–711. Springer, 2016.

Bruno Korbar, Andrea Olofson, Allen Miraflor, Katherine Nicka, Matthew Suriawinata, Lorenzo Torresani, Arief Suriawinata, and Saeed Hassanpour.

Deep-learning for classification of colorectal polyps on whole-slide images. *Journal of Pathology Informatics*, 8, 03 2017. doi: 10.4103/jpi.jpi_34_17.

Oren Kraus, Jimmy Ba, and Brendan Frey. Classifying and segmenting microscopy images with deep multiple instance learning. *Bioinformatics*, 32: i52–i59, 06 2016. doi: 10.1093/bioinformatics/btw252.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012. doi: 10.1145/3065386.

Thomas Küstner, Tobias Hepp, Marc Fischer, Martin Schwartz, Andreas Fritsche, Hans-Ulrich Häring, Konstantin Nikolaou, Fabian Bamberg, Bin Yang, Fritz Schick, et al. Fully automated and standardized segmentation of adipose tissue compartments via deep learning in 3d whole-body mri of epidemiologic cohort studies. *Radiology: Artificial Intelligence*, 2(6), 2020.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Yann LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 20(5):14, 2015.

Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. pages 105–114, 07 2017. doi: 10.1109/CVPR.2017.19.

Jungbeom Lee, Jihun Yi, Chaehun Shin, and Sungroh Yoon. Bbam: Bounding box attribution map for weakly supervised semantic and instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2643–2652, 2021.

Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. *arXiv preprint arXiv:2210.11466*, 2022.

Yinjie Lei, Yan Liu, Pingping Zhang, and Lingqiao Liu. Towards using count-level weak supervision for crowd counting. *Pattern Recognition*, 109: 107616, 2021.

Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper/2010/file/fe73f687e5bc5280214e0486b273a5f9-Paper.pdf.

Jean Le'Clerc Arrastia, Nick Heilenkötter, Daniel Otero Baguer, Lena Hauberg-Lotte, Tobias Boskamp, Sonja Hetzer, Nicole Duschner, Jörg Schaller, and Peter Maass. Deeply supervised unet for semantic segmentation to assist dermatopathological assessment of basal cell carcinoma. *Journal of imaging*, 7(4):71, 2021.

Qizhu Li, Anurag Arnab, and Philip HS Torr. Weakly-and semi-supervised panoptic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 102–118, 2018.

Xuhong Li, Yves Grandvalet, Franck Davoine, Jingchun Cheng, Yin Cui, Hang Zhang, Serge Belongie, Yi-Hsuan Tsai, and Ming-Hsuan Yang. Transfer learning in computer vision tasks: Remember where you come from. *Image and Vision Computing*, 93:103853, 2020.

Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on mri. *Zeitschrift fur Medizinische Physik*, 29(2):102–127, 2019. ISSN 18764436. doi: 10.1016/j.zemedi.2018.11.002. URL https://doi.org/10.1016/j.zemedi.2018.11.002.

Marco Luppi, Matteo Cerri, Alessia Di Cristoforo, Timna Hitrec, Daniela Dentico, Flavia Del Vecchio, Davide Martelli, Emanuele Perez, Domenico

Tupone, Giovanni Zamboni, and Roberto Amici. c-fos expression in the limbic thalamus following thermoregulatory and wake–sleep changes in the rat. *Experimental Brain Research*, 237(6):1397–1407, 2019. ISSN 14321106. doi: 10.1007/s00221-019-05521-2. URL http://dx.doi.org/10.1007/s00221-019-05521-2.

R Austin McEver and BS Manjunath. Pcams: Weakly supervised semantic segmentation using point supervision. *arXiv preprint arXiv:2007.05615*, 2020.

Talha Meraj, Hafiz Tayyab Rauf, Saliha Zahoor, Arslan Hassan, M IkramUllah Lali, Liaqat Ali, Syed Ahmad Chan Bukhari, and Umar Shoaib. Lung nodules detection using semantic segmentation and classification with optimal features. *Neural Computing and Applications*, pages 1–14, 2020.

Roberto Morelli, Luca Clissa, Roberto Amici, Matteo Cerri, Timna Hitrec, Marco Luppi, Lorenzo Rinaldi, Fabio Squarcio, and Antonio Zoccoli. Automating cell counting in fluorescent microscopy through deep learning with c-ResUnet. *Scientific Reports*, 11(1):22920, 2021a. doi: 10.1038/s41598-021-01929-5.

Roberto Morelli, Luca Clissa, Marco Dalla, Marco Luppi, Lorenzo Rinaldi, and Antonio Zoccoli. Automatic cell counting in flourescent microscopy using deep learning. *arXiv preprint arXiv:2103.01141*, 2021b.

Kazuya Nishimura, Dai Fei Elmer Ker, and Ryoma Bise. Weakly supervised cell instance segmentation by propagating from detection response. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 649–657. Springer, 2019.

Seong Joon Oh, Rodrigo Benenson, Anna Khoreva, Zeynep Akata, Mario Fritz, and Bernt Schiele. Exploiting saliency for object segmentation from image level labels. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pages 5038–5047. IEEE, 2017.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE transactions on neural networks*, 22(2):199–210, 2010.

Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Training object class detectors with click supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6374–6383, 2017a.

Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Extreme clicking for efficient object annotation. In *Proceedings of the IEEE international conference on computer vision*, pages 4930–4939, 2017b.

George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1742–1750, 2015.

Joseph Paul Cohen, Genevieve Boucher, Craig A Glastonbury, Henry Z Lo, and Yoshua Bengio. Count-ception: Counting by fully convolutional redundant counting. In *Proceedings of the IEEE International conference on computer vision workshops*, pages 18–26, 2017.

Anggraeini Puspitasari, Matteo Cerri, Akihisa Takahashi, Yukari Yoshida, Kenji Hanamura, and Walter Tinganelli. Hibernation as a tool for radiation protection in space exploration. *Life*, 11(1):54, 2021.

Saqib Qamar, Hai Jin, Ran Zheng, Parvez Ahmad, and Mohd Usama. A variant form of 3d-unet for infant brain segmentation. *Future Generation Computer Systems*, 108:613–623, 2020.

Hui Qu, Pengxiang Wu, Qiaoying Huang, Jingru Yi, Zhennan Yan, Kang Li, Gregory M Riedlinger, Subhajyoti De, Shaoting Zhang, and Dimitris N Metaxas. Weakly supervised deep nuclei segmentation using partial points annotation in histopathology images. *IEEE transactions on medical imaging*, 39(11):3655–3666, 2020.

Maryam Rahnemoonfar and Clay Sheppard. Deep count: Fruit counting based on deep simulated learning. *Sensors*, 17(4):905, Apr 2017. ISSN 1424-8220. doi: 10.3390/s17040905. URL http://dx.doi.org/10.3390/s17040905.

Shan e Ahmed Raza, Linda Cheung, David Epstein, Stella Pelengaris, Michael Khan, and Nasir Rajpoot. Mimo-net: A multi-input multi-output convolutional neural network for cell segmentation in fluorescence microscopy images. pages 337–340, 04 2017. doi: 10.1109/ISBI.2017.7950532.

Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL http://arxiv.org/abs/1506.02640.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. pages 779–788, 06 2016. doi: 10.1109/CVPR.2016.91.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. volume 9351, pages 234–241, 10 2015. ISBN 978-3-319-24573-7. doi: 10.1007/978-3-319-24574-4_28.

Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.

B. Sahiner, Heang-Ping Chan, N. Petrick, Datong Wei, M. A. Helvie, D. D. Adler, and M. M. Goodsitt. Classification of mass and normal breast tissue: a convolution neural network classifier with spatial domain and texture images. *IEEE Transactions on Medical Imaging*, 15(5):598–610, 1996. doi: 10.1109/42.538937.

Michael J Sanderson, Ian Smith, Ian Parker, and Martin D Bootman. Fluorescence microscopy. *Cold Spring Harbor Protocols*, 2014(10):pdb–top071795, 2014.

S. Segui, O. Pujol, and J. Vitria. Learning to count with deep object features. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 90–96, Los Alamitos, CA, USA, jun 2015. IEEE Computer Society. doi: 10.1109/CVPRW.2015.7301276. URL https://doi.ieeecomputersociety.org/10.1109/CVPRW.2015.7301276.

K. Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

Pierre J Soille and Marc M Ansoult. Automated basin delineation from digital elevation models using mathematical morphology. *Signal Processing*, 20(2):171–182, 1990.

Ying Su, Dan Li, and Xiaodong Chen. Lung nodule detection based on faster r-cnn framework. *Computer Methods and Programs in Biomedicine*, 200: 105866, 2021.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

Walter Tinganelli, Timna Hitrec, Fabrizio Romani, Palma Simoniello, Fabio Squarcio, Agnese Stanzani, Emiliana Piscitiello, Valentina Marchesano, Marco Luppi, Maximiliano Sioli, et al. Hibernation and radioprotection: gene expression in the liver and testicle of rats irradiated under synthetic torpor. *International journal of molecular sciences*, 20(2):352, 2019.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

Michel Vandenberghe, Marietta Scott, Paul Scorer, Magnus Soderberg, Denis Balcerzak, and Craig Barker. Relevance of deep learning to facilitate the diagnosis of her2 status in breast cancer open. *Scientific Reports*, 7, 05 2017. doi: 10.1038/srep45938.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Juan P Vigueras-Guillén, Jeroen van Rooij, Bart TH van Dooren, Hans G Lemij, Esma Islamaj, Lucas J van Vliet, and Koenraad A Vermeer. Dense-unets with feedback non-local attention for the segmentation of specular microscopy images of the corneal endothelium with guttae. *Scientific Reports*, 12(1):1–12, 2022.

Haonan Wang, Peng Cao, Jiaqi Wang, and Osmar R Zaiane. Uctransnet: rethinking the skip connections in u-net from a channel-wise perspective with transformer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 2441–2449, 2022.

Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

Yu Weng, Tianbao Zhou, Yujie Li, and Xiaoyu Qiu. Nas-unet: Neural architecture search for medical image segmentation. *IEEE Access*, 7:44247–44257, 2019.

Lian Xu, Wanli Ouyang, Mohammed Bennamoun, Farid Boussaid, Ferdous Sohel, and Dan Xu. Leveraging auxiliary tasks with affinity learning for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6984–6993, 2021.

Yonghui Xu, Sinno Jialin Pan, Hui Xiong, Qingyao Wu, Ronghua Luo, Huaqing Min, and Hengjie Song. A unified framework for metric transfer

learning. *IEEE Transactions on Knowledge and Data Engineering*, 29(6): 1158–1171, 2017.

Yao Xue, Nilanjan Ray, Judith Hugh, and Gilbert Bigras. Cell counting by regression using convolutional neural network. pages 274–290, 2016.

Samir S. Yadav and Shivajirao M. Jadhav. Deep convolutional neural network based medical image classification for disease diagnosis. *Journal of Big Data*, 6(1), 2019. ISSN 21961115. doi: 10.1186/s40537-019-0276-2. URL https://doi.org/10.1186/s40537-019-0276-2.

Yi Yao and Gianfranco Doretto. Boosting for transfer learning with multiple sources. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 1855–1862. IEEE, 2010.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

Zitao Zeng, Weihao Xie, Yunzhe Zhang, and Yao Lu. Ric-unet: An improved neural network based on unet for nuclei segmentation in histology images. *Ieee Access*, 7:21420–21428, 2019.

Yanzhao Zhou, Yi Zhu, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Weakly supervised instance segmentation using class peak response. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3791–3800, 2018.

Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53, 2018.