ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Dottorato di Ricerca in Data Science and Computation

Supporting Scientific Research Through Machine and Deep Learning: Fluorescence Microscopy and Operational Intelligence Use Cases

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

Settore Scientifico Disciplinare:
FIS/01 Fisica Sperimentale

Coordinatore Dottorato
Prof. Andrea Cavalli

Candidato
Luca Clissa

Supervisore
Prof. Antonio Zoccoli

Co-supervisore
Prof. Lorenzo Rinaldi

XXXIII Ciclo
Esame Finale Anno 2022

Abstract

Although the debate of what data science is has a long history and has not reached a complete consensus yet, *Data Science* can be summarized as the process of learning from data. Guided by the above vision, this thesis presents two independent data science projects developed in the scope of multidisciplinary applied research.

The first part analyzes fluorescence microscopy images typically produced in life science experiments, where the objective is to count how many marked neuronal cells are present in each image. Aiming to automate the task for supporting research in the area, we propose a neural network architecture tuned specifically for this use case, *cell ResUnet* (*c-ResUnet*), and discuss the impact of alternative training strategies in overcoming particular challenges of our data. The approach provides good results in terms of both detection and counting, showing performance comparable to the interpretation of human operators. As a meaningful addition, we release the pre-trained model and the *Fluorescent Neuronal Cells* dataset collecting pixel-level annotations of where neuronal cells are located. In this way, we hope to help future research in the area and foster innovative methodologies for tackling similar problems.

The second part deals with the problem of distributed data management in the context of LHC experiments, with a focus on supporting ATLAS operations concerning data transfer failures. In particular, we analyze error messages produced by failed transfers and propose a Machine Learning pipeline that leverages the *word2vec* language model and *K-means* clustering. This provides groups of similar errors that are presented to human operators as suggestions of potential issues to investigate. The approach is demonstrated on one full day of data, showing promising ability in understanding the message content and providing meaningful groupings, in line with previously reported incidents by human operators.

To my parents, who were my foremost motivation.
To don Bruno, who taught me that even mistakes may lead to some good when done with the right intent.

The future of data analysis can involve great progress, the overcoming of real difficulties, and the provision of a great service to all fields of science and technology. Will it? That remains to us, to our willingness to take up the rocky road of real problems in preference to the smooth road of unreal assumptions, arbitrary criteria, and abstract results without real attachments. Who is for the challenge?

John Wilder Tukey, 1962

Acknowledgments

Undertaking a PhD program in an interdisciplinary field and focusing on various research areas at the intersection of different application domains has presented several challenges but also thrilling growth opportunities.

For this reason, I am deeply grateful to my supervisor, Antonio Zoccoli, who believed in me to initiate a line of research centered around artificial intelligence and data science in Bologna. Without his trust, support and vision, this would not have been possible, and I am sure my professional and personal life would have been much different. Another big thank you goes to my co-supervisor, Lorenzo Rinaldi, for his advice, continuous support, and patience during my PhD study. I truly appreciated his open-mindedness when considering my requests regarding research topics and training opportunities. Even more, his trust in my technical and organizational capabilities was fundamental during the writing of the thesis, allowing me to work peacefully at my own pace and deliver good results despite tight deadlines.

Then, I am profoundly grateful to the INFN and ATLAS Bologna, from the coordinators to all the nice people I had the luck to share my daily activity with. They created a productive work environment and provided me with many opportunities, instruments, and financial resources that enriched my research activities. A special mention goes to Benedetto Giacobbe and Roberto Spighi for helping me shaping better my research project; Mauro Villa for his support in the first part of my PhD; Roberto Morelli for our fruitful discussions and his crucial contributions to our research; Marco Dalla, with whom I shared part of my academic activities; Silvia Biondi and Giuseppe Carratta, for their selfless help and for being of *key* importance; Riccardo Ridolfi for his competent and thoughtful support concerning academy and its mechanisms; Daniele Manuzzi and Serena Maccolini for sharing lovely moments in the office.

A warm thank you then goes to Marco Luppi and all his team, particularly Fabio Squarcio and Timna Hitrec, for their collaboration in an exciting interdisciplinary project. My gratitude also extends to the whole Operational Intelligence collaboration that sustained me during my stay at CERN. In particular, Alessandro Di Girolamo and Federica Legger for their guidance and coordination; Mario

Lassnig for his wise indications and timely supervision; Mayank Sharma and Panos Paparrigopoulos for their precious advice about coding; Jaroslava Schovancová for her kindness and collaborative spirit, and Maria Grigorieva for the compelling discussions about our research projects.

A big thank you then goes to Iota for her precious aid with the bureaucracy. Amid many hardships of a nascent PhD program, her extreme availability and kindness made my experience much smoother. Her care for each of us was priceless, and I wish all PhD students to encounter a figure like her in their paths.

Finally, the most heartfelt acknowledgment goes to all my friends and family, who shared with me much more than my academic journey. To my friend and colleague Antonio, without whom I would have never applied for this PhD. More than that, his hard work and will to overcome hampers have been inspiring to me, and our collaboration gave me a real sense of purpose that helped me go through the ups and downs of my research path. To my dear Elena, for always being on my side, cheering me up and accommodating my needs. Her work ethic and dedication constituted a shining example of how to navigate through the uncertainty of life, set goals and commit to them. Knowing I can always count on her in moments of need has been reassuring during these years. To my lifelong friends, who are always there for me despite the distance and the little time spent together. Last but not least, the most profound, passionate and deserved thanksgiving is for my parents and my two brothers. They ensured me a life of dreams through their hard work and waivers. Their unconditional love, trust and support allowed me to proceed flawlessly, focusing on my goals without worries and distractions.

I sincerely believe I deserve little merit for this achievement, for which I owe the greatest part of the credit to all these people who contributed with their hopes, advice, trust and efforts. None of this would have been possible without them.

Contents

List of Figures

List of Tables

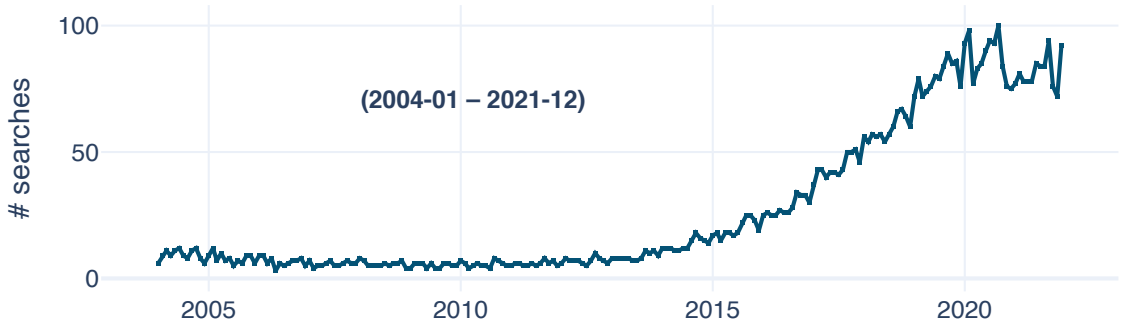
Chapter 1

Introduction

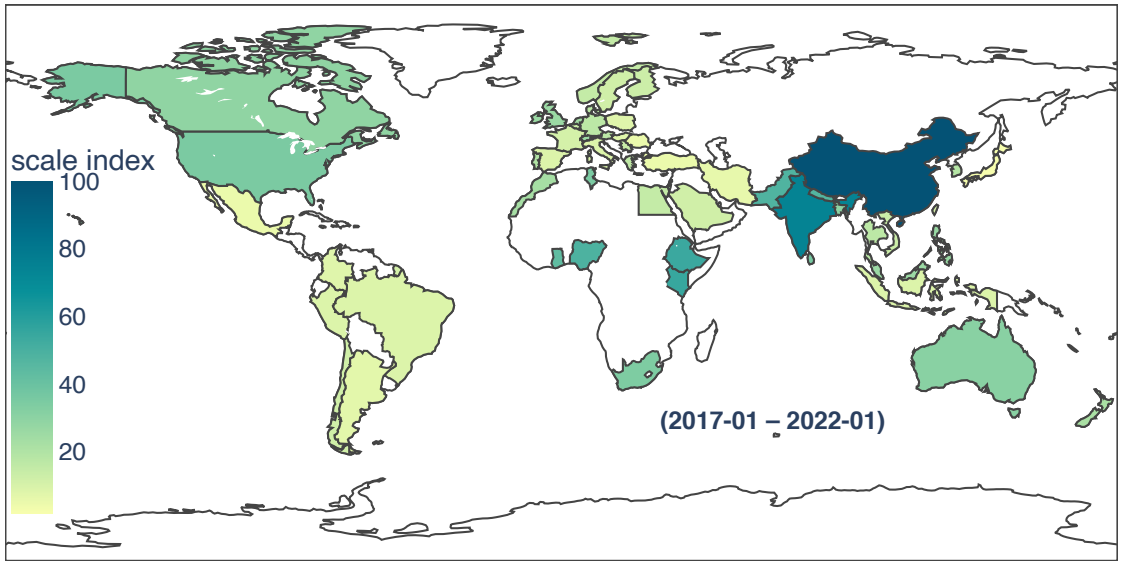
Data Science is a very vibrant field of research that has been gaining more and more interest across all the world in the past decade (Fig. 1.1), both in the educational and industrial sectors. That much so that it was awarded the title of "sexiest job of the 21st century" (Davenport & Patil, 2012), and only American universities counted 78 data science programs in 2020 (Z. Zhang & Zhang, 2021).

However, the discussion over data science's essence has a long history, and multiple definitions have been proposed over the years (Donoho, 2017). Although researchers and practitioners are yet to reach a complete agreement on its exact meaning (van Dyk et al., 2015), *five* common pillars can be identified by the various definitions. First, **multidisciplinarity** is indisputably a key element stressed in every definition of data science. Second, as the name suggests, the **focus on data** and adequate techniques to manage and process them is inevitably an essential aspect. Third, data science requires adopting suitable **statistical models** to convert data into knowledge. Fourth, the **computing infrastructure** that is necessary to manage the data and analyze them efficiently. Fifth, a compelling **visualization and communication** of results that is simple enough to be understood by a heterogeneous and non-technical audience, yet comprehensive of all relevant details to convey valuable insights.

Inspired by these principles, we reckon that the essence of data science ought to be traced back to its practical aspect, which is the starting sparkle and the ultimate destination of any data science investigation. Indeed, all data science



(a) time series



(b) geolocalization

Figure 1.1: **"Data Science" Google searches.** The trend of Google searches for the term "data science" (a) shows an evident increased popularity in latest years. The corresponding geolocalizations (b) testify the global penetration of the subject. The scale index is a re-scaled measure of the number of searches, where 100 corresponds to the maximum number of searches observed and 0 means that insufficient data were available.

projects share the common target of solving real-life problems in a data-driven fashion. Although overcoming hampers is not a novel challenge – in fact, one may see it as a key aspect of the evolution really, which has always been crucial for the progression of species, especially regarding the human kind – the distinctive feature of data science is its framing into a data-oriented paradigm. Not surprisingly, its explosion comes as a normal response to the increased availability of data we have been experiencing in modern times. Having data about virtually any aspect of life poses the question of how to leverage this information to tackle unsolved problems and re-think current approaches. This in turn urges for a methodological framework that provides tools for modeling natural phenomena and learning from their data. As a consequence, computing infrastructures are needed to manage the data and enable the practical implementation and use of the developed theories. Hence, the foundational core of data science can be established in the trinomial *data-modeling-computing* that summarizes the practical, theoretical and technical challenges related to the discipline. Importantly, the above chain of causes-consequences is not static, as advancements in any of the three areas pave the way for novel research in the others. Therefore, each of the components can be seen as both an enabling factor and a stimulus for the others, thus generating a virtuous circle that encourages progress.

Clearly, this process typically requires mixing heterogeneous competencies, from which the necessity for contamination between multiple disciplines contributing to sub-tasks propaedeutic to the final solution. Also, a clear visualization and communication of the results is paramount to guarantee effective usage of the developed methodologies and avoid misunderstandings in the various inter-domain interactions.

Guided by the former vision, this work discusses two independent data science projects developed in the scope of multidisciplinary applied research, and describes how the five pillars above are declined for the two use cases. Despite treating the proposed methodologies in detail, a particular emphasis posits on *i)* the interdisciplinary nature of the projects, *ii)* the origin of the data, the information they carry and the corresponding challenges, and *iii)* the practical impact of our approaches. Also, specific attention is devoted to data visualization, both in the initial exploration and the presentation of the results, for its crucial role in

enabling the communication among experts from various domains.

Structure of the Thesis

After an initial definition of the discipline of *Data Science*, this thesis is organized as follows.

Section 1.1 draws a historical reconstruction of the evolution of the data science concept over time, trying to clarify what this subject is all about and set an unambiguous reference framework. Section 1.2 then introduces the major learning paradigms and their characteristics.

Part I explores in greater detail the work presented in Morelli et al. (2021). In particular, Chapter 2 describes the technique of microscopic fluorescence and its application to life science and biology experiments. The task of counting objects in digital images is then presented, and some relevant literature is reviewed. Chapter 3 describes the **Fluorescent Neuronal Cells** dataset (Clissa et al., 2021), focusing on data acquisition, data annotations, and peculiar characteristics and challenges. In Chapter 4, the **cell ResUnet (c-ResUnet)** (Morelli et al., 2021) model is introduced and compared with several alternative architectures. Also, three experimental settings are detailed, which are then used for testing competing architectures through ablation studies. In Chapter 5, the performances achieved by the proposed approaches are evaluated both quantitatively and qualitatively. Finally, Chapter 6 summarizes the main findings of the study and discusses possible extensions.

Part II deals with the issue of computing infrastructure management in the context of the Worldwide Large Hadron Collider Computing Grid. Chapter 7 introduces the High-Energy Physics community and the related experiments conducted through the Large Hadron Collider (LHC) at the CERN laboratory, with a particular focus on the computing infrastructure adopted for sharing and analyzing the experimental data. In particular, Chapter 8 describes more in detail the data flows generated by the LHC and introduces the Operational Intelligence project (Di Girolamo, Alessandro et al., 2020), a joint effort of different collaborations to automate infrastructure management. Among its various activities, the subject of troubleshooting transfer failures is taken into account, and some ongoing attempts

are outlined alongside our contributions. Chapter 9 discusses a possible approach to the problem aimed at supporting current operations by automatically extracting a few error categories (in the order of tens) from the massive amount of daily failed transfers (in the order of millions). Chapter 10 reports a demonstration of our approach applied to one day of data as a proof of concept for its potential, taking into account both a qualitative interpretation of the results and a quantitative proxy of the performance. Finally, Chapter 11 summarizes the advantages and limitations of the proposed methodology and suggests a possible direction for future developments.

1.1 History of Data Science

Some authors attribute the first appearance of the term "data science" in literature to Naur (1974), where he presents data science as an alternative name for computer science and defines it as "*the science of dealing with data, once they have been established, while the relation of the data to what they represent is delegated to other fields and sciences*".

However, the first mention of something resembling what we currently perceive as data science dates back to Tukey (1962), where the term *data analysis* is used to indicate a discipline with the connotations of *science*, which is "*defined by a ubiquitous problem rather than by a concrete subject*". Tukey's description incorporates many aspects seemingly tied closely to applied statistics:

> *procedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analyzing data.*

Nevertheless, the extent Tukey attributes to data analysis is broader than its philological meaning, as it comprises all of statistics and embeds it in a larger entity (Huber, 2012; Donoho, 2017). Indeed, Tukey himself sets the boundaries between data analysis and statistics in their respective binding to the strict formalism of mathematics. In fact, he appeals for a looser attachment to mathematical rigor and suggests focusing on actionable insights rather than theory ("*scope and usefulness*

over security"). In Tukey's vision, data analysis is framed within the paradigm of an empirical science proceeding by trial and error, whose role is to advise actions and insights for new investigations. More precisely, this has to hold even when not enough evidence is available to be (mathematically) sure of the conclusions, thus accepting a reasonable degree of error ("*Data analysis must be willing to err moderately often in order that inadequate evidence shall more often suggest the right answer*"). As a result, the mathematical foundations should be used as an element to directing the strategy of investigation rather than validating its results ("*as bases for judgment rather than as bases for proof*"). Finally, Tukey identifies four driving forces in the emerging data analysis science:

> *Four major influences act on data analysis today:*
>
> 1. *The formal theories of statistics*
> 2. *Accelerating developments in computers and display devices*
> 3. *The challenge, in many fields, of more and ever larger bodies of data*
> 4. *The emphasis on quantification in an ever wider variety of disciplines*

Despite being released 60 years ago, Tukey's description is astonishingly modern and well depicts many activities under the umbrella of what we refer to as data science today. In line with Tukey's vision, several authors have defined – more or less explicitly – data science as an extension of applied statistics.

In 1985, Wu used the term "Data Science" for the first time as an alternative name for statistics, and in 1997 advocated the renaming "*Statistics = Data Science*" to avoid the limiting association of statistics with accounting and data description. In particular, he indicates *large/complex data*, *data-driven methodologies* and *representation and exploitation of knowledge* as promising future directions for statistics. Also, he suggests that statistics degree programs should be grounded on the "*Statistical Trilogy: Data Collection – Modeling and Analysis – Problem Understanding/Solving and Decision Making*", with a strong emphasis on a balanced, interdisciplinary curriculum (Wu, 1997).

In 1993, Chambers provocatively distinguished between "*Greater or Lesser Statistics*", drawing the line between the *traditional data analysis supported by mathematical statistics* and a broader concept of *learning from data* (Chambers,

1993). In his view, data science reshapes statistics by enlarging its focus in the areas of data preparation and visualization. In particular, Chambers points to the opportunities offered by new types of data and new types of presentation, deeming such extended subject to be even larger than Tukey's data analysis.

In 2001, Breiman shed new light on the debate about the nature of data science. Namely, he distinguishes between two cultures of statistical modeling depending on the ultimate goal of the analysis of data: *information* and *prediction* (Breiman, 2001). The former is related to the traditional statistics approach, where we assume that natural processes can be described by mathematical formulations, i.e. the *models*. Thus, by fitting them to the data we can infer knowledge about the model's parameters and understand the dynamics of the phenomenon. The latter, instead, is what we intend as data science. This prioritizes prediction over inference without caring much about the underlying mechanism that generates the data.

In the same year, Cleveland presented an "*action plan to expand the technical areas of statistics focuses on the data analyst*" (Cleveland, 2001). In particular, he posits a great deal of attention to the practical impact of the data analysis procedures, suggesting that results in data science "*should be judged by the extent to which they enable the analyst to learn from data*" (rather than their theoretical properties). Also, he outlines a precise structure for future academic programs, where it stands out the equal balance given to *multidisciplinarity investigations (25%), model and methods for data (20%), computing for data (15%) and theory (20%)*. Notably, he also leaves some space devoted explicitly to *pedagogy (15%)*.

Donoho (2017) presents a thorough historical review of the evolution of data science as a discipline. Specifically, he describes data science as an applied field growing out of traditional statistics, and he names it as "Greater Data Science" (GDS) in analogy with the Greater Statistics nomenclature adopted by Chambers. In practice, Donoho proposes a taxonomy for the activities of the would-be such discipline of the future, based on the *merging, relabeling and generalization* of the divisions proposed by Chambers and Cleveland:

> *The activities of GDS are classified into six divisions:*
>
> 1. *Data Gathering, Preparation, and Exploration*

2. *Data Representation and Transformation*
3. *Computing with Data*
4. *Data Modeling*
5. *Data Visualization and Presentation*
6. *Science about Data Science*

Interestingly, Donoho also includes the study of data science (GDS6) as a part of the discipline itself, and he thereby encompasses activities devoted to:

> *identify commonly occurring analysis/processing workflows, for example, using data about their frequency of occurrence in some scholarly or business domain; when they* ***measure the effectiveness of standard workflows in terms of the human time, the computing resource, the analysis validity, or other performance metrics****, and when they uncover emergent phenomena in data analysis, for example, new patterns arising in data analysis workflows, or disturbing artifacts in published analysis results.*

More recently, Z. Zhang and Zhang (2021) inspected a clever approach to define the discipline of data science that fits exactly – perhaps unwittingly – in Donoho's GDS6. In fact, they adopt a data-driven strategy by analyzing the course descriptions of graduate degree programs in analytics, business analytics, and data science offered by American universities.

Nonetheless, researchers and practitioners are yet to reach a complete agreement on its definition. Indeed, other schools of thought, especially in industry, suggest that data science should be considered as a separate body of knowledge due to its focus on digital and qualitative data (Silver, 2013; Priceonomics, 2015; Dhar, 2013), and that statistics is only a marginal, nonessential part of it (Gelman, 2013). The interested reader is referred to Donoho (2017) and L. Cao (2017) for a more extensive dissertation on the topic.

This thesis shares the vision of data science as an extension of applied statistics. In particular, we frame it as an applied science that studies how to combine domain expertise with suitable learning approaches and necessary computing resources to process real data for a tangible purpose. Also, we devote particular care to the visualization of data. Furthermore, we try to embrace GDS6 when assessing

the results to provide a comprehensive evaluation of our approaches from the application point of view.

1.2 Learning theory

By and large, data science can be summarized as the process of learning from data. Multiple strategies are available to achieve that, differing for their fundamental assumptions, adopted methodologies, and ambit of applicability. Although an exhaustive list of the possible approaches would be much longer, the different methods can be categorized into three main learning paradigms: supervised, unsupervised, and reinforcement learning.

The **Supervised Learning** (SL) paradigm is based on the concept of learning by examples, sometimes also referred to as "learning with a teacher" (Friedman, Hastie, Tibshirani, et al., 2009, Chapter 14). Formally[1], this can be expressed as the task of predicting a (set of) response/target[2] variable Y based on a set of predictors/inputs[2] $\boldsymbol{X}$. In other words, the objective is to learn a mapping function, f, between the predictors and the response such that it is possible to forecast the value of Y given the values of $\boldsymbol{X}$ up to some random noise ϵ:

$$Y = f\left(\boldsymbol{X}; \boldsymbol{\theta}\right) + \epsilon \tag{1.1}$$

where $\boldsymbol{\theta}$ is a parameter vector that defines the precise form of the function f. In practice, the model (*student*) starts by guessing the association between some input data $\boldsymbol{x}$ and the corresponding response based on an initial configuration of the parameters. Then, the corresponding *label*, y, is used to score the quality of the produced association according to a given performance measure (*loss function*). This comparison between the predicted answer and the right label is what furnishes the supervision, thus allowing the parameter updates to reduce the ob-

[1] the variables in this section are represented according to the statistical conventions, namely: random variables are capitalized, while observations are expressed as the corresponding lowercase letters; also, bold symbols indicate vectors, meanwhile normal font stands for univariate quantities

[2] the terms *predictors* and *response* proper of the statistical community are used hereafter interchangeably with *inputs* and *output*, respectively, that are instead more used in the machine learning jargon

served mismatch between the prediction and label. Finally, the whole procedure is iterated until the learned parameters generate a sufficiently satisfying mapping that allows the generalization to new data. 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 target. 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, 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.

Contrarily, **Unsupervised Learning** (UL), or "learning without a teacher", addresses the task of learning when no labels are available. In this case, the goal is to directly infer properties of the input data $\boldsymbol{X}$ without the help of a supervisor (Friedman et al., 2009, Chapter 14), i.e. identifying hidden structures and commonalities in the data. Cluster analysis is one of the most popular families of unsupervised learning algorithms. In this case, the problem is formulated as finding subgroups of observations (clusters) that can be considered similar according to a given measure of distance/similarity. The training phase proceeds by assigning the data points to the clusters by minimizing an overall measure of internal compactness and/or separation between different groups. Thus, UL is very convenient in many practical situations since it does not require labels (e.g. market basket analysis, anomaly detection, pattern recognition, dimension reduction), and it can be pursued as a goal per se – i.e. to discover hidden patterns in data – or as a preprocessing for subsequent elaborations – e.g. learn a convenient representation for further analysis. However, the absence of a reference target makes it difficult – usually impossible – to objectively measure the goodness of the results – as opposed to the intuitive notion of score given by the loss function in SL (Von Luxburg, Williamson, & Guyon, 2012). Rather, the evaluation typically resorts to heuristic arguments based on the interpretability of the results for a given use case, which makes the whole process somewhat arbitrary.

Alternative learning paradigms have been proposed at the intersection between supervised and unsupervised learning to access the benefits of both strategies and

mitigate their limitations. For instance, *semi-supervised learning* adopts a mixed training strategy where only a small fraction of the data are labeled. Nonetheless, such approaches perform significantly better than purely unsupervised methods. Another example is represented by *weakly supervised learning*. In this case, the collected labels may be scarce in number and/or quality (i.e. inaccurate). Again, this implies considerable improvements in performance at lower annotation costs with respect to supervised strategies. An emerging alternative that is showing great potential is *self-supervised learning*. This approach tries to leverage large amounts of unlabeled data to learn representations to reuse for different purposes. In particular, a model is first pre-trained on a so-called pretext task, i.e. a learning objective other than the desired target but propaedeutic for its learning, for which labels can be automatically/easily retrieved. Then, the learned representation is reused to fine-tune the model on the downstream task of interest.

A completely different approach is represented by **Reinforcement Learning** (RL). In its simplest formulation, the learning task is formalized in terms of an agent interacting with an evolving environment, and the goal is to learn a policy – i.e. a mapping between the states of the environment and the corresponding actions the agent may undertake in those states. These actions generate a reward for the agent, and an optimal mapping must be sought to maximize this notion of reward over time. Drawing a comparison with the above paradigms, in this case the inputs correspond to the environment settings at a given time t, and the agent has to output an action a_t. Like in UL, no explicit labels are provided to the agent to learn from. Instead, the reward signal is used as an indirect measure of ranking between the goodness of different actions for a given state. This is conceptually different from the label in SL, which instead represents the right prediction for a given input (Sutton & Barto, 2018, Chapter 1). The RL framework is particularly appealing for its resemblance with the way we humans learn. Indeed, we interact with an environment without an explicit supervision, and we learn to associate actions to given situations. For this reason, RL is also very adopted in practice for an wide range of applications ranging from autonomous driving to playing games.

In this thesis, we adopt different learning paradigms based on the nature of the data and the corresponding use cases. In particular, Part I exploits convolutional neural networks for classifying pixels into signal and background classes. Part II

explores the domains of self-supervised and unsupervised learning instead. Specifically, a language model is first used to get a convenient numeric representation for textual data (Section 9.1.2), and a K-means algorithm is then applied to get clusters of similar error messages (Section 9.1.3).

Part I

Fluorescent Neuronal Cells dataset and c-ResUnet

Chapter 2

Introduction

One of the reasons for the increasing popularity of data science is the extensive list of successes achieved thanks to statistical approaches capable of learning from data. Deep Learning is one of such techniques and comprehends a family of models derived from classical Artificial Neural Networks. Their distinctive characteristic is the exploitation of efficient implementations available in modern computers to build architectures with many hidden layers, whence the 'deep' connotation. Among these models, Convolutional Neural Networks (CNNs) (Jimenez-del Toro et al., 2017; Greenspan, van Ginneken, & Summers, 2016) were the first apparent evidence of Deep Learning's potential, showing the ability to outperform the state-of-the-art in many computer vision applications in the past decade. Successful examples range from classification and detection of basically any kind of objects (Krizhevsky, Sutskever, & Hinton, 2012; Redmon, Divvala, Girshick, & Farhadi, 2016) to generative models for image reconstruction (Cheng, Chen, Alley, Pauly, & Vasanawala, 2018) and super-resolution (Ledig et al., 2017). Thus, researchers from both academy and industry have started to explore adopting these techniques in fields such as medical imaging and bioinformatics, where the potential impact is vast. For instance, CNNs have been employed for the identification and localization of tumors (Havaei et al., 2017; Vandenberghe et al., 2017; Ciresan, Giusti, Gambardella, & Schmidhuber, 2012, 2013), as well as detection of other structures like lung nodules (Jiang, Ma, Qian, Gao, & Li, 2018; Meraj et al., 2020; Su, Li, & Chen, 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 & Lundervold, 2019; Sahiner et al., 1996; Yadav & Jadhav, 2019).

In the wake of this line of applied research, Part I tackles the problem of counting cells into fluorescent microscopy pictures. In particular, our work aims at developing a Deep Learning approach for automatic recognition and counting of neuronal cells. We start by exploiting formerly available fluorescence pictures acquired during past experiments, and we collect detailed pixel-wise segmentation maps tracking the location of the cells in such pictures. Then, we attempt different approaches based on convolutional neural networks trained from scratch in a supervised manner. In the end, the results are assessed both in terms of cell detection and counting performances. Also, a thorough qualitative assessment is conducted with the help of domain experts. Importantly, the collected dataset and the best pre-trained model are released to foster future research in this and related areas.

The following sections describe the most relevant stages of our project. In particular, Chapter 2 presents a panoramic of fluorescence microscopy, its application to natural sciences and the more general task of counting objects in images, as well as our contributions to such domains. Chapter 3 then introduces the **Fluorescent Neuronal Cells** dataset with its characteristics and challenges. Chapter 4 discusses the alternative techniques we adopted to tackle the problem and the specifics of our experimental settings, with a particular focus on the **cell ResUnet** architecture of our proposal and the weight map adopted for promoting precise segmentation. In Chapter 5, the output of our experiments is reported. Particular attention is devoted to comparing alternative approaches and evaluating various study design choices. Finally, Chapter 6 summarizes the main findings of our work and outlines some possible future lines of research related to this application.

2.1 Fluorescence microscopy

Fluorescence is a luminescence phenomenon that was first discovered in 1852 by George G. Stokes (Stokes, 2010). He observed that some molecules, denominated fluorophores, are susceptible to emitting light when they are in electronically ex-

cited states. These states can be caused by a physical mechanism (e.g. absorption of light), a mechanical process (e.g. friction) or chemical interactions. In other words, fluorescence is the property of some atoms and molecules to absorb light at a specific wavelength. In turn, this causes a transition from a ground state to an excited one. When that happens, the fluorophore becomes unstable and releases the absorbed energy by emitting light of a longer wavelength (Stokes shift) to get back to the ground state. This difference in wavelengths between the absorbed and emitted light is the enabling factor of microscopic fluorescence. In practice, synthetic fluorophores having desired fluorescence properties are adopted, and the instrumentation is carefully set up to illuminate the specimen with a precise wavelength. The Stokes shift is then exploited to filter out the exciting light without blocking the emitted fluorescence, thus making the fluorescent objects visible (Lichtman & Conchello, 2005).

Many experiments in the life science domain are based on this technique. Specifically, the fluorophore is designed to couple with the molecular structures of interest and interact with the tissues under study. In this way, the efficacy of a treatment or the organism response to a given environment is assessed by tracking the activity/presence of the targeted compounds. This process often resorts to counting how many molecular structures produced fluorescent emissions in the different conditions (Hitrec et al., 2019, 2021; da Conceição, Morrison, Cano, Chiavetta, & Tupone, 2020). For example, Hitrec et al. (2019) investigated the brain areas of mice that mediate the entrance into torpor, showing evidence of which networks of neurons are associated with this process.

Torpor, also referred to as dormancy, is a behavioral and physiological state often observed in both animals and plants. In particular for animals, this condition is typically characterized by reduced body temperature and depressed metabolism, and it is exploited by living organisms in response to a variety of hostile environmental stimuli, including low temperature, water or food deprivation (Ganslo\ßer & Jann, 2019; Withers & Cooper, 2019). Interestingly, some studies have shown how this condition can induce resistance to radiations (Cerri et al., 2016; Tinganelli et al., 2019; Cerri, Negrini, & Zoccoli, 2021), which could be crucial for a broad spectrum of medical purposes. Certainly, knowing the mechanisms that rule the onset of lethargy, and understanding how to trigger their activation, may have

a significant impact when coming to human applications. For instance, such an approach could be very beneficial when dealing with patients who need invasive surgery, e.g. intensive care or oncology treatments (Bouma et al., 2012; Alam et al., 2012; Bellamy et al., 1996). Pushing the imagination even further, one could think of hibernation as an enabling factor for long interplanetary trips, where astronauts could overcome or limit side effects of space travels (Cerri et al., 2016; Cerri, Hitrec, Luppi, & Amici, 2021; Puspitasari et al., 2021; Bradford, Schaffer, & Talk, 2020).

As a result of all these implications, it becomes evident how the matter assumes considerable interest and qualifies for further in-depth studies. Nevertheless, the technical complexity and the manual burden of these analyses often hampers fast developments in the field. Indeed, these experiments typically rely heavily on semi-automatic techniques that involve multiple steps to acquire and process images correctly. Manual operations like area selection, white balance, calibration and color correction are fundamental in order to identify neurons of interest successfully (Dentico et al., 2009; Gillis et al., 2016; Luppi et al., 2019). As a consequence, this process may be very time-consuming depending on the number of available images. Also, the task becomes tedious when the objects appear in large quantities, thus leading to errors due to fatigue of the operators. Finally, a further challenge is that sometimes structures of interest and image background may look similar, making them hardly distinguishable. When that is the case, counts become arguable and subjective due to the interpretation of such borderline cases, thus leading to an intrinsic arbitrariness.

For these reasons, this work aims at facilitating and speeding up future research in this and similar fields through the adoption of a CNN that counts the objects of interest without human intervention. The advantages of doing so are two-fold. On one side, the benefit in terms of time and human effort saved through the automation of the task is evident. On the other, using a Deep Learning model would impede fatigue errors and introduce a systematic "operator effect". In this way, the annotation would result in a more coherent process and it would guarantee similar structures are labeled consistently, both within the same experiment and across different studies.

2.2 Counting objects in images

Counting objects in digital images is a common task for many real-world applications (Segui, Pujol, & Vitria, 2015; Arteta, Lempitsky, & Zisserman, 2016; Cohen, Boucher, Glastonbury, Lo, & Bengio, 2017; Rahnemoonfar & Sheppard, 2017) and different approaches have been explored to automate it (Lempitsky & Zisserman, 2010; Ciresan et al., 2012, 2013; Kraus, Ba, & Frey, 2016; Raza et al., 2017).

Multiple paradigms for counting objects in images have been proposed depending on the study's specific needs and the available data. The natural setting to tackle this problem is the so-called *counting-by-regression* scheme. In this case, the input data consist of the image and, optionally, other features. The model is then trained to output the raw count of objects directly. However, this approach does not provide any immediate justification of which elements generated the final count. Possible refinements are also available, based on regressing density maps instead of counts directly (W. Xie, Noble, & Zisserman, 2018; Cohen et al., 2017). In this case, the predicted density maps provide some hints of where the objects are generally located, but without identifying individual instances. Another strategy is *counting-by-detection*. In this case, the model is trained to reproduce ground-truth masks having bounding boxes surrounding the objects to detect. In this way, the output becomes an image where pixels are classified either into the signal class (within the boxes) or as background (outside). This outcome provides the raw count as the number of sets of connected pixels, plus a justification in terms of the localization furnished by the bounding boxes. Building on the latter framework, one can refine the model's ability to detect and localize the objects by including semantic labels for each pixel in the ground-truth masks. This allows pixel-wise classification that enables to discern the exact boundaries of each object. The total count is then retrieved again by looking at groups of connected pixels. Such an approach is referred to as *counting-by-segmentation*. This work is framed under the latter paradigm so to support the results with a clear, visual evidence of which objects contribute to the final counts.

2.2.1 Related works

Some interesting approaches have been proposed for detecting and counting cells in microscopic images. Faustino, Gattass, Rehen, and de Lucena (2009) propose an automated method leveraging the luminance information to generate a graph representation from which counts of cells are retrieved after a careful mining process. Nonetheless, their approach relies on the manual setting of some parameters, like the optimal threshold for separating cell clusters and the luminance histogram binning adopted for retrieving connected components, which hampers the extension to different data.

Ronneberger et al. (2015) present a Deep Learning approach for segmentation of cells in an image. Their main contribution is the introduction of a novel network architecture, *U-Net*, which is still state-of-the-art in several applications with only slight adaptations (Masin et al., 2021; Ritch et al., 2020). The basic idea is to have an initial contracting branch used to capture relevant features, and a symmetric expanding one that allows for accurate localization. The main drawback is that its enormous number of parameters requires relevant computing power and makes the training difficult because of vanishing gradient (Hochreiter, 1998; Guan, Khan, Sikdar, & Chitnis, 2020; Y. Cao, Liu, Peng, & Li, 2020; Qamar, Jin, Zheng, Ahmad, & Usama, 2020). For this reason, a commonly used variation adopts residual units (K. He, Zhang, Ren, & Sun, 2016) with short-range skip-connections and batch normalization to prevent that problem. Also, this typically guarantees comparable performance with much less parameters.

Kraus et al. (2016) combine deep CNNs with multiple instance learning in order to classify and segment microscopy images using only whole image level annotations. Raza et al. (2017) propose a novel multiple-input multiple-output convolution neural network (MIMO-Net) that utilizes multiple resolutions of the input image, connects the intermediate layers for better localization and context and generates the output using multi-resolution deconvolution filters.

A common downside of these approaches is the need of ground-truth labels (or masks) with accurate annotations of whether each pixel belongs to an object – in this case a cell – or the background, resulting in an additional and laborious data preparation phase. In an attempt to overcome this limitation, some works

try to tackle the problem in an unsupervised fashion. For example, Riccio, Brancati, Frucci, and Gragnaniello (2018) address segmentation and counting with a step-wise procedure. The whole image is first split into square patches, and a combination of gray level clustering followed by adaptive thresholding is adopted for foreground/background separation. Individual cells are then labeled by detecting their centers and applying a region growing process. While this procedure bypasses the need for ground-truth masks, it still requires handcrafted hyperparameters selection that needs to be tuned for new data. For additional examples of segmentation in biological images, the interested reader is referred to Riccio et al. (2018).

2.3 Contribution

Part I tackles the issue of automating cell counting in fluorescence microscopy using Deep Learning. Building upon Morelli et al. (2021), the following focuses on a supervised learning approach in the context of semantic segmentation. The main contributions of this work are the following.

First, we propose an automatic approach for counting neuronal cells based on Deep Learning. To achieve that, two families of network architectures are compared – Unet and its variation *ResUnet* – in terms of counting and segmentation performance. In particular, we introduce a slight modification of the ResUnet explicitly tailored for our use case, which we call **cell ResUnet** (**c-ResUnet**).

Second, an error weighting mechanism is proposed to penalize misclassifications on cell boundaries and its effectiveness is demonstrated through ablation studies, showing how this strategy promotes accurate segmentation, especially in cluttered areas.

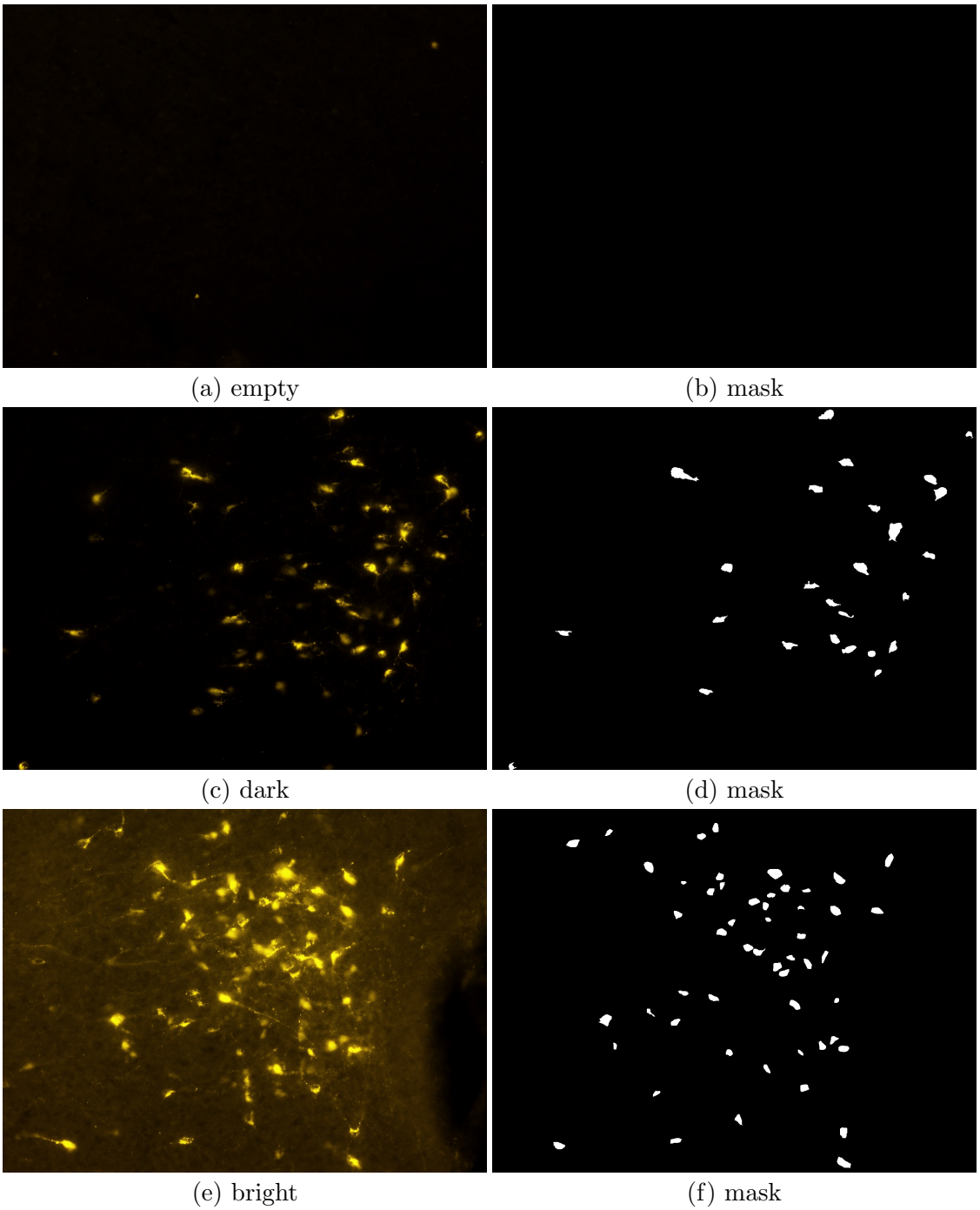
Last but not least, our pre-trained model[1] and the rich dataset with the corresponding ground-truth annotations (Clissa et al., 2021) are released to foster methodological research in both biological imaging and deep learning communities.

[1] available at: `https://l.infn.it/linkmodel`

Chapter 3

Fluorescent neuronal cells dataset

The **Fluorescent Neuronal Cells** dataset (Clissa et al., 2021) consists of 283 pictures of mice brain slices and the corresponding ground-truth labels. In order to acquire these images, the mice are first subjected to controlled experimental conditions. Then, a monosynaptic retrograde tracer (b-subunit of Cholera Toxin, CTb) is surgically injected into a small region of the brain called raphe pallidus. Subsequently, some areas of interest – i.e. dorsomedial hypothalamic nucleus (DM), lateral hypothalamic area (LH) and ventrolateral part of the periaqueductal gray matter (VLPAG) – are observed to spot signals of the tracer indicating the neurons that projects into the injection site (Hitrec et al., 2019), thus highlighting functional connections between brain regions. These tracers are widely used to understand neuronal links among neural structures since their composition allows synaptic termini to capture the tracer and then retrogradely transport it into the cell soma through the axon. As the term "monosynaptic" suggests, they cannot pass into other cells once captured by the synaptic terminal, thus enabling a unique association between the marked neurons and the area where the tracer was injected. After some time required for capturing and transporting the tracer – 7 days in our case, as typical for laboratory rodents –, the brain is cut into slices and observed through a fluorescence microscope. This tool allows to lit the specimens with light at a specific wavelength and select the corresponding narrow frequency emitted by a fluorophore associated with the tracer. In our case, the samples are observed at a 200x zoom through a Nikon eclipse 80i microscope equipped with a



(a) empty (b) mask

(c) dark (d) mask

(e) bright (f) mask

Figure 3.1: **Sample data.** Raw images and corresponding ground-truth masks.

Nikon Digital Sight DS-Vi1 color camera (3.1876 pixels/micron of resolution). The configuration adopted exploits absorption (excitation) and emission wavelengths of 555 and 580 nanometers, respectively, corresponding to a hue between yellow and orange. Thus, the resulting images depict neurons of interest as yellow-ish spots over a composite, generally darker background (Figs. 3.1a, 3.1c and 3.1e).

3.1 Ground-truth labels

Under a supervised learning framework, the training phase leverages ground-truth labels acting as examples of desired outputs that the model should learn to reproduce. In the case of image segmentation, such targets are in the form of binary images (*masks*) where the objects to segment and the background are represented by white and black pixels, respectively (Figs. 3.1b, 3.1d and 3.1f).

Obtaining target masks usually requires a great effort in terms of time and human resources, so an initial automatic procedure was exploited to speed up the labeling process. In particular, starting from a large subset composed by 252 pictures, gaussian blurring (with size $\sigma = 7$)[1] was first applied to mitigate small-frequency noise. Then, the resulting images were subjected to a thresholding operation. For this step, the image histogram of the pixel intensity was considered, and a cutoff equal to the 97-*th* percentile of the intensity distribution was adopted for binarization. The goal was to obtain a loose selection of good candidates to be labeled as neuronal cells. After that, knowledgeable operators reviewed the results to discard the false positives introduced with the previous procedure, taking care of excluding irrelevant artifacts and misleading biological structures. The remaining 31 images were segmented manually by domain experts. Significant pictures with challenging traits – such as artifacts, filaments and cell agglomerates (see Section 3.3) – were included in the latter set to have highly reliable masks for the most arduous examples[2].

Despite the huge popularity Deep Learning has gained in computer vision in the last decade, the lack of annotated data is a common curse when dealing with applications involving non-standard images and/or tasks (J. Xie, Kiefel, Sun, &

[1] the `skimage.filters.gaussian` utils was adopted for this task

[2] check the *README* file in Clissa et al. (2021) for the list of manually segmented images

Geiger, 2016). Since ground-truth labels are expensive to acquire in terms of time and costs (Vijayanarasimhan & Grauman, 2009; Mullen Jr, Tanner, & Sallee, 2019), a common approach is to fine-tune models that are pre-trained on giants datasets of natural images like ImageNet (Deng et al., 2009) or COCO (T.-Y. Lin et al., 2015), possibly using as few new labels as possible for the task of interest. However, this strategy often does not apply to use cases where the pictures under analysis belong to extraneous domains with respect to the ones used for pre-training (Alzubaidi et al., 2021). For this reason, by releasing the annotated dataset[3] and the pre-trained model[4] we hope to *i)* foster advances in fields like biomedical imaging through the speed up guaranteed by the automation of manual operations, and *ii)* promote methodological research on new techniques of data analysis for microscopic fluorescence and similar domains.

3.2 Data exploration

Fluorescent Neuronal Cells images are high-resolution RGB pictures of constant shape (1200 pixels height by 1600 pixels width) collected under fixed experimental conditions. The data can be explored at two complementary levels: pixel features and cell characteristics. On the one hand, interesting insights can be retrieved by looking at pixels' color and luminance information. Also, analogous analyses on the ground-truth masks reveal essential information about class-imbalance between signal and background. On the other hand, examining object properties can highlight potential nuisances and suggest how to evaluate model performances.

The above data explorations are presented in the following sections of this chapter, and a summary table of the most important data features is reported in Table 3.1.

3.2.1 Salient features

The picture appearance is dominated by two prevalent tints due to the intentional selection of a specific wavelength: a darker hue corresponding to areas whose light

[3]available at: `http://amsacta.unibo.it/6706/`
[4]available at: `https://l.infn.it/linkmodel`

	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

Table 3.1: **Distributions summary.** Summary of the distributions illustrated in Section 3.2. For each distribution we report 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.

was filtered out and a yellow tone emitted by the fluorophore (Figs. 3.1a, 3.1c, 3.1e and 3.7). As a consequence, the only color channels to be populated are red and green, while blue is typically empty. An example of this effect is reported in Fig. 3.2, where the average distribution of pixel intensity is illustrated[5]. In practice, the blues have an extremely narrow distribution squashed on zero, which makes it even difficult to visualize. The red and green channels are instead more populated. Their central tendency is still concentrated on low values due to the prevalence of background pixels, however we observe longer and thicker right tails, especially for the red channel (see *red*, *green* and *blue* columns in Table 3.1 for a numeric summary). Guided by this observation, one may argue that all this information is superfluous, so resorting to a grayscale transformation could be better since the images are ultimately shades of yellow. A nice way to visually investigate such relationships is by exploring the colorspace representations of several images. Figure 3.3 reports the RGB and HSV encodings for two randomly sampled images. Indeed, the RGB representations (Figs. 3.3a and 3.3c) corroborate the previous intuition, as most pixels lay almost on a straight line in the red-green plane. This suggests that the two channels are highly correlated, so a one-dimensional

[5]the box captures the central half of the distribution (25-*th*/75-*th* percentile); the solid and dashed lines represent the median and the mean, respectively. The same convention is adopted in the following violin plots

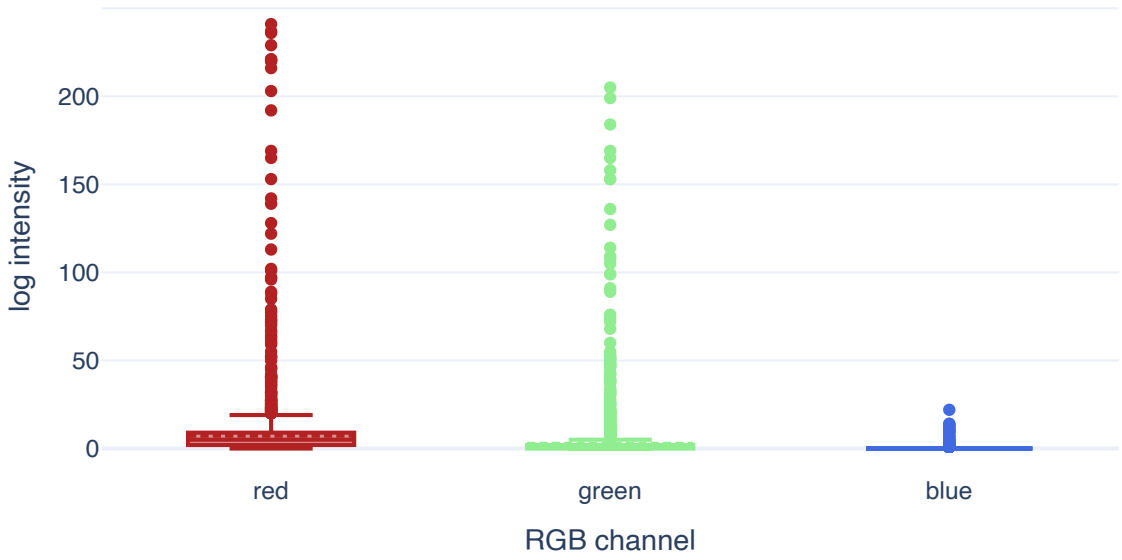


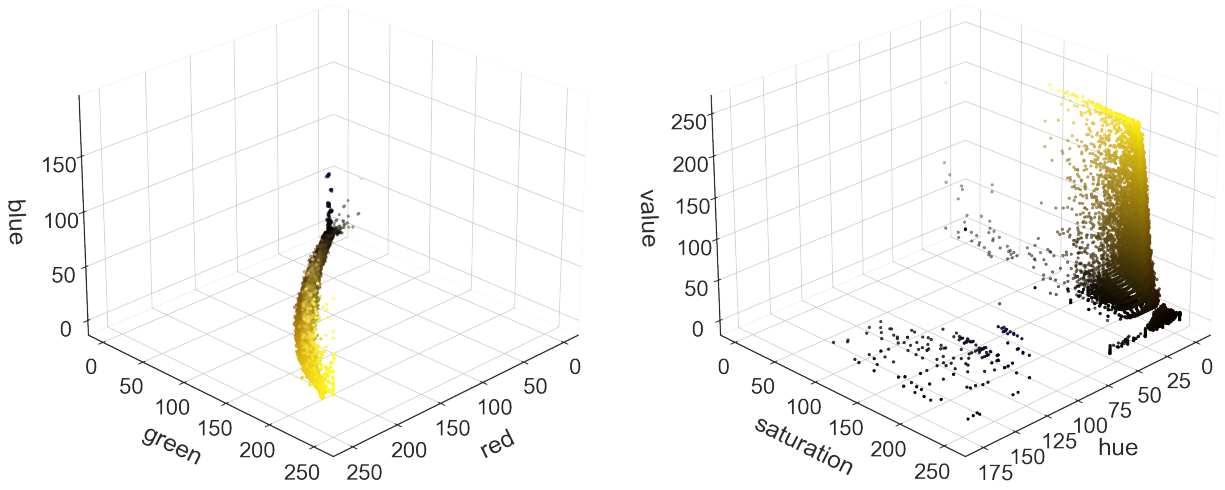
Figure 3.2: **Pixel intensity distribution.** Boxplot of the average distribution of pixel intensities across the RGB channels.

subspace may be enough to represent most of the variability of the data. In turn, this would bring two advantages: ease the learning process – as neural networks typically suffer when inputs are correlated – and make it more efficient – as only one channel is considered instead of three.

However, the use-case at hand has no stringent requirements in terms of computing resources and runtime, so the 3-channels training is still feasible. More importantly, the information thrown away when converting to grayscale, although tiny, may be crucial to discriminate background and signal. Hence, a 3D-encoding may still be worthed but the RGB colorspace may not be the optimal representation to learn this separation. A hint of that is demonstrated in Figs. 3.3b and 3.3d, where the same images are depicted according to the HSV encoding. In this case, the separation between dark and colored tones appears more evident. Moreover, most of the pixels are concentrated in low hue values and their distribution seems more spread across the saturation-value plane.

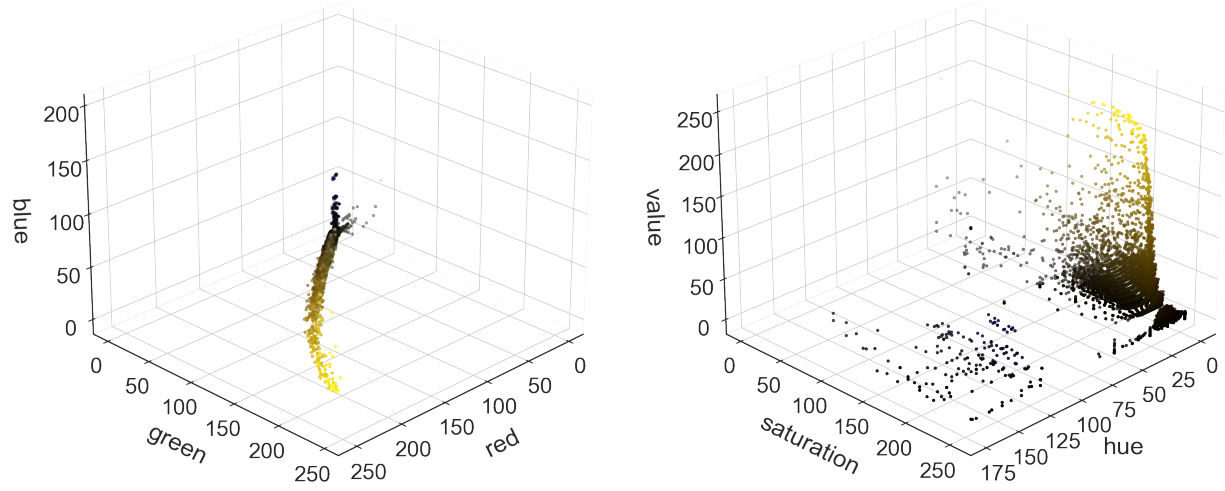
All that being considered, we try to leverage the insights of both approaches. On one side, the RGB colorspace is taken as a starting point to retain all available information. On the other, the model first layer is designed to incorporate a colorspace transformation from RGB to a single channel. In this way, we avoid

Mar19bS1C4R3_LHl_200x_y.png



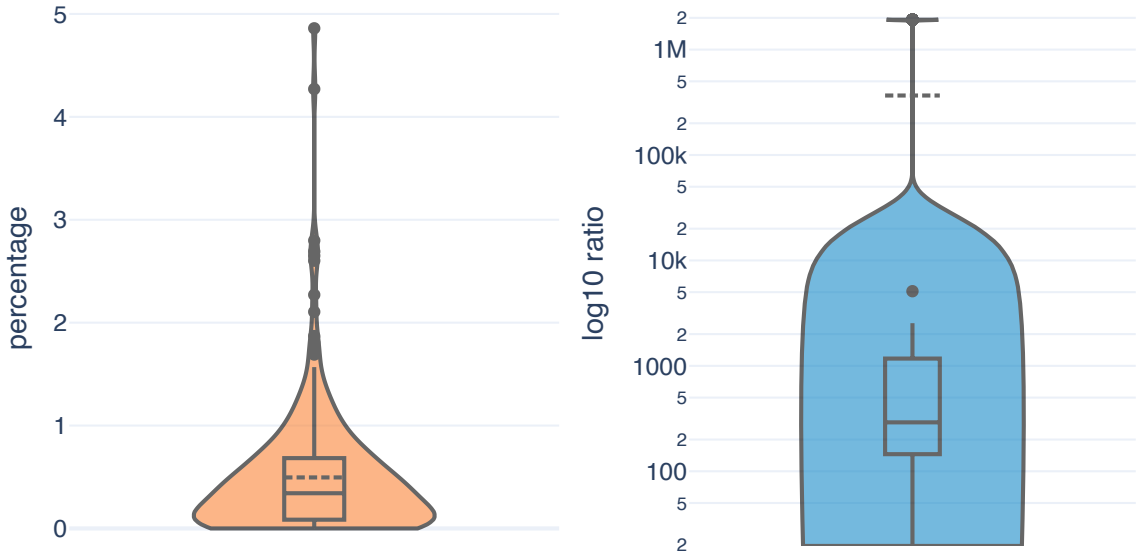
(a) RGB (b) HSV

Mar21bS1C1R3_VLPAGl_200x_y.png



(c) RGB (d) HSV

Figure 3.3: **Colorspace.** Two images represented as 3D points according to their RGB (left) and HSV (right) encodings. Each point is colored as the corresponding pixel in the original image.



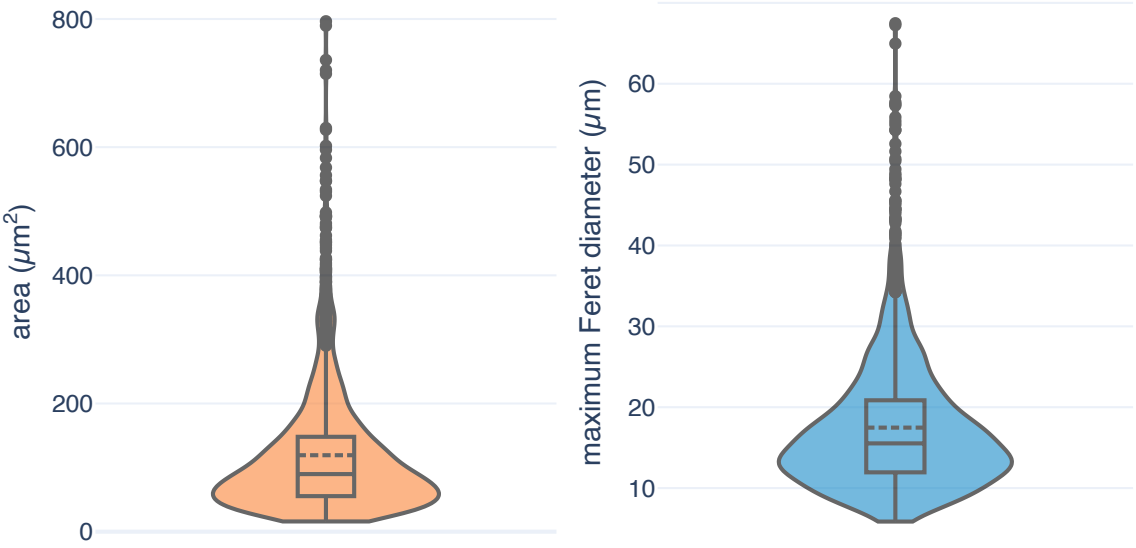
(a) Signal % over total pixels (b) Background to signal ratio

Figure 3.4: **Class imbalance.** Violin plot and boxplot of signal percentage (a) and background to signal ration (b).

introducing any colorspace-related bias since the model learns the most convenient representation. At the same time, we exploit the observation that a one-dimensional manifold is probably enough to express the data variability by forcing the learned encoding to one channel.

3.2.2 Class imbalance

Inspecting ground-truth masks at pixel level reveals important characteristics that affect the training process. By looking at the cardinality of pixels belonging to the background and the signal it is possible to notice how the two classes are extremely unbalanced (see *signal (%)*, and *signal ratio* columns in Table 3.1 for a numeric summary). Figure 3.4a shows a violin plot of the percentage of signal pixel over the total image pixels across the 283 pictures. The distribution is deeply skewed towards 0, with a median of 0.34% and a 90-*th* percentile of 1.07%. Hence, almost 90% of the images contain less than 1% of pixels belonging to the signal. Even more significantly, the right tail does not exceed 5% of signal coverage, with a maximum of 4.86%.



(a) Area (b) Feret diameter

Figure 3.5: **Geometrical features.** Distributions of the area (μm^2) (a) and maximum Feret diameter (μm) (b) across all annotated cells.

Figure 3.4b illustrates the same concept but focuses on the relative proportion of background to signal. The distribution is left-skewed, with a lower half concentrated in the range (19, 291), i.e. background pixels are roughly from 20 to 300 times the signal pixels in 50% of the images. Remarkably, the disproportion grows even faster in the right tail, where the ratio explodes up to over 1000. Finally, notice that the bulk of outliers accumulates in the higher end of the domain. This is caused by the contribution of empty masks that cover more than 10% of the total images.

These considerations expose the need for dedicated training strategies to face this strong class imbalance and correctly learn to classify image pixels.

3.2.3 Objects features

After the initial exploration of the data characteristics at the pixel level, additional investigations can be devoted to discovering meaningful insights about images' macroscopic content. The Fluorescent Neuronal Cells pictures present a rich collection of 2137 neuronal cell instances of various shapes and sizes (see *area*, and

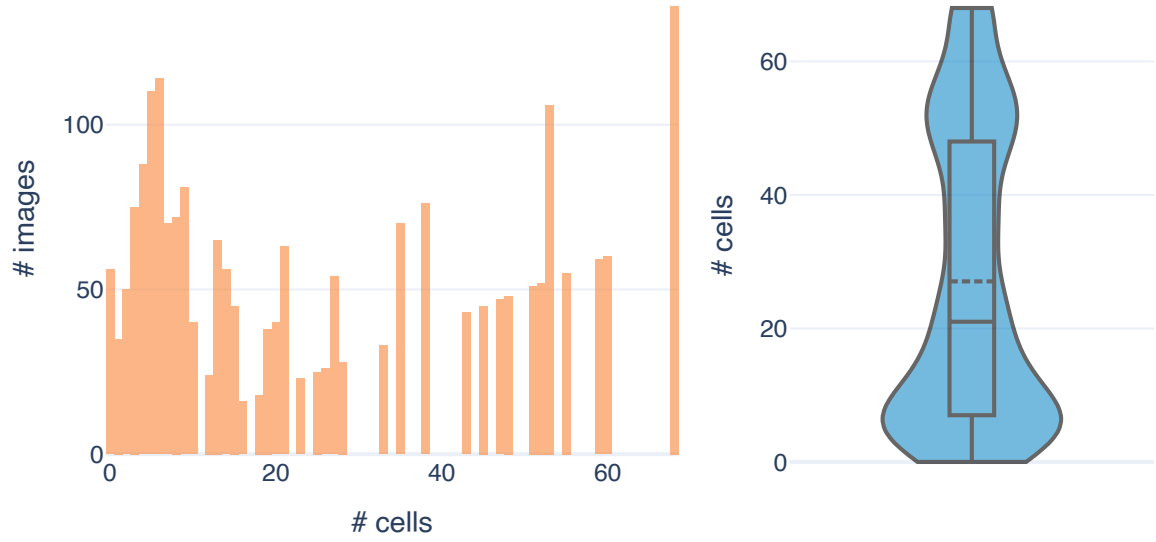
Feret diameter columns in Table 3.1 for a numeric summary) that are unevenly distributed across the images.

Figure 3.5 shows the distributions of the most interesting geometrical features of the annotated objects. Regarding the area (Fig. 3.5a), the bulk of the distribution presents cells with a surface within 55.51 and 148.02 μm^2. Figure 3.5b reports the maximum Feret diameter (Merkus, 2009) instead. This measure is computed as the longest distance (in pixels) between points of a convex cell countour[6]. In both cases, the distribution is left-skewed, with a slight prevalence of values lower than the median. In fact, 90% of objects are small and medium cells with prevalently regular circular shapes, having an area in the range [35.23, 237.09] μm^2 and a Feret diameter between 9.42 and 27.61 μm. The remaining 10% of the distribution stretches up to a maximum of 796.39 μm^2 and 67.48 μm, respectively. This effect is due to the contribution of more oversized or prolonged objects that cause a long, heavy tail.

Finally, 3.6 illustrates the distribution of the number of cells across the dataset (see *# cells/image* column in Table 3.1 for a numeric summary). In this case, the distribution presents multiple modes that can be summed up by the five major peaks, namely 6, 35, 38, 53 and 68 (Fig. 3.6a). The empty spaces are a consequence of the fact that not all of the possible values were actually observed in the data. Interestingly, a lower peak is observed at 0 because of the 56 images where no cells were annotated.

By looking at the estimated density in the violin plot (Fig. 3.6b), it appears that the distribution can be interpreted as a mixture of two components. The first is centered around 6 and is made of the images with lower counts, i.e. the ones depicting brain areas where the fluorophore did not yield abundant emissions. The second, instead, is a combination of the four higher peaks that represent areas having anatomical connections with the injection site (i.e. they project into where the tracer was injected in the first place).

[6]obtained using skimage package version '0.18.1'



(a) Histogram (b) Violin plot

Figure 3.6: **Counts distribution.** Distributions of the number of annotated cells across all images in the dataset.

3.3 Challenges

Although many efforts were made to stabilize the acquisition procedure, the images present several relevant challenges for the detection task.

A first source of complexity is given by the high variability in terms of color, saturation and contrast from image to image. For example, sometimes the tissues can soak in some of the marker (see Fig. 3.1e), causing irrelevant compounds to emit light which is then captured by the microscope. When that is the case, the background's hue shift towards values similar to the ones of some fainted neuronal cells (cf. Fig. 3.1c). In such circumstances, the sheer pixel intensity is not enough to distinguish between signal and background, which forces the identification to fall back to other characteristics such as saturation and contrast. However, the latter is likewise not trivial as fluorescent emissions are naturally unstable, thus generating fluctuations of the saturation levels exhibited by cell pixels (cf. Figs. 3.1c and 3.7a).

Moreover, the substructures of interest have a fluid nature. Also, the shot can capture different two-dimensional sections depending on how the cells are oriented within the tissues. As a consequence, the size and the shape of the stained cells may

change significantly (see Figs. 3.5a and 3.7a), making it even harder to discriminate between them and the background.

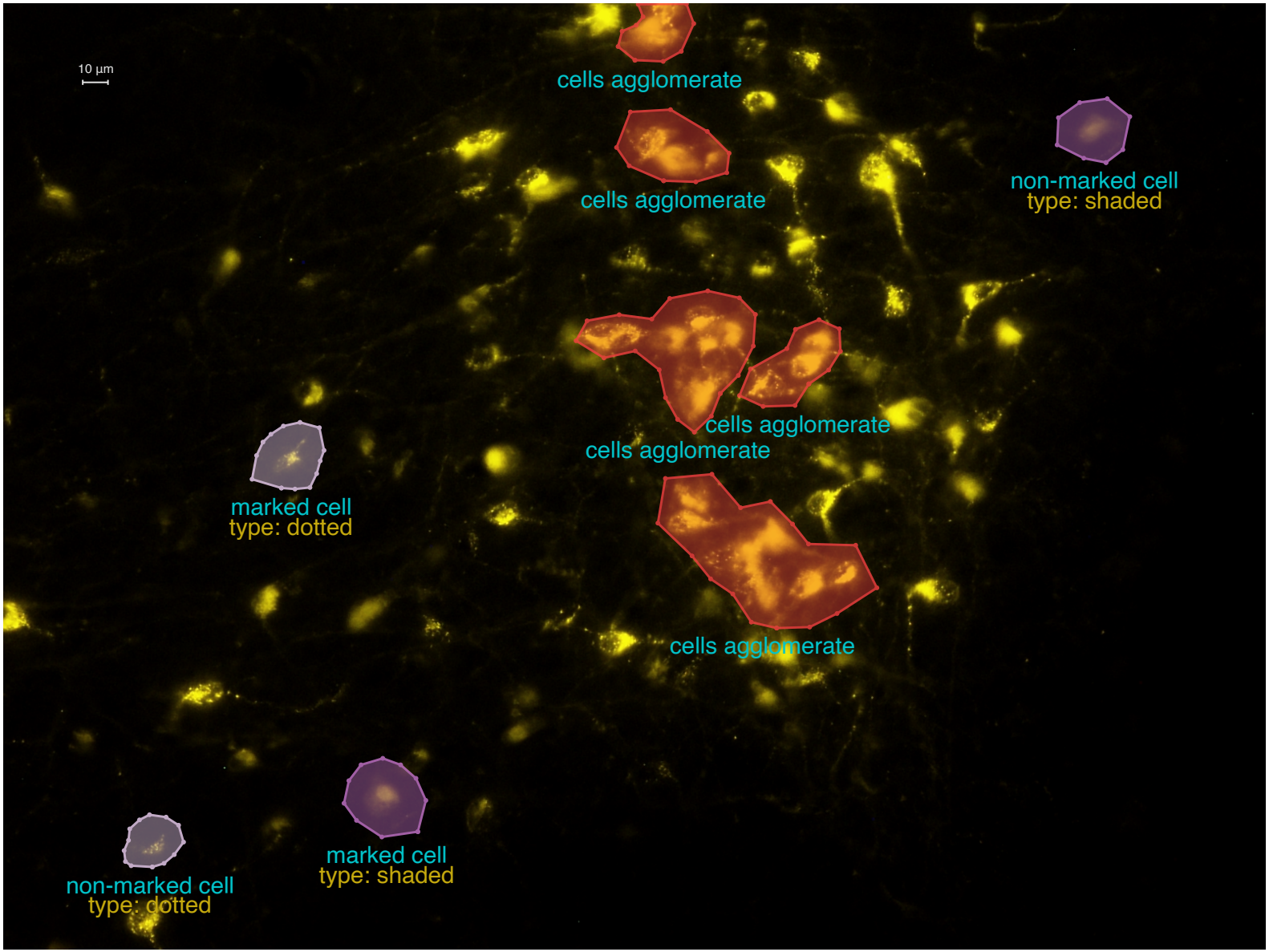
Another challenge is due to the occasional presence of accumulations of fluorophore in narrow areas that generate emissions very similar to the ones of cells. When that happens, the pictures may contain fictitious objects or uninteresting structures that resemble neuronal cells in terms of shape, size or color. These artifacts may vary from small areas – as in the case of point artifacts and filaments (see Figs. 3.7b and 3.7c)– to bigger structures as the stripe in Fig. 3.7b or the "macaron"-shaped object in Fig. 3.7c. Again, their presence hampers the detection task, making the recognition and the understanding of cells structure and size mandatory for the model.

A further source of complexity is represented by the broad shift in the number of target cells from image to image. Indeed, the total counts range from no stained cells (Fig. 3.1a) to several dozens clumping together (Fig. 3.7a). As a consequence, a certain degree of flexibility is required for the model so to handle both cases. In particular, the precise localization of cell boundaries may be hard to achieve in the presence of overcrowding, and some escamotages may be necessary to avoid close-by cells are joint in single agglomerates by the model.

Furthermore, the objects are typically small and cover only marginal portions of the images. This generates an extreme imbalance between signal and background (see Section 3.2.2), which is even worsened by the high resolution of the pictures. Hence, dedicated learning strategies are demanded to mitigate this issue during the training phase.

Last but not least, in some occasions the recognition of cells may be ambiguous even for human operators. Of course, this poses an issue of intrinsic subjectivity in the annotation process, which in turn affects both the training and assessment phases.

By and large, all of these factors make the recognition and counting tasks harder and complicate the learning process. Likewise, borderline annotations hinder model evaluation as their subjectivity deprives the model of a reliable and indisputable testbed.



(a) Challenges

Figure 3.7: **Challenges and artifacts.** Some of the images present cells agglomerate that require sharp boundary segmentation. Also, marked cells may look very similar to non-marked objects due to an intrinsic arbitrariness of the recognition task.

(b) Biological artifacts

Figure 3.7: **Challenges and artifacts. (2)** Although biological structures as the tissue border (stripe) or the axons (filaments) naturally have similar emission properties compared to neuronal cells, they are not of interest and ought to be discarded by the model.

(c) Technical artifacts

Figure 3.7: **Challenges and artifacts.** **(3)** The fluorophore may accumulate in small (point-artifacts) or even large ("macaroni"-shaped artifact) areas; this causes emissions hard to distinguish from cells just looking at the pixels' color features.

Chapter 4

Methods

This chapter discusses a deep learning attempt to tackle the problem of segmenting and counting cells in microscopic fluorescence using a *supervised learning* framework. For this purpose, we leverage the recently proposed **cell-ResUnet** architecture (Morelli et al., 2021) for object segmentaion, and we adopt careful training strategies to address the principal challenges of the Fluorescent Neuronal Cells dataset (Clissa et al., 2021) described in Section 3.3. Once the cells are detected, the final count is retrieved as the number of connected pixels in the post-processed output. In order to demonstrate the efficacy of our method, we compare the cell-ResUnet against three CNN architectures belonging to the Unet and ResUnet families. In doing so, we also test the impact of study design choices intended to reduce false negatives and promote accurate segmentation. In addition to that, an adaptive thresholding approach is also tested as a baseline for performance measurement.

4.1 Non-ML baseline

Machine and deep learning have succeeded in many applications from several domains lately, thus building great expectations and becoming hot topics in current innovation processes at various societal levels. Nevertheless, these powerful techniques are not a magic bullet to solve any data-related problem (Wolpert & Macready, 1997). They come with their own challenges and limitations that are often overlooked in real-world applications, possibly causing thunderous failures due to unjustified expectations. In fact, it is not unusual to fall victim to their popularity only to find months down the line that more straightforward methods work best for some specific task or data.

To avoid incurring this "ML hype curse", this work considers a simple non-ML

approach as a baseline. In particular, this consists in an adaptive thresholding mechanism implemented to exploit the pixel intensity information for binarization. In practice, the input image is read as grayscale and a selection cutoff is set at a configurable quantile of the pixel intensity distribution. A binary mask is then obtained by labeling pixels above that threshold as cells and the rest as background. After that, the same post-processing steps as the ones adopted for the output of the ML models are applied (see Section 4.5). This operation is performed for all training and validation pictures, and the goodness of fit is assessed as described in Section 4.6. The whole procedure is then repeated varying the quantile value used for thresholding. A starting search is conducted using a coarse grid from 0.9 to 0.99 with steps of 0.01. This is intended to explore the hyperparameter space and get an idea of where the approach performs best. Since this happens for higher values, a finer grid from 0.97 to 0.999 with steps of 0.001 is exploited for fine-tuning. Finally, the cutoff corresponding to the highest F_1 score is chosen as the optimal threshold and is later used to assess the baseline performance on the test set.

4.2 Model architecture

This section describes the four architectures adopted for our studies. In particular, we tap into two network families commonly used for segmentation tasks, i.e. Unet and ResUnet, and experiment with two alternative implementations for each family. The following subsections discuss the two families and each architecture in greater detail.

4.2.1 Unet architectures

The Unet architecture was introduced in Ronneberger et al. (2015), explicitly targeting the task of semantic segmentation in biomedical images. The idea behind this approach is to build upon a fully convolutional network (Long, Shelhamer, & Darrell, 2015) that extends the CNN models commonly seen for classification. In particular, the proposed architecture comprises a usual contracting branch supplemented with successive layers where upsampling operators replace pooling counterparts (Fig. 4.1).

The first network portion is built upon a classical block, hereafter referred to as *unet block* or *units*. The former is made of two convolutional layers using a varying number of 3×3 filters without padding (Goodfellow, Bengio, & Courville, 2016, Section 9.1), each followed by a *rectified linear unit* ($ReLU$) activation function (Fukushima & Miyake, 1982; Nair & Hinton, 2010 and Goodfellow et al., 2016,

Section 6.1). This generates an equivalent number of feature maps (or channels) of approximately the same shape as the input image. After the previous processing, the resulting feature maps undergo a 2×2 max-pooling operation (Weng, Ahuja, & Huang, 1993 and Goodfellow et al., 2016, Section 9.3) with stride 2, thus downsampling the input size. In practice, the contracting branch takes a grayscale image as input and applies four consecutive unet blocks. In the first step, 64 filters are adopted, and the downsampling path then continues roughly halving the input size at each block, meanwhile doubling the number of feature channels. The resulting output consists of 1024 feature maps having approximately 1/16 the size of the original image. Finally, the contracting branch ends with the so-called *bottleneck* layer, whereby two `convolution-activation` blocks are applied.

Conversely, an almost symmetrical upsampling structure follows in the second part of the network, from which the *U-shape* comes. This time the unet blocks are similar to the ones of the contracting branch except that max-pooling operations are replaced by 2×2 up-convolutions (Dumoulin & Visin, 2016; Zeiler, Krishnan, Taylor, & Fergus, 2010) and long-range concatenations. In this way, the up-convolutions enlarge individual images to nearly double their size. At the same time, the feature channels are halved at each step by the upsampling layer. Although this part is crucial to returning an output of a similar shape as the input, the upsampling generally produces blurred pixels that hamper precise localization. For this reason, each up-convolution is concatenated with *long-range connections*, i.e. the feature channels are integrated by copying a crop of compatible size from the corresponding downsampling block. As a result, this escamotage establishes a flow of information between lower and higher layers such that both low-level finer details and high-level semantic features contribute to the learning process. In practice, a total of three upsampling unet blocks are applied in the expanding path, and the architecture terminates with an additional convolutional block. Specifically, the latter comprises two of the previous `convolution-activation` blocks followed by a last 1×1 convolutional layer to map the resulting 64-component feature vector to the desired number of classes. In the case of cell segmentation, this corresponds to two feature maps which can be interpreted as pixel-level scores for the classification as 0 (background) or 1 (signal) classes.

In summary, the Unet architecture aims to create an encoding branch intended to capture relevant features that enable the recognition of the objects without caring where they are placed. Once this goal is achieved, the decoding structure is then applied to refine the localization of the detected objects.

This work considers two implementations of the above structure. The first one corresponds to the original version of Ronneberger et al. (2015) except for the padding strategy – we use padded convolution to retain the initial size. The resulting model will be referred to as *Unet* in the following. The second, instead, consists

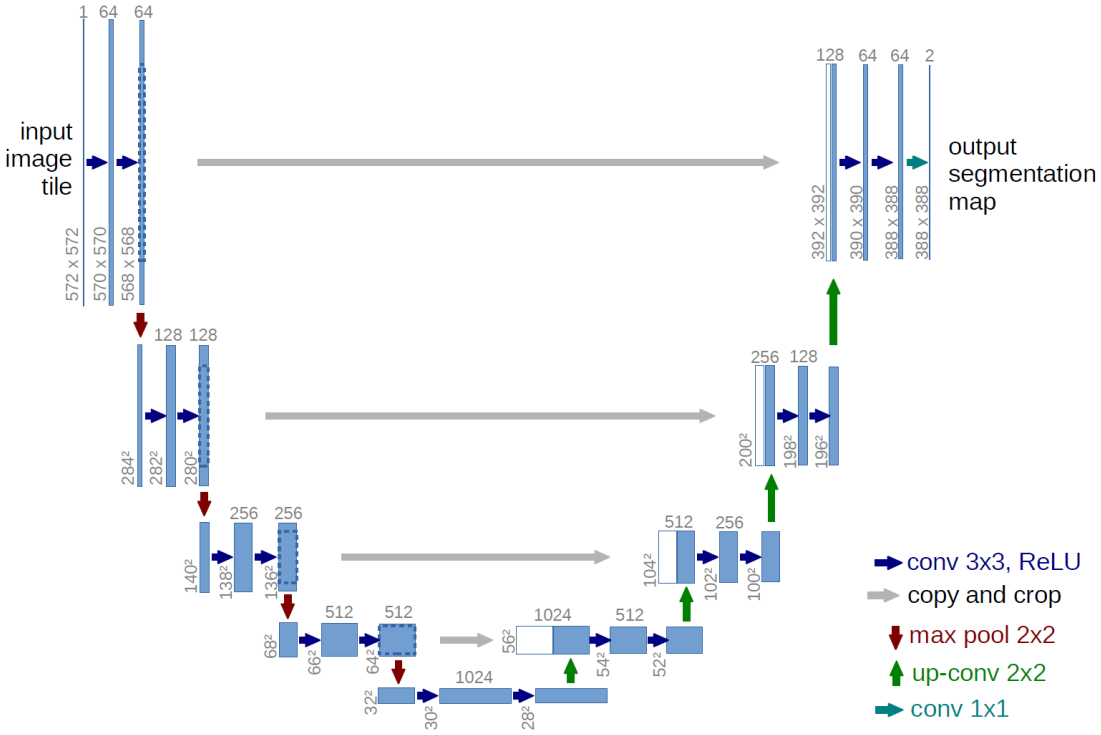


Figure 4.1: **Unet architecture** *(example for 32x32 pixels in the lowest resolution).* Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. This figure is borrowed from Ronneberger et al. (2015).

of a lighter version obtained by setting the initial number of filters equal to 16 and scaling the following units consequently. Importantly, three unet blocks are used (plus the bottleneck) rather than the four of the original implementation. Both of the previous choices are motivated by the intent of testing a unet-like implementation with a comparable number of parameters with respect to the ResUnet alternatives (see Section 4.2.2 for more details). However, a trivial transposition of the original unet blocks does not work in practice for our use case, so we resort to the insertion of additional batch normalization (Goodfellow et al., 2016, Section 8.7.1) layers after each `convolution-activation` block. The resulting "light" unet architecture will be referred to as *small Unet* hereafter.

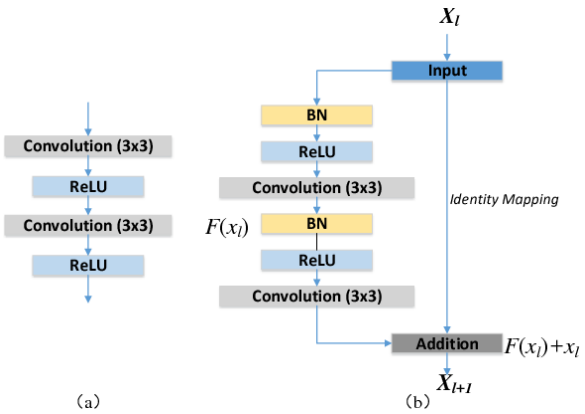


Figure 4.2: **Unet versus Residual units.** Building blocks of neural networks. (a) Plain neural unit used in the Unet. (b) Residual unit with identity mapping used in the ResUnet model. This figure is borrowed from Z. Zhang et al. (2018).

4.2.2 ResUnet architectures

Building on the architecture described in Section 4.2.1, multiple variants have been proposed, each exploiting more or less substantial tuning of some of its building blocks. One of such derived versions is the so-called *deep ResUnet* (Z. Zhang et al., 2018), whose main difference is the adoption of *residual units* (K. He et al., 2016) instead of unet blocks (see Fig. 4.2 for a visual comparison).

The motivation behind the last modifications is to contrast the vanishing gradient (Hochreiter, 1998) issue that generally affects the Unet models. In practice, the deep architecture poses a severe challenge related to the backpropagation of the gradients during the learning phase. In fact, each training iteration updates the network parameters by an amount proportional to the partial derivative of the loss function with respect to the previous weight values. Specifically, the backpropagation algorithm is based on chained multiplications that propagate such gradients backward in the network to update all layers' parameters. However, this reverse

flow may entail repeated multiplications of small values (≤ 1) that cause the back-propagated gradients to "vanish" – i.e. to approach 0. When that is the case, the update rule implies that network weights are also shrunk towards 0. Consequently, the corresponding connections between neurons will be deactivated in the next forward pass, thus preventing the full exploitation of the available information and hampering the learning phase. Of course, the deeper the architecture, the higher the chances of incurring this drawback. In this regard, the residual units represent an ingenious gimmick to mitigate the impact of vanishing gradients. The idea is to leverage an identity mapping (also referred to as short-connection) parallel to the usual convolutional block, which is added to the result of the convolutions. In this way, the input information is directly propagated to the next layers and considered should it be relevant for the subsequent processing. Also, batch normalization layers are typically added to stabilize the range of the input values. As a result, residual units typically guarantee comparable performance with fewer parameters, meanwhile enhancing the convergence speed.

Given the advantages described above, we experiment with two architectures belonging to the ResUnet family inspired by the implementation presented in Z. Zhang et al. (2018) (see Fig. 4.3). In particular, a first version is obtained with the same components as in Fig. 4.3, but using 16 initial filters – instead of the 64 adopted in the original version (Z. Zhang et al., 2018) – and scaling the following structure consequently. Also, we adopt the exponential linear unit (Elu) (Clevert, Unterthiner, & Hochreiter, 2015) rather than the ReLU activation. The resulting model will be referred to as *ResUnet* in the following.

Finally, we propose a slight modification to the above architecture specifically developed for our use case. In particular, we add an initial 1×1 convolution to simulate an RGB to grayscale conversion. The advantage of doing so – as opposed to applying a standard grayscale conversion – is that the transformation is learned during training to improve the segmentation performance. As a further modification, we insert an additional residual block having 5×5 filters – instead of 3×3 – at the end of the encoding path. This adjustment should provide 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 (Figs. 3.1e, 3.7b and 3.7c) can be improved by looking at a broader context. The resulting architecture is reported in Fig. 4.4 and it will be referred to as **cell ResUnet (c-ResUnet)** hereafter.

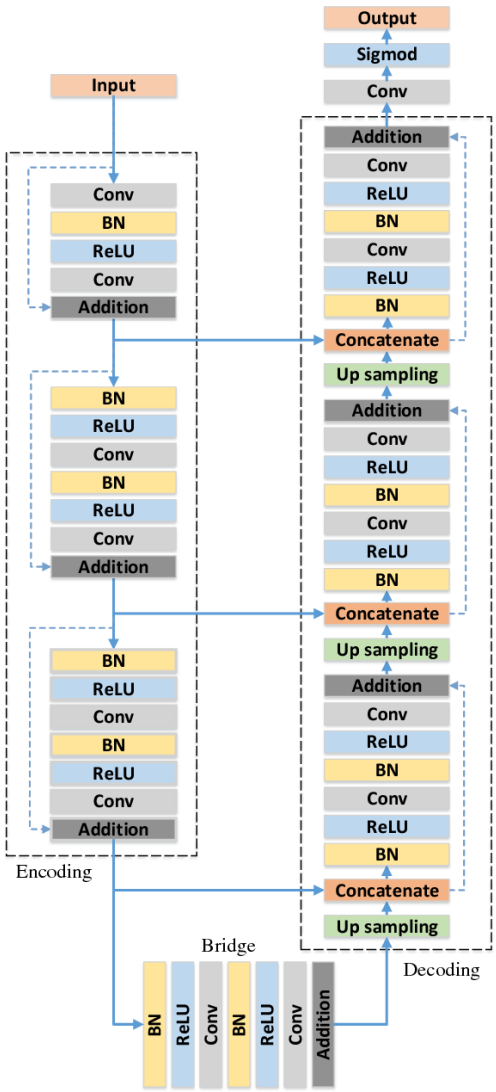


Figure 4.3: **Deep ResUnet architecture.** This figure is borrowed from (Z. Zhang et al., 2018).

4.3 Ablation studies

Alongside comparing different approaches, we also tested the effect of two design choices intended to mitigate errors on challenging images containing artifacts and cell overcrowding.

4.3.1 Artifacts Oversampling (AO)

The presence of biological structures and technical artifacts like stripes, filaments (Fig. 3.7b) or accidental flourophore accumulations (Fig. 3.7c) can often fool the model into detecting false positives. Indeed, their similarity with cells in terms of saturation and brightness hampers their correct handling and entices the model to overpredict. This situation is worsened because such structures are underrepresented in the data, especially concerning the stripes and the macaroni-shaped artifact. In fact, only a handful of artifact examples are available, which complicates their recognition even further.

For this reason, we tried to increase the augmentation factor for these inputs to facilitate the learning process. Specifically, we selected the six different crops containing stripes and re-sampled them with the augmentation pipeline described in Section 4.4, resulting in 150 new images for each crop.

4.3.2 Weight Maps (WM)

One of the toughest challenges during the inference is related to cell overcrowding. In particular, the cells sometimes tend to form agglomerates of several overlying neurons, making it difficult to reconstruct their individual shapes (see Fig. 3.7a). Although the human eye can often trace back the morphology of the original neurons, perhaps even piecing together missing/separated parts due to the superposition of other cells, this task is still much harder for automatic approaches. In particular, precise object segmentation is paramount when clumps of objects are present in the images, as failing to achieve a nitid distinction of cells boundaries may lead to spurious connections between separated objects. If that happens, multiple objects are considered as a single one and false negatives are generated, thus deteriorating the model performance.

In order to improve cell separation, Ronneberger et al. (2015) suggest leveraging a weight map that penalizes more the errors on the borders of touching cells. Building on that, we introduce a novel implementation where single object contributions are compounded additively. This procedure generates weights that decrease as we move away from the borders of each cell. At the same time, the contributions coming from single items are combined so that the global weight

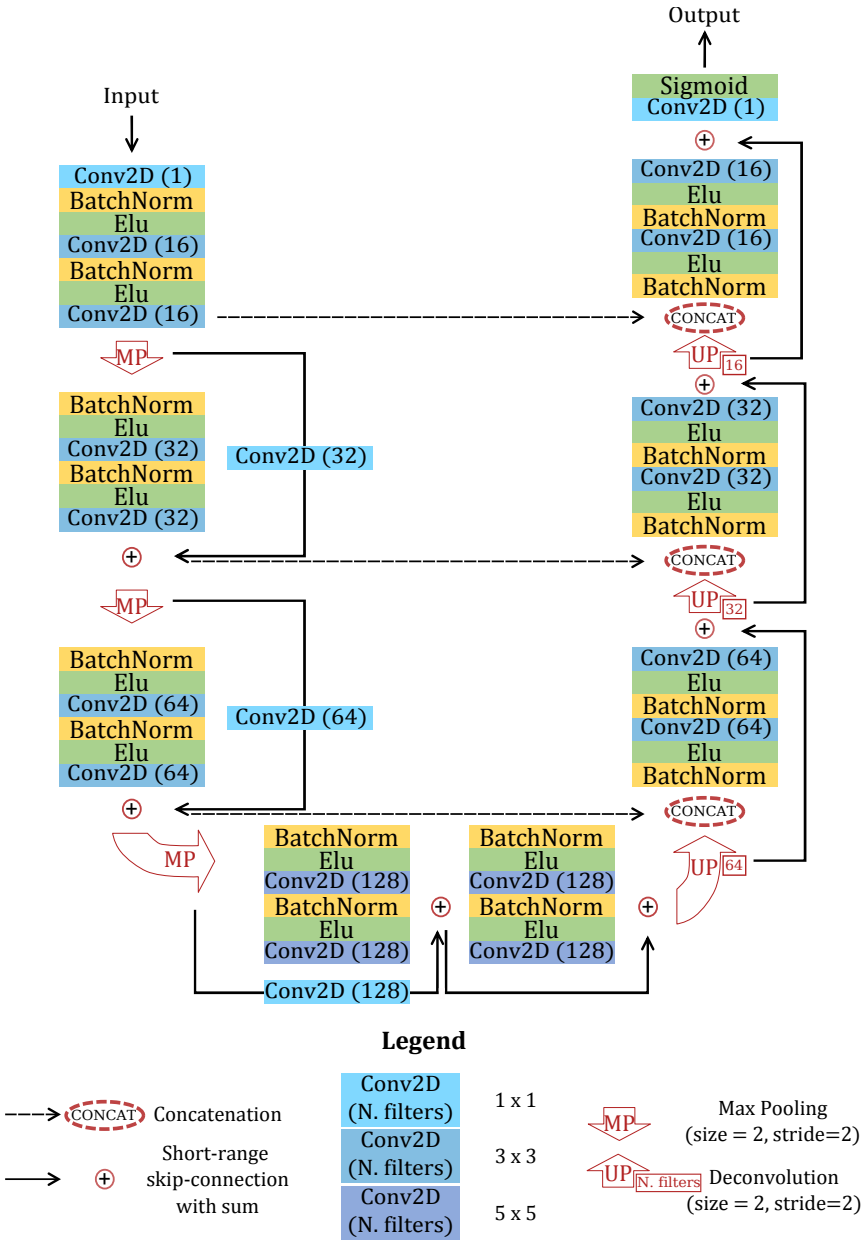


Figure 4.4: **c-ResUnet architecture.** Each box reports an element of the entire architecture (individual descriptions in the legend).

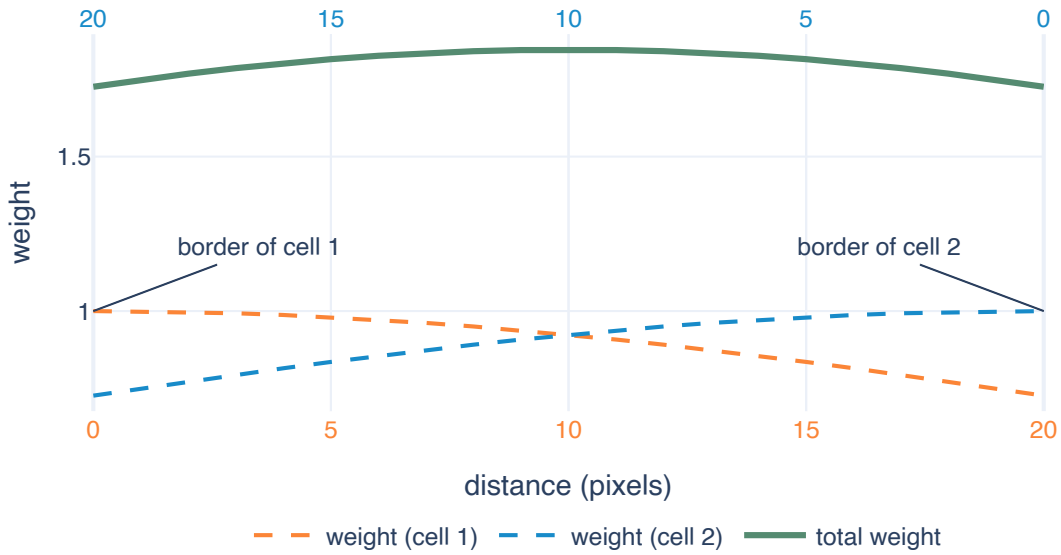


Figure 4.5: **Weight compounding.** The dashed curves depict the weights generated by single cells as a function of the distance from their borders according to Eq. (4.1). The green line illustrates the final weight obtained by adding individual contributions.

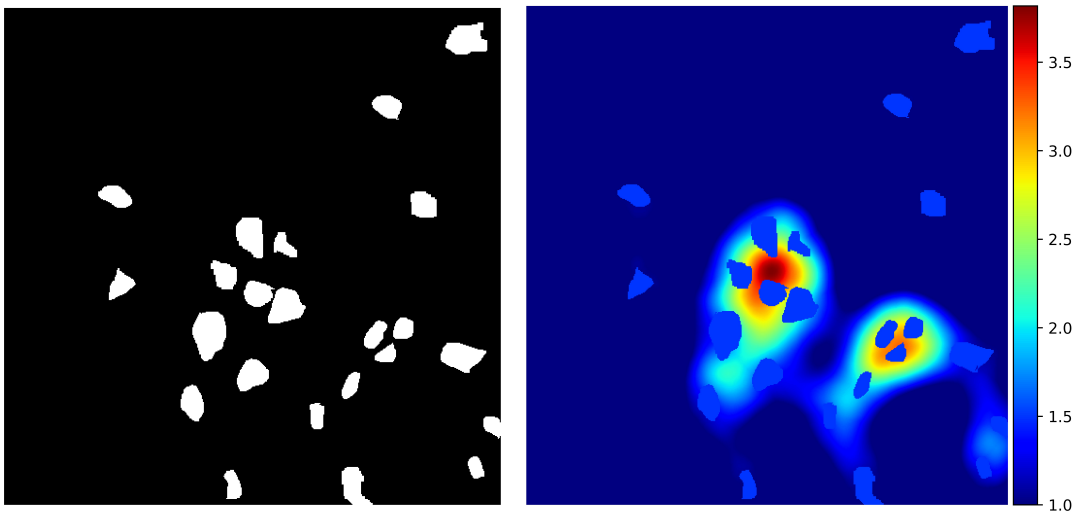
map presents higher values where more cells are close together (see Fig. 4.5). The pseudocode[1] for a weight map is reported in Alg. 1, and an example weight map is shown in Fig. 4.6.

With respect to the original Unet implementation, the main difference lies in the fact that all cells contribute to penalizing each pixel, at least in principle. In contrast, only the two closest cells have an impact in Unet's version. In this way, the weight is increased where more than two cells are close together, producing a greater penalization in crowded areas. Also, our implementation does not use the somewhat arbitrary parameter w_0 (cf. the original implementation in Ronneberger et al. (2015)). Finally, our weight map's σ parameter is tied to the average cell radius, whereas Unet's value does not seem attached to any tangible reference or meaning.

4.4 Model training

After randomly setting 70 full-size images apart as a test set, the remaining pictures were randomly split into training and validation sets. In particular, twelve

[1] full implementation available at: https://github.com/robomorelli/cell_counting_yellow/tree/master/030_weights_maker.py



(a) Mask (b) Weight map

Figure 4.6: **Weight map.** A target mask and the corresponding weight map.

Algorithm 1: weight map pseudocode for j-th mask.

1 Initialize empty map$_j$ (mask size) `// weight map` j`-th mask`

2 **for** *each cell in mask* **do**

`/* loop over` i`-th cell in` j`-th mask */`

3 Initialize empty map$_i$ (mask size) `// weight map` i`-th cell`

4 Add i-th cell to map$_i$

5 Compute euclidean distance between each pixel of map$_i$ and the closest pixel of the i-th cell

6 Compute each pixel's weight in map$_i$ according to a decreasing exponential function:

$$\text{weight} = \exp\left\{\frac{-d^2}{2\sigma^2}\right\} \tag{4.1}$$

`/*` d `is the distance computed at step 5 */`

`/*` σ `is a customizable parameter set to 25 (average cell radius) */`

7 Sum the resulting map$_i$ to the full map$_j$

512×512 partially overlapping crops were extracted from each image and fed as input to the network after undergoing a standard augmentation pipeline. Common transformations were considered as rotations, addition of Gaussian noise, brightness variation and elastic transformations (Simard, Steinkraus, & Platt, 2003). The augmentation factors of the crops were fixed differentially based on their contents. The six patches included in the artifact oversampling ablation study were re-sampled 25 times each. Instead, all the remaining crops produced 10 augmented versions for manually segmented images and 4 for all the others. As a result, the model was trained on a total of nearly 16000 images (70% for training and 30% for validation).

All competing architectures were trained from scratch under the same conditions to favor a fair comparison. Specifically, the Adam (Kingma & Ba, 2017) optimizer was employed with an initial learning rate of 0.006. A scheduled decrease of 30% was then applied if the validation loss did not improve for 4 consecutive epochs. A **weighted binary cross-entropy** loss was adopted on top of the weight maps to handle the imbalance of the two classes (weights equal to 1.5 and 1 for cells and background, respectively). All models were trained until no improvement was observed for 20 consecutive epochs. In this way, each model was allowed to converge and the comparison was made at the best of each architecture's capabilities. In terms of convergence speed, all the models required less than 100 epochs and the training was performed in the scale of a few hours of runtime.

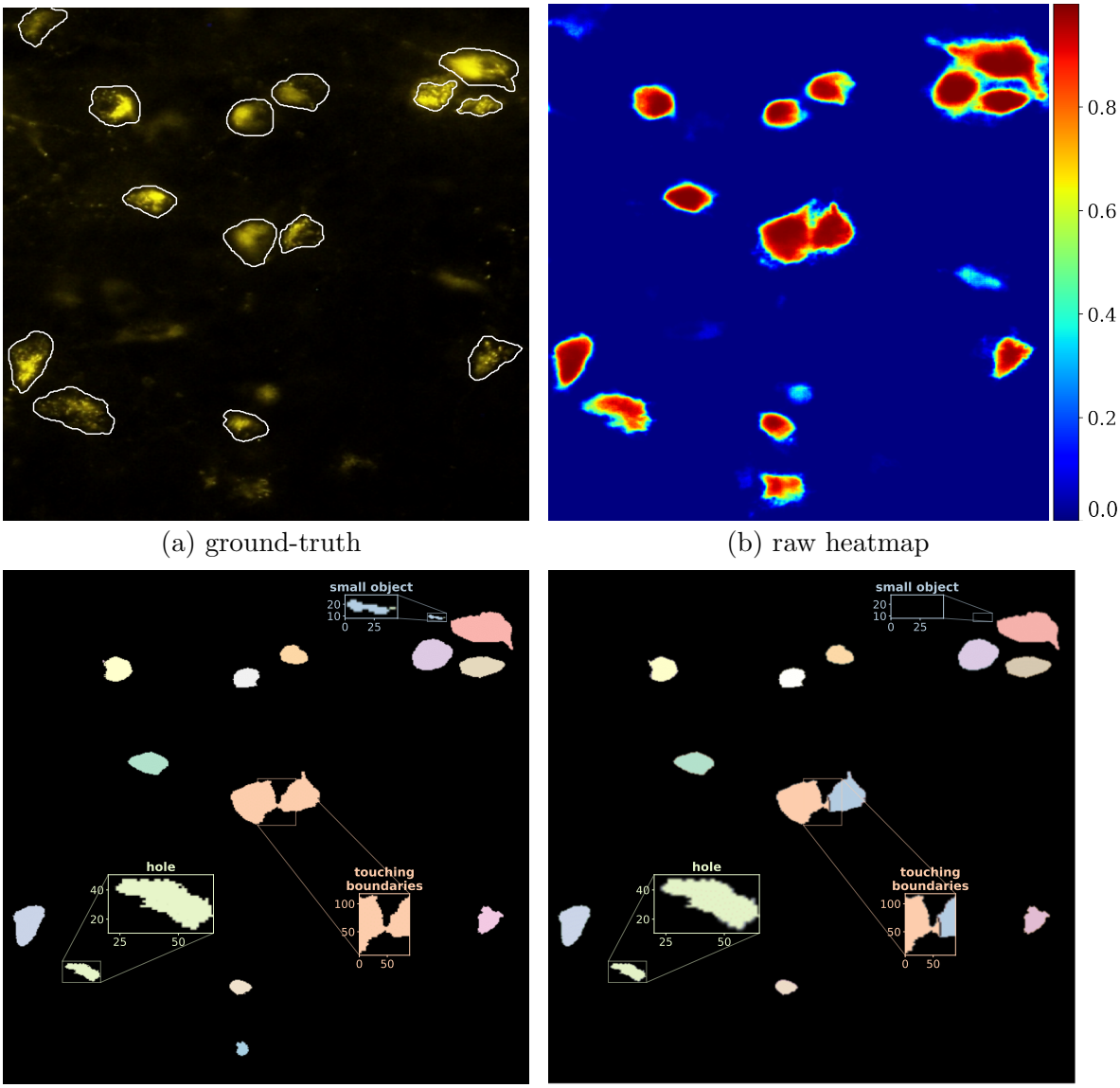
The approach was implemented through Keras API (Chollet et al., 2015) using `TensorFlow` (Abadi et al., 2015) as backend. For more details, please refer to the GitHub repository[2]. The training was performed on 4 V100 GPUs provided by the *Centro Nazionale Analisi Fotogrammi* (CNAF)[3] computing center of the *National Institute for Nuclear Physics*[4] in Bologna.

4.5 Post-processing

This section discusses the full inference pipeline adopted in our approach, with a particular focus on the post-processing impact. Figure 4.7 reports an illustration of the above process for a sample input image (Fig. 4.7a).

When the picture is passed through the network, the raw output consists of a probability map (or *heatmap*, see Fig. 4.7b) of the same size as the original input, whereby each pixel value can be interpreted as the probability of belonging to a cell. The higher the value, the higher is the model's confidence in classifying that pixel

[2]available at: https://github.com/robomorelli/cell_counting_yellow/tree/master
[3]for more details: https://www.cnaf.infn.it/en/
[4]for more details: https://www.bo.infn.it/en/welcome-to-infn-bologna-unit/



(a) ground-truth (b) raw heatmap

(c) thresholded prediction (d) post-processed prediction

Figure 4.7: **Model output.** (a) the input image with white contours indicating annotated cells; (b) the model's raw output; (c) the predicted mask after thresholding at 0.875; (d) the predicted mask after post-processing.

as the signal. A thresholding operation is then applied on the heatmap to obtain a binary mask where the pixels above the cutoff are represented in white (signal) and the rest in black (background). Hence, groups of white connected pixels represent the detected cells. The result of thresholding for the sample image is reported in Fig. 4.7c, where different colors are added for visualization purposes to highlight different detected objects after the binarization. In this case, the model seems to perform reasonably well in detecting cell instances, although not being likewise accurate in the segmentation task (cf. the white contours in Fig. 4.7a that highlight the annotated cells). Nonetheless, some pathological behaviors are clearly visible. First, the confidence of the model is not always uniform over all the pixels belonging to the same object. For instance, its value is typically higher for the central regions of the detected cells and it deteriorates moving towards the boundaries, as expected. However, this damping behavior is not always homogeneous, and a confidence decrease followed by a successive increase can be observed going towards the cell edges. When that happens, depressed regions are formed inside the objects, and the thresholding operation risks filtering them out if their values are below the adopted cutoff. As a result, single cells may be split into multiple instances, or their body may present internal holes. Examples of those behaviors are the purple-ish cell in the top-right corner, for which the left tail filament has been disconnected (cf. Figs. 4.7b and 4.7c), and the green-ish object at the bottom-left, respectively. Another limitation is related to the overcrowding problem discussed in Sections 3.3 and 4.3.2, as highlighted by the two close-by cells in the center of the image. In this case, the heatmap suggests that the model cannot draw sharp boundaries around the cells, thus failing to separate them in the binary output.

The previous observations suggest that a smarter post-processing may be required for better performances. For this reason, we adopt ad-hoc post-processing to tackle the issues above. In particular, we first remove isolated components of a few pixels and fill the holes inside the detected cells. Then, we employ the watershed algorithm (Soille & Ansoult, 1990) to separate close-by objects. An example of the results is provided in Fig. 4.7d, where the overlapping cells in the middle of Fig. 4.8 are correctly split after post-processing. Also, the small component in the top-right corner is removed, and the hole in the bottom-left object is filled.

Unfortunately, this step is customized for our dataset, which hinders its extension to other applications. However, the parameters for the above operations are set quite trivially based on the average cell size. Therefore, our approach may reasonably apply to other use cases as far as ground-truth masks are available for extracting object statistics[5].

[5]for more details, please check: https://github.com/robomorelli/cell_counting_yellow/blob/822944cf91a60c8aae6671d05fbc1c23728cad68/evaluation_utils.py#L50

4.6 Model evaluation

All the presented approaches were evaluated and compared based on both detection and counting performance. Also, ablation studies were conducted to assess the impact of artifacts oversampling and weight maps.

In order to evaluate the detection ability of the models, a dedicated algorithm was developed. Specifically, each predicted object is compared to all cells in the corresponding ground-truth label and uniquely associated with the closest one. If the distance between their centroids is less than a fixed threshold (50 pixels, i.e. average cell diameter), the predicted element is considered a match and it increases the true positive count (TP). At the end of this procedure, all true objects without matches are considered false negatives (FN). Likewise, the remaining detected items not associated with any target are considered false positives (FP). Algorithm 2 reports the pseudocode of the procedure described above[6]. Starting

Algorithm 2: metrics computation for i-*th* image.

Input: pred_i, mask_i
Output: TP, FP, FN

```
1  Set TP, FP, FN = 0
2  Get predicted objects, pred_objs^i                    // detected cells
3  Get true objects, true_objs^i                         // annotated cells
4  Get predicted centers, pred_ctrs^i
5  Get true centers, true_ctrs^i
6  for each ctr_j in pred_ctrs^i do
       /* loop over predicted centers                                  */
7      for each ctr_k in true_ctrs^i do
           /* loop over true centers                                   */
8          Compute euclidean distance between ctr_j and ctr_k
9          Store distance and indexes
10     Compute the minimum, min_dist_i of the distances stored in step 9
11     if understand then
12         Increase true positives, TP
13         Remove ctr_j from pred_ctrs^i
14         Remove ctr_k from true_ctrs^i
15 Compute false negatives as true_objs^i - TP
16 Compute false positives as pred_objs^i - TP
```

[6]full implementation available at: https://github.com/robomorelli/cell_counting_yellow/blob/822944cf91a60c8aae6671d05fbc1c23728cad68/evaluation_utils.py#L102

from these values, we resort to accuracy, precision, recall and F_1 score as indicators of detection performance. The definitions of such metrics are reported below:

$$\text{accuracy} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} = \frac{1}{1 + \frac{1}{\text{TP}}(\text{FP} + \text{FN})}; \tag{4.2}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}; \tag{4.3}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}; \tag{4.4}$$

$$F_1\text{score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} = \frac{1}{1 + \frac{1}{2\text{TP}}(\text{FP} + \text{FN})}. \tag{4.5}$$

Notice that we do not have true negatives in Eq. (4.2) since the prediction of the class "not cell" is done at the pixel level and not at the object level, so there are no "non-cell" objects predicted by the model. The above metrics have the limitation of being dependent on the threshold used for the binarization of the predicted heatmaps. Thus, we also look at the *precision/recall curves* and the corresponding Area Under the Curve (AUC) (Hanley & McNeil, 1982; Mason & Graham, 2002) for a more general measure of performance. For this purpose, we consider only test images and repeatedly compute precision and recall for each model, varying the binarization threshold from 0.05 to 1 with steps of 0.05 – these cutoffs represent quantiles of the grayscale intensity distribution in the case of the non-ML baseline.

Regarding the counting task, the Mean Absolute Error (MAE), Median Absolute Error (MedAE) and Mean Percentage Error (MPE) are used instead. More precisely, let n_{pred} be the number of detected cells in i-th image and n_{true} be the actual one. Then, the absolute error (AE) and the percentage error (PE) are defined as:

$$\text{AE} = |n_{\text{true}} - n_{\text{pred}}|; \tag{4.6}$$

$$\text{PE} = \frac{n_{\text{true}} - n_{\text{pred}}}{n_{\text{true}}}. \tag{4.7}$$

Hence, the above counting metrics are just the mean and the median of the AE and the PE. Although these quantities are intuitive and straightforward to interpret, they may hide poor performances when the counts' distribution has low variability. For this reason, we also report the R^2 coefficient of determination that represents

the percentage of variability explained by the model:

$$R^2 = 1 - \frac{\mathrm{SS}_{\mathrm{residual}}}{\mathrm{SS}_{\mathrm{total}}} = 1 - \frac{\sum_i^n (y_i - \hat{y}_i)^2}{\sum_i^n (y_i - \bar{y})^2}, \tag{4.8}$$

where y_i is the count of the i-th image, $\hat{y}_i$ is the corresponding predicted count and $\bar{y}$ is the average count. Hence, values close to 1 indicate that most of the intrinsic variability of the counts is correctly modeled, whereas lower values (close to 0) are a signal of poor predicting ability.

4.6.1 Threshold optimization

The choice of the optimal cutoff for binarization is based on the F_1 score computed on full-size images. In practice, the DL models are evaluated on a grid of values and the best one is selected according to the *Kneedle* method (Satopaa, Albrecht, Irwin, & Raghavan, 2011). The same is done for the non-ML approach, with the only difference of considering the cutoff yielding the maximum F_1 value. The resultant thresholds is then used to assess performances on the test set. Although

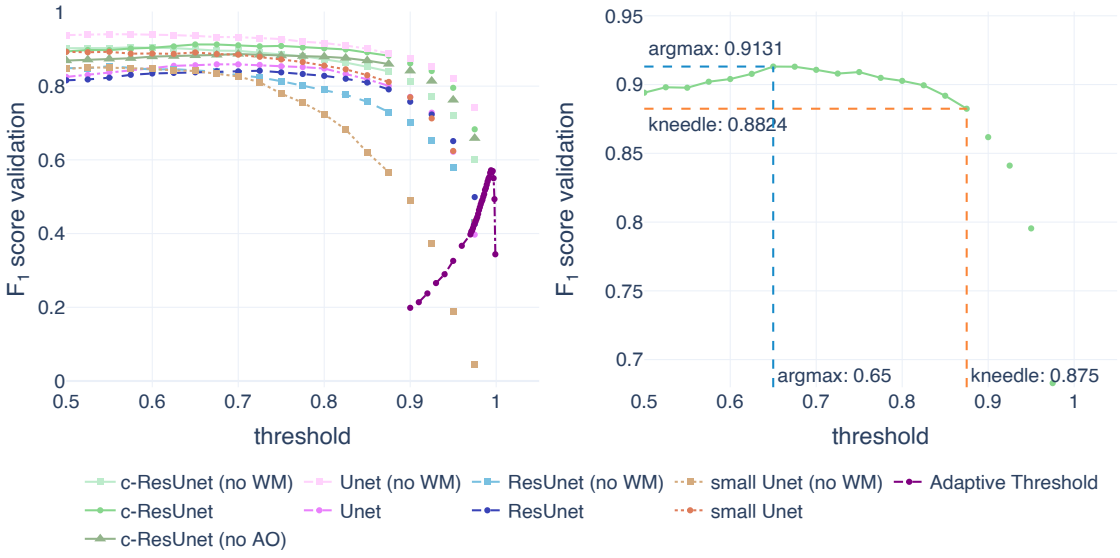


Figure 4.8: **Threshold optimization.** On the left, the F_1 score computed on validation images as a function of the cutoff for thresholding. On the right, the same is reported for the c-ResUnet to illustrate the selection of the best threshold for binarization according to *argmax* (blue) and *kneedle* (red) methods.

the ultimate goal is retrieving the counts, we rely on detection performance to enforce accurate recognition and avoid spurious balancing between false positives and false negatives – which are indistinguishable from the counts. Also, full-size images (as opposed to crops) are used to simulate better the model performance in a real-world scenario.

Figure 4.8 shows the optimization results. On the left, we can see how each model performance varies in the validation set as a function of the cutoff for binarization. For the adaptive thresholding approach, only very high thresholds lead to acceptable performances and we observe a sharp peak followed by a rapid decrease thereafter. On the contrary, all DL models work best for lower thresholds and present F_1 curves which 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 consider clearly stained cells. For these reasons, we opted for the *argmax* value (0.994) for the baseline approach, while we resorted to the *Kneedle* method (Satopaa et al., 2011) for the selection of the optimal DL threshold. An example of that choice in the case of c-ResUnet is reported in Fig. 4.8 (right).

Chapter 5

Results

After the training, the four competing architectures and the non-ML baseline are compared in three different scenarios: full design, weight maps only (no AO) and artifacts oversampling only (no WM). The 70 full-size images of the test set are used as a testbed. Table 5.1 reports individual model performances in terms of both detection and counting ability.

Model	Threshold	F_1	AUC	Accuracy	Precision	Recall	R^2	MAE	MedAE	MPE (%)
c-ResUnet	**0.875**	**0.8149**	**0.8705**	**0.6877**	0.9081	**0.7391**	**0.8215**	**3.0857**	**1.0**	-5.13
c-ResUnet (no AO)	0.875	0.8047	0.8741	0.6732	0.9019	0.7264	0.8077	3.0857	1.5	-6.24
c-ResUnet (no WM)	0.875	0.7613	0.8594	0.6147	0.9418	0.6389	0.7048	3.6857	**1.0**	-19.14
ResUnet	0.850	0.7855	0.8579	0.6468	0.8865	0.7052	0.7831	3.3286	**1.0**	**-4.84**
ResUnet (no WM)	0.850	0.7513	0.8643	0.6016	0.9387	0.6262	0.6955	4.0571	2.0	-24.12
Unet	0.875	0.7724	0.8609	0.6291	0.9117	0.6700	0.7560	3.5143	1.5	-14.36
Unet (no WM)	0.850	0.7886	0.8461	0.6510	0.8989	0.7024	0.8069	3.1571	2.0	-9.23
small Unet	0.875	0.7563	0.8691	0.6081	0.9264	0.6389	0.7682	3.5714	2.0	-21.37
small Unet (no WM)	0.825	0.6697	0.8326	0.5034	**0.9483**	0.5176	0.5723	4.7714	2.0	-32.01
Adaptive Threshold	0.994	0.6106	0.0865	0.4394	0.5680	0.6601	0.3565	8.0143	6.0	78.26

Table 5.1: **Performance metrics.** Test set performance using the optimal *kneed* threshold. The first five columns report the detection metrics, while the latter ones evaluate counting performance.

5.1 Performance

As a first observation, it is soon apparent how deep learning approaches perform better than adaptive thresholding according to all the considered metrics.

Starting with detection performance, c-ResUnet clearly outperforms all competitors in terms of F_1 score. Remarkably, the Unet is consistently worse than

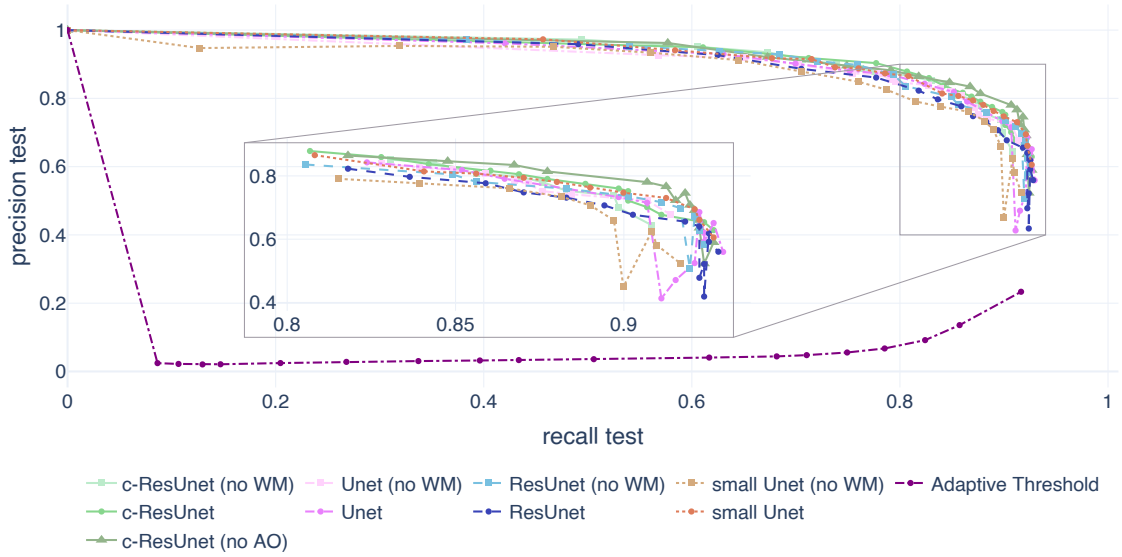


Figure 5.1: **Precision/Recall plot.** Test set precision/recall curves varying the threshold for predicted heatmap binarization. The inset plot reports a zoom of the top right corner to highlight differences of the various curves.

c-ResUnet and ResUnet despite having far more parameters (nearly 14M against 1.7M and 887k, respectively). The advantage of the ResUnet architectures is even more evident with respect to the lighter Unet version which has a comparable number of parameters (876k). The values of the area under the precision/recall curves also confirm this supremacy. While the F_1 can be interpreted as a measure of maximum performance achieved, the AUC can be even more powerful since it indicates a global measure of performance independent of the choice of the threshold for binarization (see Fig. 5.1). Again, c-ResUnet sits on top of the list, with ResUnet and Unet following shortly after. Interestingly, the small Unet performs on par according to the AUC metrics, suggesting that the choice of the threshold is perhaps suboptimal for the test set.

In addition, c-ResUnet keeps its leading role also when extending the evaluation to the other metrics. The only meaningful exception is precision, for which the Unet architectures are better. This is probably due to a tendency to overdetection. Nonetheless, the ResUnet counterparts well balance this behaviour with a significant improvement in accuracy and recall.

c-ResUnet remains the most accurate model also when shifting the focus on counting performance. Although the difference is hardly noticeable concerning the MAE, the gap becomes more prominent when looking at the R^2, with the c-ResUnet reaching a decent value of roughly 82% of explained variability. Interestingly, this time the ranking between ResUnet and Unet is inverted, with the

latter performing slightly better in terms of counting.

Finally, it is worth noticing that adopting the kneed optimal threshold ensures large cutoffs and enforces only detections with high confidence. Although desired, this behavior also increases false negatives as less cells are detected. As a result, we observe a drop in the accuracy whereby the impact of false negatives is twice as much the one in the F_1 score (cf. Eqs. (4.2) and (4.5)), thus explaining the gap between these two metrics. In conclusion, the model provides reliable predictions and satisfies the design requirement of being conservative with counts, as suggested by the negative values of MPE for all experimental conditions.

5.2 Design evaluation

This section presents the comparison of the different design choices investigated through the ablation studies. In order to evaluate the impact of artifacts oversampling and weight maps, the experiments were repeated under the same conditions described in Section 4.4, alternately switching off one of the two design choices.

From Table 5.1 it is evident how penalizing errors in crowded areas generally has a positive impact. Indeed, experiments exploiting weight maps achieve consistently better results than those without this addition (no WM). The only exceptions are the Unet and ResUnet architectures – the former in terms of F_1 score, MAE and R^2, and the latter concerning AUC. In particular, this strategy seems to produce a

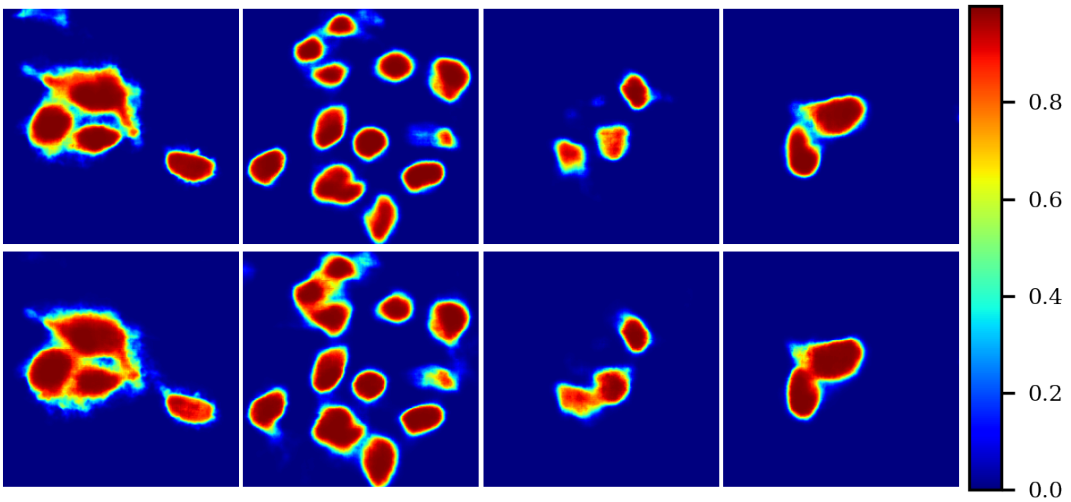
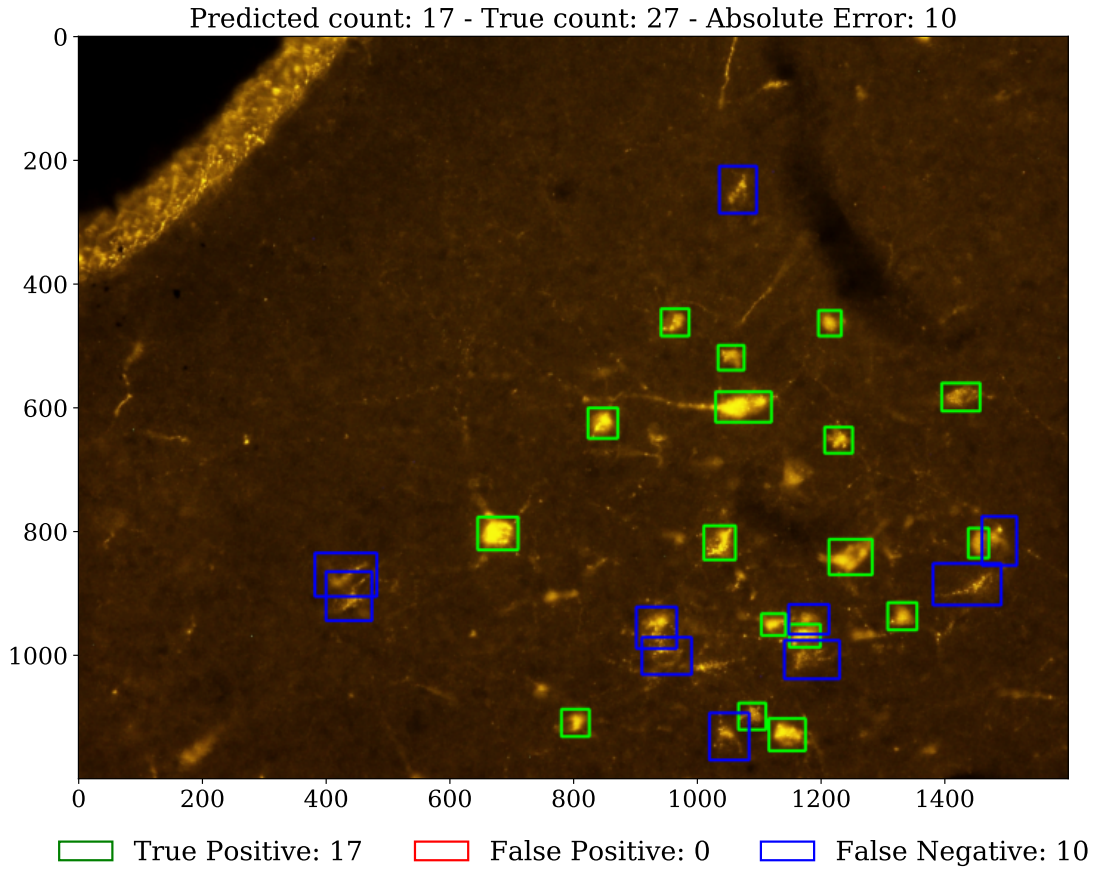


Figure 5.2: **Weight map effect.** Predicted heatmaps obtained with c-ResUnet (top row) and c-ResUnet (no WM).

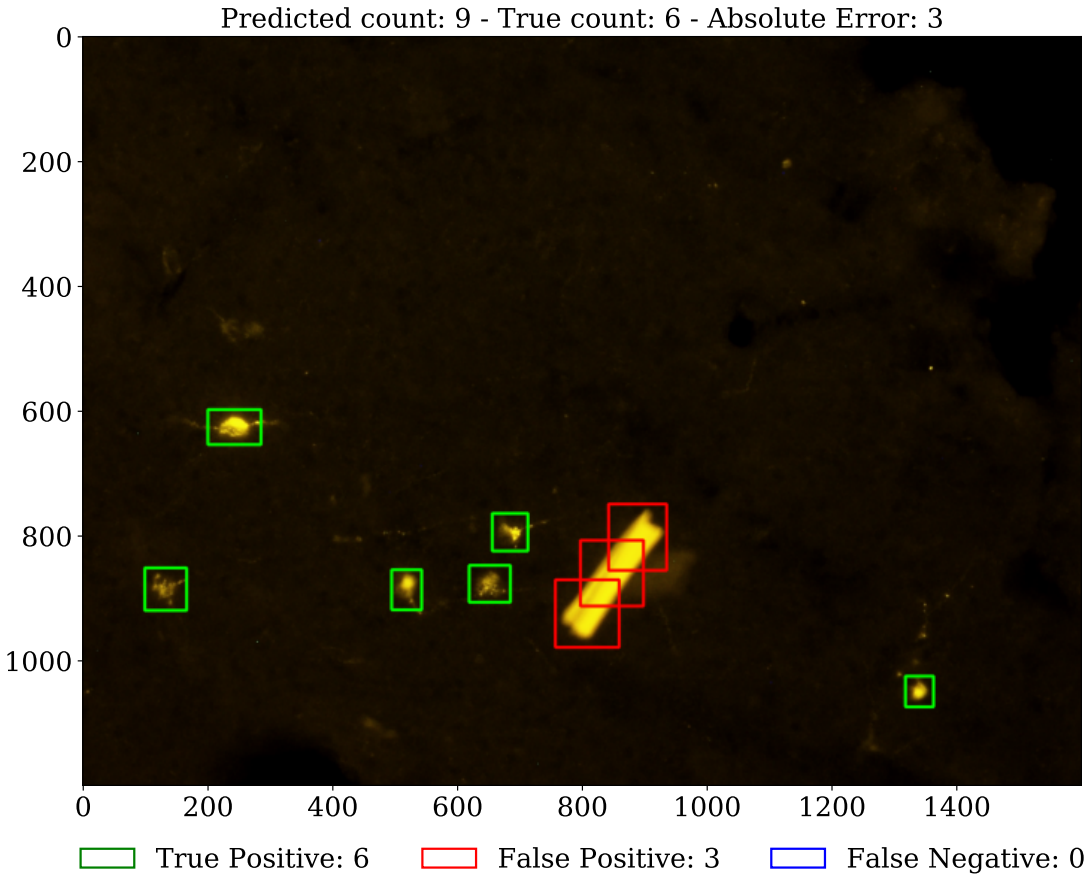
loss in precision to foster a more significant gain in accuracy and recall. Figure 5.2 illustrates a visual comparison of the c-ResUnet output in crowded areas with (top) and without (bottom) weight maps. Again, its beneficial contribution is apparent, with close-by cells sharply separated when exploiting the weight maps.

Regarding the impact of artifacts augmentation, Table 5.1 shows how there is little difference between the full c-ResUnet and the one without oversampling of challenging examples (no AO). In particular, the advantage of artifacts oversampling is numerically minimal. This is also confirmed by qualitative evaluation (Fig. 5.3). On the one hand, the c-ResUnet (no AO) avoids detecting more evident biological artifacts as the stripe in Fig. 5.3a even without specific oversampling. On the other, the c-ResUnet still fails to ignore the macaroni-shaped accumulation of fluorophore (Fig. 5.3b) although additional challenging examples are provided during training. Probably, this is due to the lack of similar structures in the training set, which makes the oversampling ineffective for such kind of artifacts. For this reason, the experiment was not replicated for the other architectures.



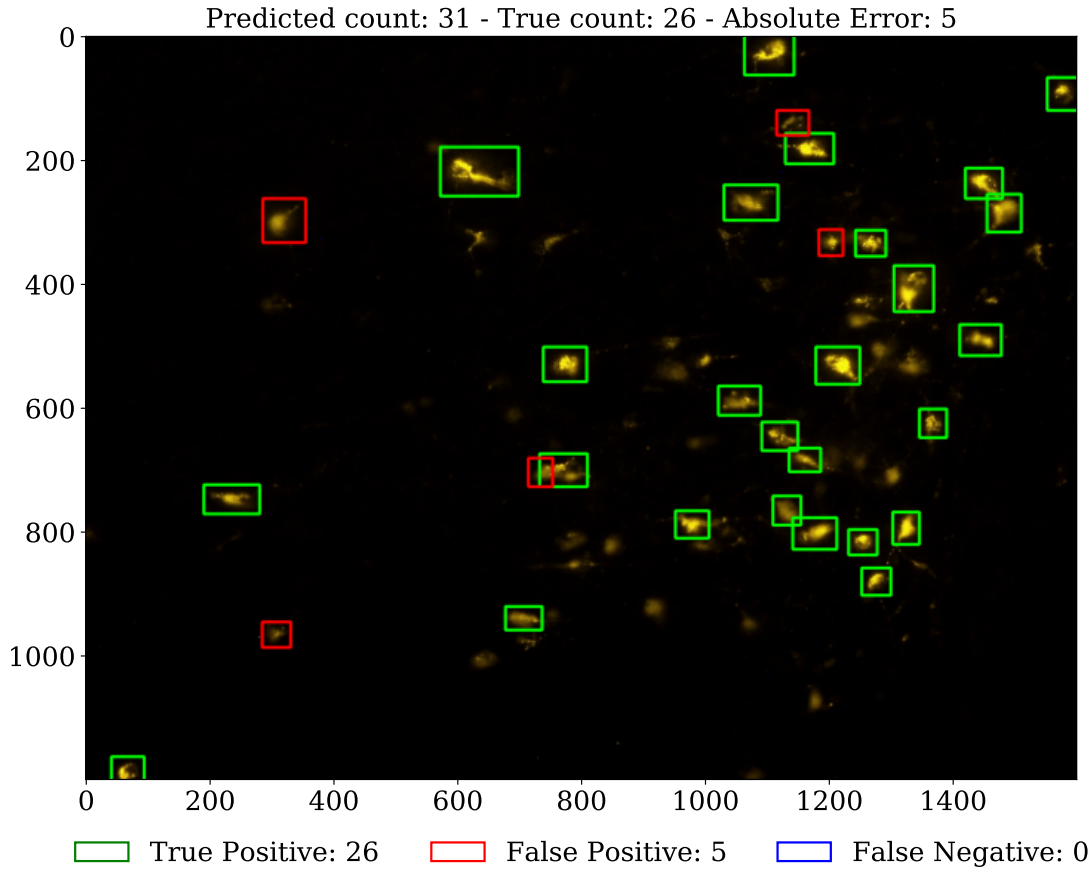
(a) stripes and filaments

Figure 5.3: **Results on test images.** The c-ResUnet (no AO) correctly handles the evident stripe in the top left corner despite not receiving dedicated oversampling for such biological structures.



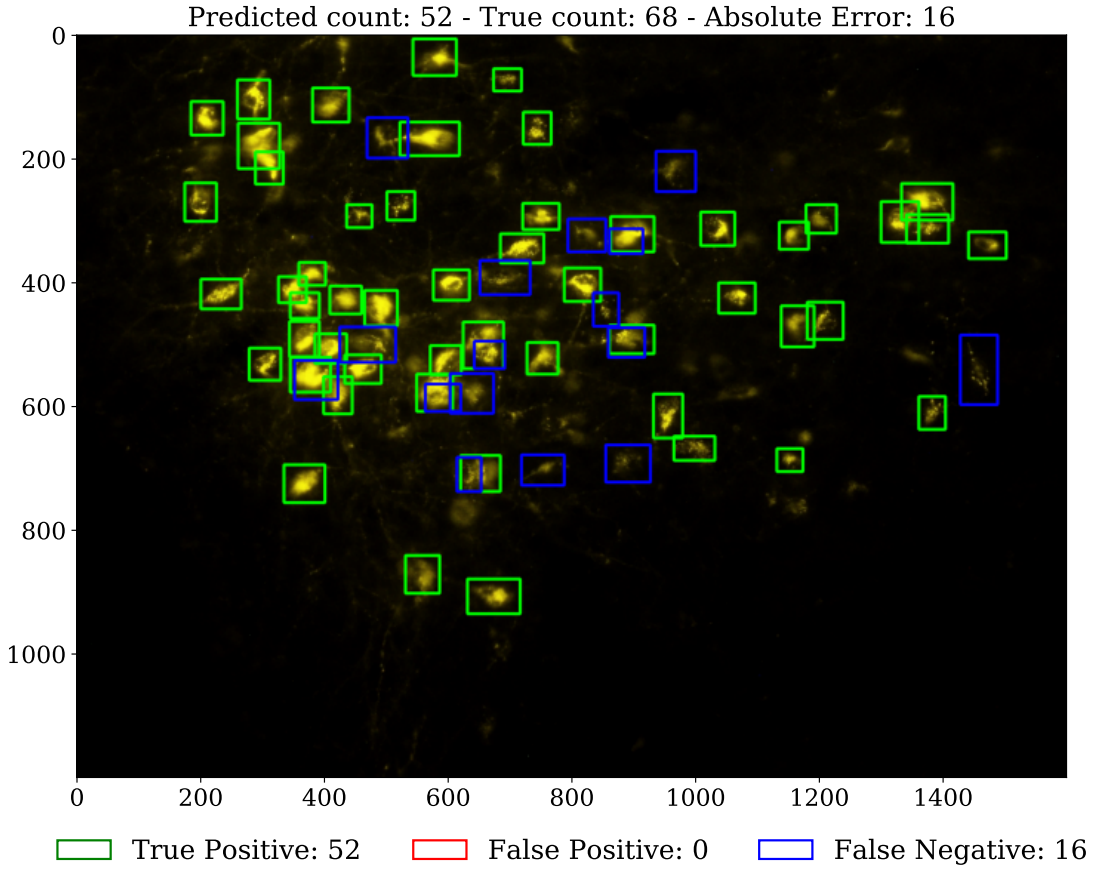
(b) technical artifacts

Figure 5.3: **Results on test images (2).** The c-ResUnet fails with the macaroni-shaped artifact in the middle of the picture, suggesting that the oversampling strategy is not effective in this case. However, notice that no other similar artifacts are present in the training set.



(c) false positives

Figure 5.3: **Results on test images (3).** The c-ResUnet sometimes produces false positives (red boxes), i.e. it labels as cell structures that are not annotated by the researcher. However, the difference with marked cells is marginal (cf. also with d), suggesting that these errors may lie within the limits of arbitrariness intrinsic to the task.



(d) false negatives

Figure 5.3: **Results on test images (4).** The c-ResUnet is generally conservative in predictions, thus generating false positives (blue boxes). However, these are similar to other stains that were not annotated (cf. also with c), thus falling again within the limits of operator's interpretation.

Chapter 6

Conclusions

In Part I, we tackled the issue of automating counting cells in fluorescent microscopy images through the adoption of Deep Learning techniques.

From the comparison of four alternative CNN architectures, the cell ResUnet (c-ResUnet) emerges as the best model amongst the investigated competitors. Remarkably, the careful additions with respect to the ResUnet (Z. Zhang et al., 2018) – i.e. a learned colorspace transformation and a residual block with 5×5 filters– enable the model to perform better than the original Unet (Ronneberger et al., 2015) despite having seven times fewer parameters.

Also, the two design choices considered in the ablation studies provide an additional boost in model performance. On one side, the adoption of a weight map that penalizes errors on cell boundaries and crowded areas is definitely helpful to promote accurate segmentation and dividing close-by objects. On the other, the effect of artifacts oversampling is less evident. Nonetheless, the combined impact of the two components guarantees better results than any of the two considered separately.

In terms of overall performance, the results are satisfactory. Indeed, the model predicts very accurate counts ($R^2 = 0.8215$ and MAE = 3.0857) and satisfies the conservative counting requirement, as testified by the negative MPE (-5.13%). The detection performance is also very good (F_1 score = 0.8149 and AUC = 0.8705), certifying that the precise counts come from accurate object detection rather than a balancing effect between false positives and false negatives.

Finally, qualitative assessment by domain experts corroborates further the previous statements. Indeed, by visually inspecting the predictions is possible to appreciate how even erroneous detections are somewhat arguable and lay within the subtle limits of subjective interpretability of borderline cases (see Figs. 5.3c and 5.3d).

In conclusion, the proposed approach proved to be a solid candidate for au-

tomating current operations in many use cases related to life science research. Thus, this strategy may bring crucial advantages in terms of speeding up studies and reducing operator bias both within and between experiments. For this reason, by releasing the c-ResUnet model[1] and the annotated data[2], we hope to foster applications in microscopic fluorescence and similar fields, alongside innovative research in Deep Learning methods.

6.1 Future work

As mentioned in Section 2.1, imaging techniques – and in particular fluorescence microscopy – are fundamental building blocks regarding many research activities in the life sciences domain. For this reason, we believe the field may highly benefit from the adoption of the available solutions presented in the literature concerning Deep Learning. At the same time, the distinctive traits of the images studied in life sciences – such as specific luminance conditions, biological and technical artifacts and cell overcrowding – may pose stimulating challenges for deep learning researchers. As a result, similar applications to the one described in Part I embrace potentially compelling problems for both communities. Therefore, the rest of this section is dedicated to outlining possible follow-ups we envision for the presented work.

A peculiar characteristic of microscopic fluorescence lies in the opportunistic association between the marker and the fluorophore. Although the choice of the former depends on the experiment's goal – since the marker must be compatible with the biological structures of interest –, the latter benefits from more freedom instead. In particular, its choice is not bound by biological properties, meaning that different fluorophores may be associated with the same marker. As a result, different studies may produce pictures of the same structures stained with different colors, and automatic detection approaches ought to recognize objects based on morphology rather than color. Clearly, this poses an interesting research line aimed at investigating the invariance of different techniques to the hue information. In this respect, a simple angle to address this requirement may be to start with grayscale images or incorporate hue-shift transformations in the augmentation pipeline.

Another interesting problem is the comparison of different strategies for precise localization. In fact, a limitation of our approach is the likely need for application-specific tuning of the post-processing parameters, especially for the watershed. This, of course, hinders a straight transposition of the method to new use cases

[1] available at: https://l.infn.it/linkmodel
[2] available at: http://amsacta.unibo.it/6706/

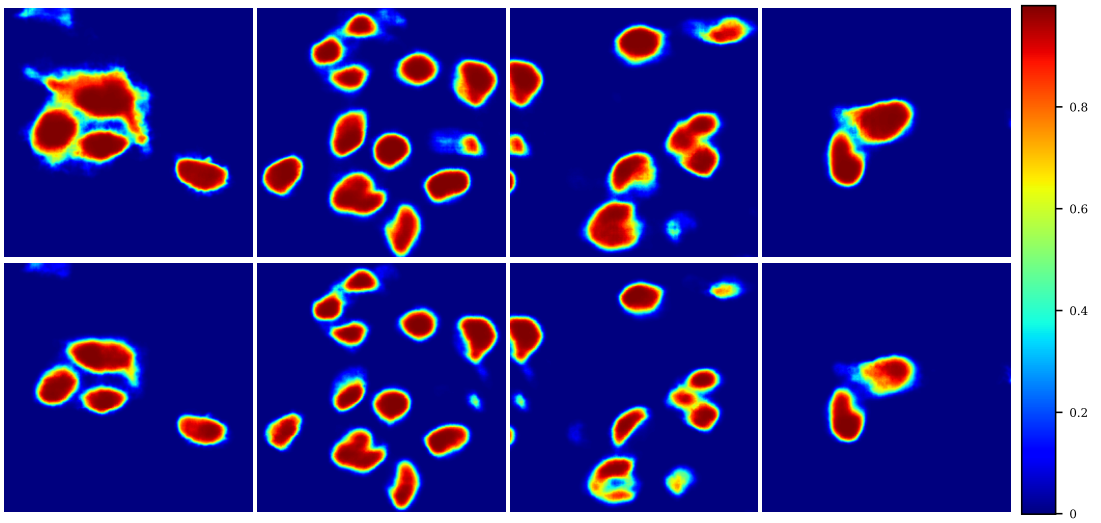


Figure 6.1: **Weight map effect.** Predicted heatmaps obtained with c-ResUnet using `keras` (top) and `fastai` (bottom) default initializations of the `BatchNorm` layers.

and paves the way for the exploration of more or less substantial variants. A possible approach would be to experiment with alternative post-processing strategies, perhaps including their parameters in the learning process to automate their optimization (see for example Wolf, Schott, Kothe, & Hamprecht, 2017). Furthermore, a possible strategy to tackle this issue is to tweak the model to enforce more refined localization in the first place. An attempt in this direction is currently under investigation by the authors of this work. In particular, preliminary results have shown that changing from the `keras` initialization of the batch normalization layers – `momentum=0.01, eps=0.001` – to that adopted by `fastai` (Howard & Gugger, 2020) – `momentum=0.1, eps=1e-5` – seem to produce predictions with sharper boundaries (cf. top and bottom rows in Fig. 6.1).

Apart from variants of the study discussed in Part I of this thesis, a whole new set of experiments may be devoted to analyzing different data from those contained in the current version of the Fluorescent Neuronal Cells dataset. In particular, we are currently studying the release of an extended dataset where more pictures are available. These features two additional target biological structures, and the images are reported both in single exposure and with more markers visible at the same time. This opens to novel learning tasks such as non-binary classification (one signal class per image, but more than two classes in total), multi-label classification (double marked pictures, more than one signal class in the same image) and transfer learning of the pre-trained architecture to

the additional biological structures. Also, another fascinating research direction would be to consider new data referring to different markers as a natural domain shift. Given this framework, it would be interesting to explore the extension of our approach in the context of active learning, thus aiming to enhance the model to account for additional biological structures without forgetting previous ones. In this regard, a technical matter related to the practical implementation of the latter use case is the adoption of helpful Machine Learning Operations (MLOps) tools. For example, this work combined `SuperAnnotate`[3] (commercial) and `Label Studio` (open source) (Tkachenko, Malyuk, Shevchenko, Holmanyuk, & Liubimov, 2020-2021) to produce the visualizations presented in Fig. 3.7. Both frameworks are designed for easing MLOps pipelines and let the researchers focus on the models and the learning strategies. The nice idea behind these tools is the concept of *human in the loop*. Indeed, apart from easing the annotation of new data, the user can leverage such solutions to keep track of their experiments and smoothly navigate through the typical stages of practical applications, i.e. the *training* $\longrightarrow$ *visualization* $\longrightarrow$ *labels refinement* $\longrightarrow$ *re-training* loop. Also, it would be nice to integrate our codebase within powerful open source solutions for continual learning, such as the recently released `Avalanche` framework (Lomonaco et al., 2021).

Despite the vast availability of fluorescence microscopy pictures, a significant bottleneck for supervised approaches is the lack of annotations. For this reason, a vital task would be to come up with ingenious strategies to leverage this massive amount of dispersed information without direct annotations in a *Self Supervised Learning* (SSL) framework. Figure 6.2 reports the draft of an ongoing attempt we are testing to pursue the above goal. In particular, we are trying to exploit the extended Fluorescent Neuronal Cells dataset to improve the encoding branch of our c-ResUnet and get rid of the weight maps. The currently studied strategy is based on a first pre-training of the c-ResUnet from scratch for marker classification. Specifically, the images are randomly sampled from three datasets depicting different biological structures present on the same tissues, i.e. obtained by injecting various marker-fluorophore couples into the same tissues and acquiring multiple pictures with the appropriate wavelength filtering for the corresponding marker. Then, the single-marked images are fed grayscale into the network to predict the corresponding marker label, used as a proxy for teaching the model to discriminate images with different structures (*pretext task*). This is achieved quite easily with human-like performance. Once that is accomplished, the same decoding path described in Fig. 4.4 is attached to the end of the pre-trained encoding branch, and the model is further trained for the downstream segmentation task presented in Part I. However, the results are not yet mature and were therefore excluded

[3]available at: `https://www.superannotate.com/`

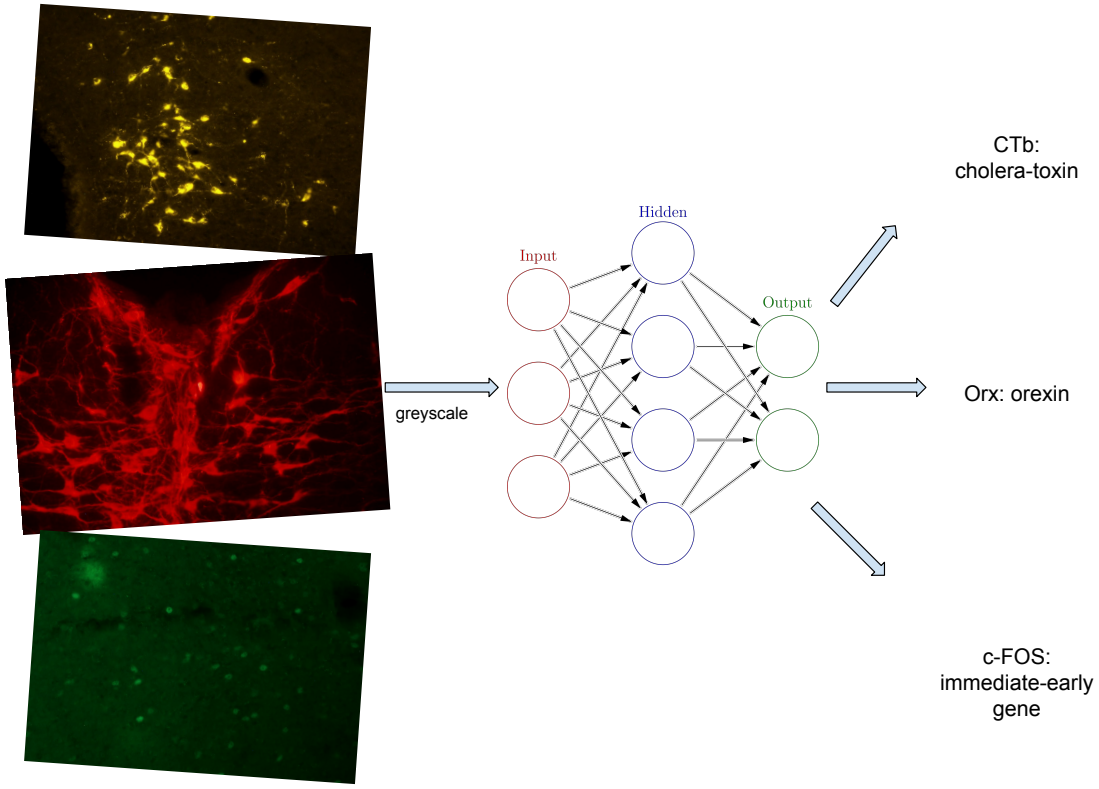


Figure 6.2: **SSL pretext task (pre-training).** A self-supervised strategy to exploit the extended Fluorescent Neuronal Cells data. The c-ResUnet backbone is first pre-trained using marker classification as a pretext task. Then, the resulting hidden representation can be leveraged for learning a downstream task of interest more easily. The network illustration is borrowed from Wikipedia.

from this dissertation.

Finally, another engaging project from the data science perspective would be deploying our model in production as a service for a wider community. At the moment, this task is subject to a feasibility study and the collection of requirements is being conducted. These involve the development of reliable service with suitable storage and computing resources, proper access mechanism (hosting and authentication), plus efficient scalability to a reasonable number of users.

Part II

Operational Intelligence for Distributed Data Management Monitoring

Chapter 7

Introduction

In the last twenty years, we have witnessed an unprecedented and ever-increasing trend in data production. Hilbert and López (2011) date the rise of this phenomenon back to 2002, with the beginning of the digital age. Indeed, the transition from analog to digital storage devices enormously expanded the capacity of accumulating data, thus leading to the *Big Data* era.

The term "big data" was first introduced in 1990s (Mashey, 1998; Lohr, 2013) and it is commonly adopted to describe datasets whose size exceeds the potential to manipulate and analyze them within reasonable time limits (Snijders, Matzat, & Reips, 2012). However, the expression does not target any specific storage size but rather assumes a deeper meaning that goes well beyond the sheer amount of data points. In fact, big data embrace a broad spectrum of data sources including structured, semi-structured and, mostly, unstructured data (Dedić & Stanier, 2016). Although multiple connotations have been attributed to the concept of big data over the years, a commonly shared definition is related to the so-called *5 Vs* (Jain, 2016):

- **Volume**: the actual quantity of generated data is huge, in the order of magnitude of terabytes and petabytes (Sagiroglu & Sinanc, 2013). More generally, it indicates amounts that are too large and complex to exploit conventional data storage and processing technologies;

- **Variety**: the data may come in several data types and from diverse origins. These include sources as sensors, social media, log files and more, plus they encompass heterogeneous formats like text, images, audio, video and so on;

- **Velocity**: data are produced and/or processed at high rates (Kitchin & McArdle, 2016), typically nearly real-time;

- **Value**: data must carry valuable information that, if correctly analyzed, bring business value and profitable insights (Uddin, Gupta, et al., 2014). In a scientific context, this means information that contribute to the advancement of human knowledge;
- **Veracity**: data sources must be reliable and generate high-quality data that can produce value (Onay & Öztürk, 2018; *Big Data's fourth V*, 2017);

Nonetheless, the community has not reached a complete agreement on the big data definition (Grimes, 2013; Kitchin & McArdle, 2016), with some authors suggesting moving their characterization from the intrinsic properties to the techniques adopted to acquire, store, share and analyze the data (Balazka & Rodighiero, 2020).

Besides the modification of the storage supply, multiple factors significantly enhanced data production and, hence, favored the rise of the big data era. In the first place, the diffusion of the internet and the progress of computer technologies provided more processing capabilities and easier access to data, thus stimulating further their production. Consequently, several stakeholders as big tech companies, traditional industries, governments, healthcare institutions and more started increasingly contributing to this growth. Finally, the introduction of *smart* everyday objects that not only receive but also produce data exponentially accentuated individual contributions to the total data produced. Modern objects, in fact, are endowed with technologies that allow to collect data and share them via a network – the so-called Internet of Things (Ashton et al., 2009) –, thus augmenting the production rate even more. For instance, sensors measuring the status and operation are now commonly used in industrial machinery and household appliances to ease their control and automate maintenance. The same paradigm is also influencing the direction of the personal items market in various ways. For example, some tech companies are recently investing in wearable devices like watches and glasses to enable the users to be always connected with a rapidly mutable environment, track their progress and explore the world in unparalleled manners thanks to virtual reality. Furthermore, the solutions that digitization offers are being explored to respond to the emerging challenges of current times. Think, for instance, of the urge for modernization of institutional processes posed by the pandemic. The massive spread of the infections has required unprecedented amounts of patients needing access to health assistance. However, the impossibility to scale up services and equipment correspondingly caused huge issues and jeopardized people's safety. In such context, the availability of intelligent systems capable of remotely monitoring patients' conditions and providing them with specialist support would have enormously helped.

In order to cope with the growing amount of data to store and process, the big data players of both industry and academy have gradually moved to new

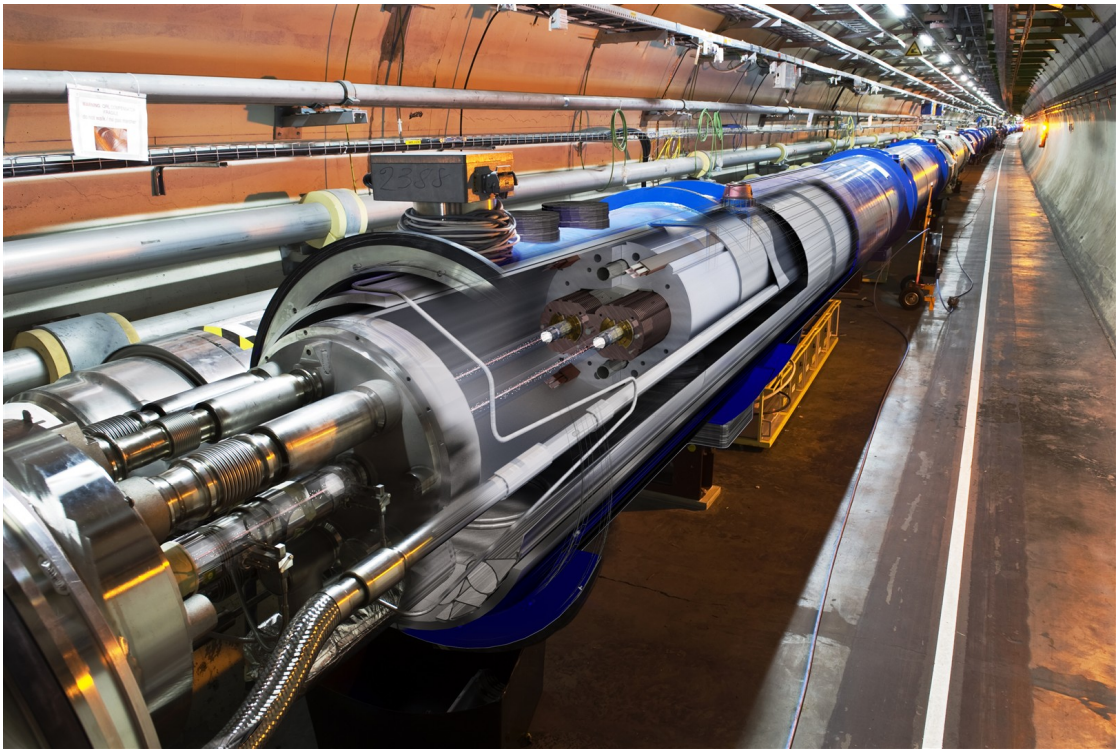
computing paradigms in recent years. For instance, new solutions as **distributed** and **cloud computing** (Kshemkalyani & Singhal, 2011; Wang et al., 2010) have been specifically designed to address these new requirements, taking advantage of multiple and heterogeneous resources geographically displaced and accessible via a network.

However, the boost in performance guaranteed by these technologies comes with the price of requiring very complex interactions of both hardware and software components. Aside from the enormous benefits these solutions bring, their most relevant drawback is that the wider the infrastructure, the higher the chances of something going wrong, and the bigger the effort to detect, inspect and solve the issues. Part II explores this domain and tries to propose a data-driven pipeline to ease and support people working to maintain the infrastructure integrity. Despite applying to many different applications with some tuning, the presented approach is discussed in the context of **data transfer failures** within the *Worldwide Large Hadron Collider Computing Grid* (WLCG).

7.1 The HEP community and LHC

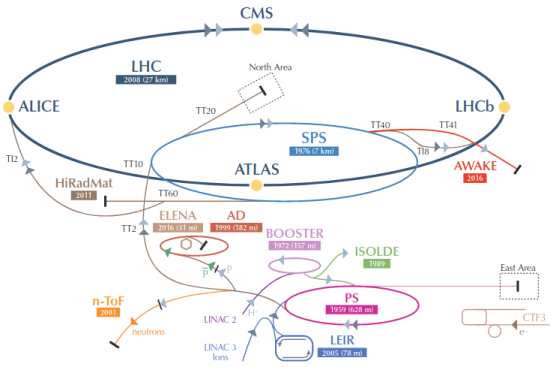
High-Energy Physics (HEP) is a branch of physics that studies the elementary constituents of matter and the fundamental principles that govern their interaction to understand how our universe has formed and is evolving. These particles, however, are not visible at the scales whereby we experience reality today. Thus, HEP experiments need to either look at natural phenomena generated in pressure and temperature conditions similar to those of the primordial universe – like cosmic rays – or recreate such settings artificially.

The European Council for Nuclear Research (CERN) is part of this second strand of experiments, and it constitutes the largest particle physics laboratory in the world. From 2008, CERN facilities also include the Large Hadron Collider (LHC), the longest particle accelerator ever built. LHC consists of a 26.7-kilometer ring located in a tunnel about 100 meters underground in the Geneva area (Fig. 7.1), and it is made of superconducting magnets with several accelerating structures (CERN, n.d.). Inside the accelerator, bunches of protons are revved up to nearly the speed of light, forming two high-energy particle beams that travel in opposite directions inside separated pipes. When they acquire the desired energy, the beams are directed towards dedicated interaction points where the experiments occur. In practice, LHC hosts four major experiments built in correspondence of these locations – ALICE (ALICE Collaboration, 2008), ATLAS (ATLAS Collaboration, 2008), CMS (CMS Collaboration, 2008) and LHCb (LHCb Collaboration, 2008) – and equipped with giant detectors (Fig. 7.2). Once the beams get there, the two pipes cross and the particles are squeezed through substantial magnetic fields



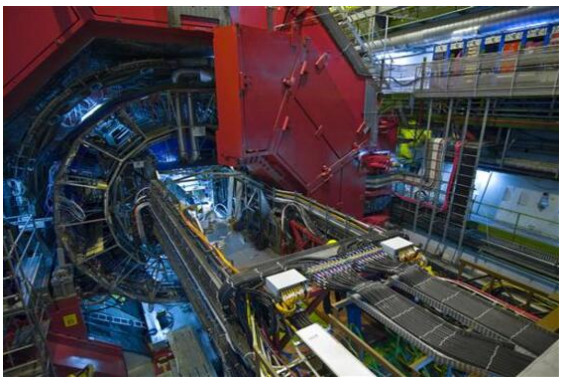
(a) LHC accelerator

(b) Aerial view

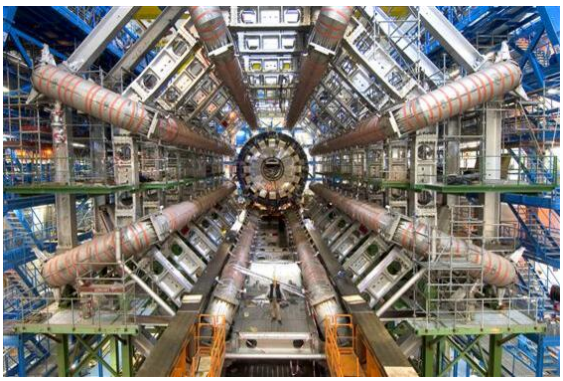


(c) LHC scheme

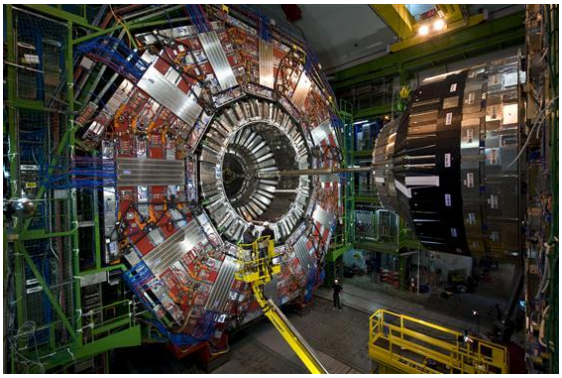
Figure 7.1: **LHC accelerator complex.** (a) the underground tunnel that hosts LHC and its transverse section; (b) aerial view of the LHC complex at the boundary between Switzerland and France; (c) schema of the various LHC accelerating structures. The pictures are borrowed from various online sources: (1), (2), (3).



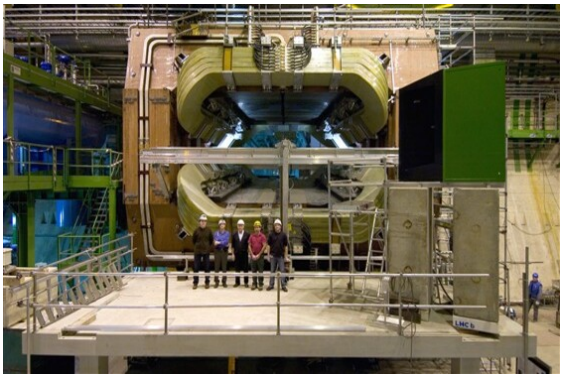
(a) ALICE



(b) ATLAS



(c) CMS



(d) LHCb

Figure 7.2: **CERN major experiments.** The images are borrowed from the CERN experiments image gallery.

to increase their chances of colliding. In this way, a massive amount of energy is concentrated in an extremely tiny area, generating billions of particles at each collision. Indeed, the high intensity of the beams causes roughly 40 million crossings per second at each interaction point (Albrecht et al., 2019; Grandi, 2017). When a crossing happens, an average of 60 bunch collisions – also referred to as pileup – are observed (Albrecht et al., 2019). The particles produced by each scattering then fly around the interaction point to be eventually detected through high-technology experimental devices endowed with over 100 million electronic channels (Grandi, 2017; Aad et al., 2020). According to the latest experimental setup, this delivers 100 MegaBytes (MB) of data per collision and it would generate 40k ExaBytes (EB) every year (Grandi, 2017). However, storing such a tremendous amount of data is unattainable with current technology and budget. In addition, the events of interest are typically rare, so there is actually no need to record all of the information detected by the electronic channels. Thus, the vast majority of *read-out* data from collisions is discarded straight away using hardware and software trigger selection systems, thus lowering the *recorded* event rate to 1k crossings per second. As a result of this reduction, the actual acquisition rate amounts to nearly 1 PetaBytes (PB) per day (CERN, 2017), translating to roughly 160 PB[1] a year in 2018. Besides that, physics analyses require comparing experimental results with Monte Carlo data simulated according to current theories, thus producing somewhat between 1 and 2 times additional data (Grandi, 2017). Furthermore, the CERN community is already working at enhancing the Large Hadron Collider capabilities. The project involves boosting the energy of the beam and gradually increasing the pileup towards 200 collisions per bunch crossing (Albrecht et al., 2019), thus leading to the so-called High Luminosity LHC (HL-LHC) (Aberle et al., 2020). Thanks to this upgrade, the observed events are expected to increase of a factor ≥ 5 (Aberle et al., 2020) and produce an estimated 800 PB of new data each year by 2026.

Although it is difficult to replicate such a punctual measurement of the data production for other big data players, some hints can be retrieved by comparing multiple online resources (Clissa, 2022). Figure 7.3 tries to summarize a reasonable, up-to-date "guesstimate"[2] of yearly data production for the main big data companies. Despite not being the most popular among the mainstream audience, the HEP community is one of the most prominent players concerning big data.

[1] LHC registered 161 days of physics data taking in 2018 (Todd, Ponce, Apollonio, & Walsh, 2018)

[2] These data are reconstructed based on multiple online sources about the amount of contents produced, streamed or hosted by big data companies and reasonable estimates of unitary sizes for such contents, e.g. average mail or picture size, average data traffic for 1 hour video, and so on. However, the actual values reported are not meant to be extremely accurate and only serve the purpose of giving an idea of the orders of magnitude of the various phenomena.

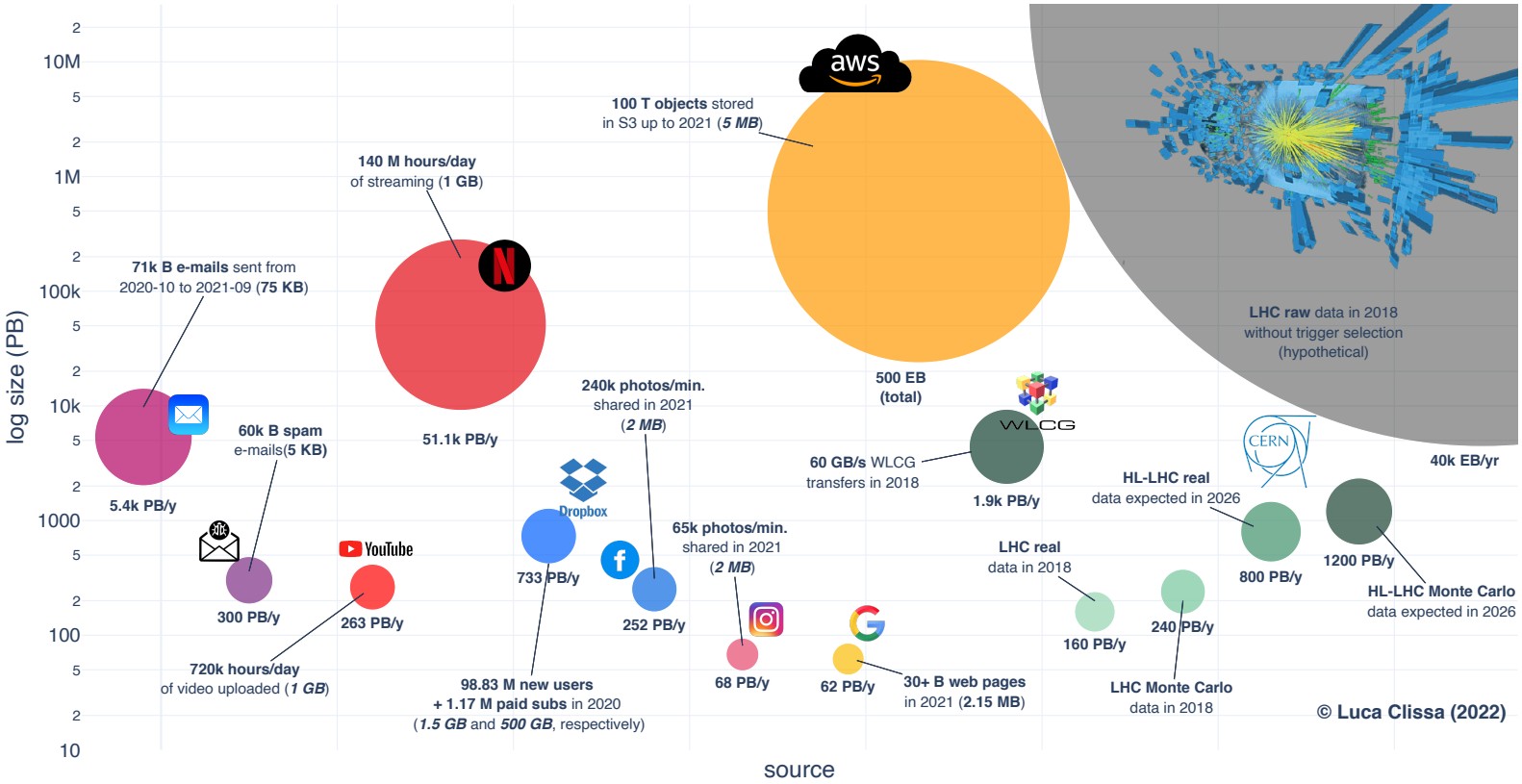


Figure 7.3: **Big Data sizes.** Bubble plot of the orders of magnitude of data produced by important big data players. The balloon areas illustrate the amount of data and the text annotations highlight the key factors considered in the estimates. Average per-unit sizes are reported in parentheses, where italic indicates measures reconstructed based on likely assumptions because no references were found. Interactive version available at: BigData2021.html.

Indeed the read-out data LHC produced every year in Phase 2 (40k EB) is around one order of magnitude bigger than the total size of objects ever stored on Amazon AWS cloud service (500 EB)[3]. Considering effectively recorded data, LHC figures are comparable with those of other most renowned big data entities. The last run (2018), in fact, produced hundreds of PetaBytes (PB) (roughly 160 of real data and 240 of Monte Carlo simulations), which is similar to the orders of magnitude generated by Google searches (62 PB)[4], Instagram and Facebook shared photos (68 and 252 PB, respectively)[5] and YouTube video uploads (263 PB)[6]. Moreover, LHC will climb the table even further with the upgrade to high luminosity, when the real and Monte Carlo data production rate is expected to rise to levels comparable to those of storage services like Dropbox (800, 1200 and 768 PB[7], respectively).

Apart from the nominal values of the generated information, streaming data comprise a significant slice of the big data market. As a matter of fact, the continual movement of small- to medium-sized files spawns massive traffic when scaled up to millions of users, as testified by e-mails (5.7k PB)[8], and Netflix (51.1k PB)[9] bubbles in Fig. 7.3. A similar usage is generated also by the LHC, whose data are continuously transferred across the HEP community thanks to the Worldwide LHC Computing Grid (see Section 7.2) to fuel innovative research. For example, a throughput of 60 GB/s was generated by the 4 experiments together in 2018 (WLCG, 2019) thus giving a yearly projection (1.9k PB) close to half of the global e-mails traffic and only one order of magnitude lower than Netflix usage.

Given the large quantities of data involved by the LHC, it is not surprising how careful planning must be done in order to meet the needs of the LHC com-

[3]Obtained considering the total number of objects reportedly stored in Amazon S3 (100 trillion, Barr (2021)) and assuming an average size of 5 MB based on some average bucket example (Hampton, 2021)

[4]Obtained considering that Google search index contains at least 30 billion web pages (Van den Bosch, Bogers, & De Kunder, 2016; De Kunder, 2021; Djuraskovic, 2021; Indig, 2020) and that the average page size is 2.15 MB (Teague, Karamalegos, Rebecca, Peck, & Pollard, 2021)

[5]Obtained considering that 65k and 240k pictures are shared every minute on Instagram and Facebook (Domo, 2021), and assuming 2 MB as a reasonable average picture size (Adobe, 2021)

[6]Obtained considering that 720k hours of video are uploaded daily (Dean, 2021b) and assuming an average size of 1 GB (Vera, James, & Dan, 2019)

[7]Obtained considering that Dropbox registered 100 million of new users in 2020, 1.17 million of which were paid subscriptions (Dean, 2021a). For the average per-unit size, it was assumed that free accounts exploited 75% of the 2 GB storage available, while paid ones exploited 25% of the total 2 TB

[8]Obtained considering that 71k billion e-mails and 60k billion spam messages were sent from October 2020 to September 2021 (Statista, 2021), and that the average size is 75 KB for e-mails (Tschabitscher, 2021) and 5 KB for spam (Baker, 2014)

[9]Obtained considering that Netflix users consumed 140 million hours per day of streaming (Domo, 2021) and that counts for 1 GB of data for standard definition videos (Perry, 2021)

munity, and tailored strategies and technologies must be adopted to cope with such requirements. Luckily, the presence of other stakeholders facing analogous problems provides the HEP community with some alternatives to draw from, and it allows researchers to tap in from existing solutions and customize them for their necessities.

7.2 The Worldwide LHC Computing Grid

The Worldwide LHC Computing Grid (WLCG) (Bird, 2011) is a global collaboration that links up more than 170 computing centers in 42 countries, serving an audience of more than 12000 physicists all around the world. As of 2022, WLCG constitutes the largest computing grid in the world and it is supported by many associated national and international grids, such as the European Grid Initiative and the Open Science Grid, as well as many other regional grids. Founded in 2002 by CERN, the WLCG mission is to provide computing resources to store, distribute and analyze the data generated by the Large Hadron Collider.

Given the scale and complexity of the LHC data, this requires massive storage facilities, immense computing power, global networking, tailored software, adequate personpower and, of course, funding. In order to achieve such challenging goals, WLCG leverages a distributed computing paradigm, where resources are shared among member states and made equally available to all the partners, regardless of their physical location. Figure 7.4 summarizes the WLCG infastructure composition. The Worldwide Large Hadron Collider Grid is structured in 4 levels, called *tiers*, differing in terms of computing resources, storage capabilities and delivered services. Its bottom layers comprise a few computing centers having great amounts of storage and processing resources, ultra-fast network connectivity (up to 100 GB/s), and they are devoted to general processing tasks. The shallower layers, instead, group many smaller data centers devoted to more specialized activities. In particular, the CERN data center is located at the bottom of this infrastructure, constituting the cornerstone of the whole architecture. It is located in Geneve (Switzerland) and it is endowed with more than 73000 processor cores, providing around 20% of the total compute capacity of WLCG. In terms of activity, the Tier-0 is responsible for i) the management of the raw data streams coming from the LHC experiments and their archiving for safe-keeping, ii) the reconstruction of physical entities like particles energy and velocity starting from the raw read-outs recorded by the electronic equipment, and iii) the distribution of raw and reconstructed data to the next tier layers. Moving up the WLCG architecture, we find 13 large computer centres of the *Tier-1s*. These are directly linked to the Tier-0 and contribute to WLCG operations with sufficient storage capacity and round-the-clock support for the users. They are responsible for i) the

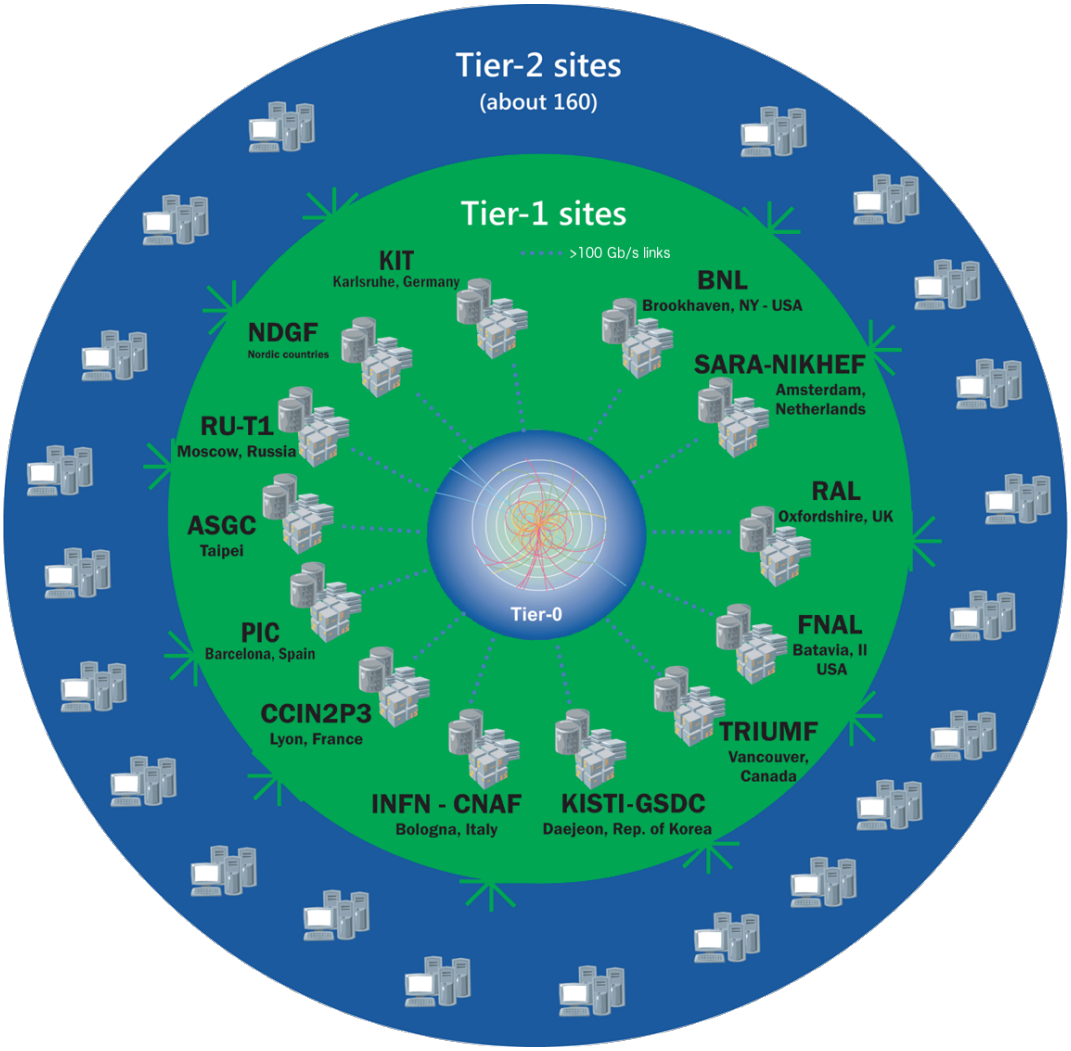


Figure 7.4: **WLCG structure.** The Worldwide Large Hadron Collider Grid has a tiered structure organized into three levels and comprising more than 170 computing centers spread across 42 countries. The image is borrowed from the WLCG website.

safe-keeping of a proportional share of raw and reconstructed data, ii) large-scale reprocessing and safe-keeping of corresponding output, iii) access and distribution of data to the next infrastructure levels, and iv) safe-keeping of Monte Carlo simulated data. One of these Tier-1 sites is located in Bologna[10] and it represents one of the biggest data centers in Italy. Its facilities count 40000 CPU cores, 40 PB of disk storage, 90 PB of tape storage, and the center is connected to the Italian (GARR) and European (GEANT) research network infrastructure with more than 200 Gbps (INFN-CNAF Collaboration, 2019; dell´Agnello, Luca et al., 2019). The subsequent layer involves around 160 *Tier-2* sites. These are data centers offered by universities and other scientific institutes, which are connected through regional networks. They essentially act as analysis facilities to perform specialized tasks as the experiments production jobs and the Monte Carlo simulated data, which are then either stored locally or shared across the infrastructure. Finally, the last rung of the ladder is constituted by *Tier-3s*. These sites enormously vary in scale, ranging from local computing resources like university clusters or even individual pc to large national analysis facilities. In fact, they do not formally belong to the WLCG but solely serve as entry points to its infrastructure for end-users analyses.

Alongside the hardware facilities, the WLCG supplies also advanced software solutions to provide researchers with seamless access to resources in a transparent way, without needing to worry about where the computing resources are coming from or where the data are physically stored. This means that users can request access to data or resources from one of the many entry points into the system, and the grid infrastructure will then take care of spawning all the needed processes under the hood. This may entail establishing the user identity and its access rights to the various sources, checking their credentials, and searching for available sites that can provide the requested resources.

[10]for more details: https://www.cnaf.infn.it/en/

Chapter 8

Operational Intelligence

The automation of infrastructure management and maintenance has become crucial in recent years. The increasingly large scale of modern data centers, and the adoption of distributed resources that necessitate the interaction of diverse hardware and software components, have made this task extremely complex. Consequently, traditional approaches to infrastructure management where manual human intervention is required have become impractical or even useless. For this reason, several communities involved in the Worldwide LHC Computing Grid have started a project named **Operational Intelligence**[1] that aims at increasing the level of automation in computing operations, thus reducing human interventions. As a result of the joint effort, several strategies have already been proposed to support operational workflows in various ways (Clissa, Lassnig, & Rinaldi, submitted; Di Girolamo, Alessandro et al., 2020; Di Girolamo et al., 2022; Leite, Decker, Santana, & Souza, 2020; Diotalevi et al., 2019). Although listing a precise taxonomy of alternative methods is hard – since the boundaries between different classes are often blurred and the categories may overlap – a first distinction can be established based on the analysis intent. A common approach is to focus on *anomaly detection*, where the objective is to spot anomalous behaviors that may entangle underpinning faults in the system. The detected anomalies are then reported to experienced operators for further investigations and fixes. However, sometimes the malfunctions are too many to be inspected and solved singularly. Hence, an option is to rely on *error categorization* to reduce the number of reports to check by grouping similar issues. This approach assumes that similar problems have similar solutions, therefore it is possible to address all the events of a group by inspecting only one (or a few) of them. A more desirable yet more complicated target is *root-cause analysis* (Solé, Muntés-Mulero, Rana, & Estrada, 2017). In

[1] for more details: https://operational-intelligence.web.cern.ch/

this case, the objective is to identify the origin of the problem directly, thus entirely automating the diagnosis phase. In turn, these approaches can be further split into methods that seek just root causes – what induced the issues – or plain explanations – why/how the faults were conceived.

Another essential distinction is based on the time of intervention with respect to a system failure. The simplest approach is *reactive maintenance*, where the human intervention occurs after a failure is detected. In this case, the fault diagnosis is performed *post-mortem* and, if effective, it helps identify the problems and speed up the restoration of good operating status. Nonetheless, the failures are not avoided and downtimes or denial of services are impossible to avert. Some approaches try to intervene proactively to overcome this limitation, performing the so-called *preventive maintenance*. In this case, the objective is to set an optimal schedule of periodic interventions to preserve high Quality of Service (QoS) and prevent faults directly. However, completely eradicating failures requires frequent maintenance that is often unnecessary, which increases management costs. An alternative strategy to limit these extra expenses is *predictive maintenance*. The idea is to monitor the infrastructure on the fly (*real-time* or *online* analysis) and try to predict when an intervention is required. In this way, the inconveniences deriving from system downtimes are limited, and the costs imputable to unnecessary hardware replacement are cut.

Apart from the end goal and the time requirements of a given use case, a further distinction can be established based on the analyzed information. Indeed, the choice of which strategy to pursue is bound by the available data or, vice-versa, it restricts the applicable techniques. A first family of approaches leverages overall workloads – e.g. number of running processes, hardware resources usage, network saturation – as indicators of infrastructure health and monitor their trends over time. The deviations from normal operations are considered anomalies and trigger alerts to be investigated by experts (Giordano, Domenico, Paltenghi, Matteo, Metaj, Stiven, & Dvorak, Antonin, 2021). A second class relies on *event logs* as the primary way to register key runtime information. These reports record events happening during the execution of a system to provide an audit trail that can be helpful to understand the system activity and diagnose problems. This information can be exploited in various forms. Some approaches focus on log activity summary statistics (e.g. number of printed lines) and try to disentangle nominal behaviors from suspect activity (Decker, Leite, Giommi, & Bonacorsi, 2020; Decker, Leite, Viola, & Bonacorsi, 2020; Minarini & Decker, 2020). Other alternatives use the log content instead, thus directly analyzing the textual information contained in the log files (Giommi et al., 2019). These vary from traditional keyword searches – e.g. “kill”, “error”, “fail”, “exception” – and heuristics (Tisbeni, 2019) to smarter tools based on deep learning language models. The advantage of such procedures is

that the textual information can aid system experts finding root causes and explanations which are harder to grasp from sheer workloads. Both families mentioned above can be adopted for both online monitoring and offline debugging, depending on the use case. Finally, another class of techniques focuses on **error messages** and is only devoted to diagnosing issues in post-mortem analyses. Like event logs, errors can be exploited to conduct thorough root-cause investigations or just error categorization. Although logs and error messages both collect textual data, their difference lies in the format this information is presented in and the actual content. Error messages only include system printouts related to failure events, while logs typically gather the full runtime information. In terms of format, error messages are cleaner compared to log files, despite both can be listed as semi-structured data. In fact, both classes can be summarized into a free-form constant string describing the status of a system, plus some parameters that record important system attributes.

This work tackles the problem of error categorization for post-mortem diagnosis of issues during WLCG data transfers.

8.1 Distributed data management

LHC data are arguably the most valuable asset of the HEP community. As a consequence, the data are continuously transferred across the grid for several purposes, and a paramount part of the WLCG operations involves Distributed Data Management (DDM) processes.

Indeed, stringent workflows are put in place by the experiments to ensure data distribution and redundancy, thus preventing data loss and guaranteeing reliable accessibility. For example, the ATLAS experiment has drawn up an accurate plan – the so-called *computing model* – describing in detail the data life cycle (ATLAS Collaboration, 2008; Calafiura, Catmore, Costanzo, & Di Girolamo, 2020; Bird, 2011). The first data stream happens at the CERN data center, where a combination of electronic and software triggers are applied to the raw data acquired by the detectors to filter out uninteresting collisions. The skimmed data are then archived in the Tier-0 on tape supports for long-term storage, and a second copy is sent to one of the Tier-1s through a dedicated network. After that, a first-pass reconstruction occurs to retrieve physically meaningful information – such as particles energy, velocity, scattering angles and so on – from the electronic signals recorded by the experimental devices. As for the raw data, these elaborations are stored in double copy, one at CERN and one at the same Tier-1 hosting the corresponding raw data. In this way, two full copies of the same raw and reconstructed data are retained to safeguard data accessibility and recovery. The copy at CERN is archived on durable but slow storage supports, and it serves the purpose of

restoring the data in case of losses or corruption. The second copy, instead, is stored on hard drives that are more prone to faults but guarantee a faster reading speed to comply with the repeated data accesses typical of analysis workflows. Once the data are properly distributed, a second stream takes place at the Tier-1s that provide data-intensive processing facilities for large-scale organized analysis. Here, further (re)processing and calibrations are performed on the reconstructed data, and the derived outputs are stored and shipped on-demand to other sites for subsequent elaborations. The last part of the data life cycle is then performed at the Tier-2s, where Monte Carlo data are simulated and sent back to Tier-1s for long-term storage. Furthermore, the Tier-2s are exploited by smaller groups of researchers to conduct more specialized analyses. In such cases, additional data streams are needed to retrieve the reconstructed data from the Tier-1s and make the results available to the end-users on their local machines.

As a result of DDM workflows, massive amounts of data are constantly moved across the grid. In order to achieve that, various services for file transfer have been developed. These are used alternately or concurrently to create a chain of software services that act as interfaces between the end-users and the physical resources. At the lowest level there is the File Tansfer System (FTS) (Karavakis et al., 2020), which is configured to reliably interact with diverse storage devices and filesystems, execute fault-tolerant transactions and support users authentication. On top of that, the various collaborations may add other middleware layers as higher-level interfaces for the users. For example, ATLAS uses an open-source framework called Rucio (Barisits et al., 2019), that basically orchestrates the transfers creating a catalog to track data locations, managing replication rules and retries in case of failures and so on. Clearly, ensuring high QoS is very hard due to the huge volumes transferred, the heterogeneity of the software and hardware components and the large user base.

8.2 FTS transfer failures

In practice, occasional faults may happen at various levels during data transfers, which may include a wide range of root causes, provoking failures during the shipment of the files. These errors may vary from naive ones – e.g. a mistyped command or the request of an unavailable file – to more severe software and hardware defects. For instance, the requesting endpoint or archiving server might be temporarily unreachable (connection shortage). Likewise, the requested data may be corrupted (checksum error) due to storage hardware faults or unstable connection (network problem). Also, there might be timeouts when the shipment takes more than the pre-configured waiting window – e.g. when the desired data are bigger than usual and/or must be retrieved from tape, thus requiring more

time. In addition, errors of different nature may often arise due to the interactions between miscellaneous middleware layers. All of these factors, and more, can generate significant service disruptions and infrastructure malfunctions that require prompt intervention. For this reason, data transfer processes are continuously monitored by teams of shifters. When an issue is detected, the operators report it through the Global Grid User Support (GGUS) ticketing system (Antoni, Bühler, Dres, Grein, & Roth, 2008), and experts and site maintainers take care of their solution. To give an idea of the volumes involved, the ATLAS collaboration alone experienced an average traffic of more than 2 PB per day in 2019 (Calafiura et al., 2020), corresponding to roughly 1.5-2 million files moved each day. Nearly 10% of these transfers failed producing about 100-200k errors on a daily basis. In total, transfer failures generated more than 4k incident reports filed in 2019[2] for all the LHC experiments (1141 for ATLAS only). Due to the complexity of the infrastructure and its layered composition, understanding the problem root causes and fixing them demands a great human effort – more than 100 ATLAS members were involved in 2019, corresponding to roughly 50 FTEs (Full-Time Equivalent) (Schovancová, 2019) – and it may entail undesired disservices. The average solving time may vary from a few hours or days – e.g. in the case of issues that are easy to solve or have already been dealt with in the past – to entire weeks – e.g. for unknown problems or more troublesome malfunctions that imply important software or hardware interventions. In practice, the median solving time for incidents reported by the ATLAS, CMS and LHCb collaborations in 2019 was around 17 days, with a 90-*th* percentile of 44 days and a long right tail extending over 100 days (see Fig. 8.1). When a transfer failure happens, the FTS log files are parsed and the transfers more relevant features are extracted and re-organized in a structured format. In particular, this involves collecting the exit status of each of the subsystems responsible for the transfer and appending them to compose a global error message. This information is then exposed to the on-duty shifters along with other characteristics – e.g. source and destination endpoints, file size, exchange protocol and so on – and visualizations – e.g. time evolution plots or site transfer efficiency – for more in-depth investigations.

Current operations are based on a *site-centric* approach where trained personnel monitors the status of the various services almost 24/7 and tries to spot hints of incorrect or undesired behaviors. In particular, the operators look at Grafana dashboards to get a high-level overview of the system. A usual starting point is the so-called efficiency matrix (Fig. 8.2), where the percentage of successful transfers is reported. The granularity level is customizable and it may range from global transfers between national cloud infrastructures involving more computing centers to a finer tracking of particular site exchanges or even specific

[2]available at: https://l.infn.it/ggus

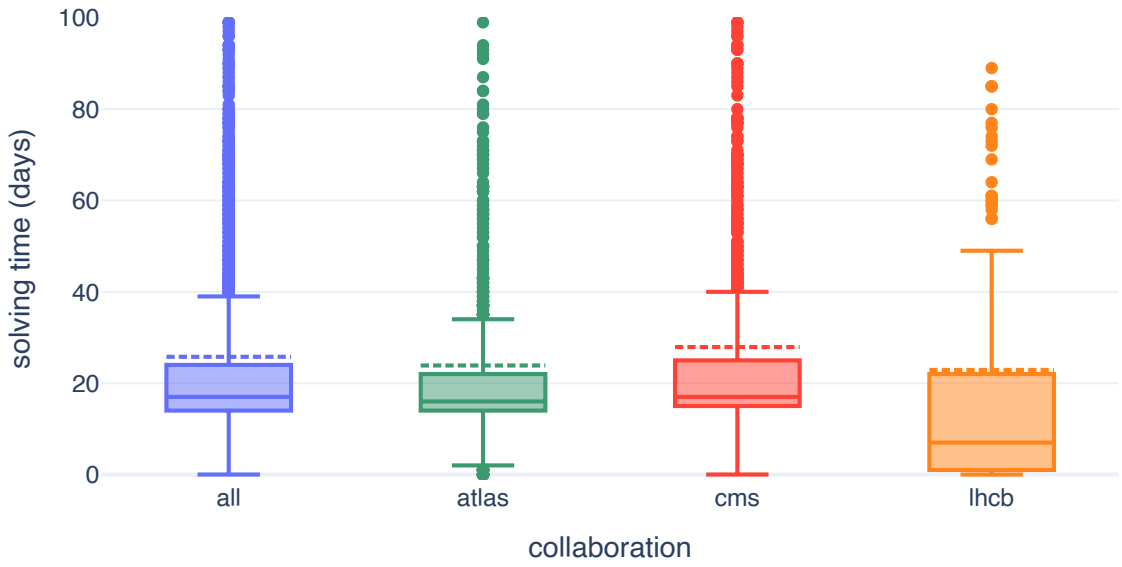


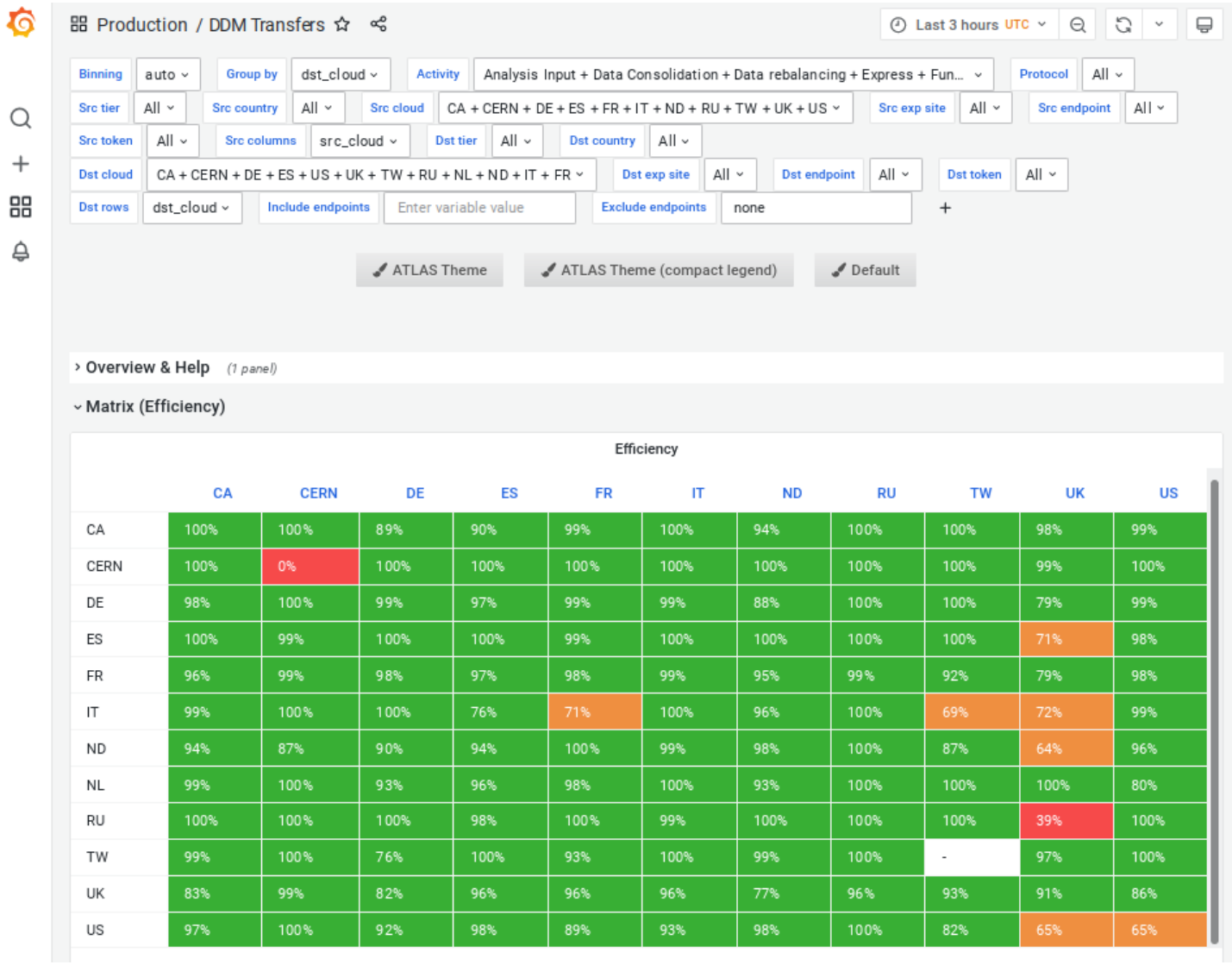
Figure 8.1: **Tickets solving time.** Boxplot of the distribution of the solving time for GGUS incidents reported in 2019 by ATLAS, CMS and LHCb collaborations.

endpoint links. When the efficiency falls below an acceptable threshold, typically 60-70%, on-duty shifters start to investigate the issue at a lower level by checking *i)* where the error happened, *ii)* how many errors are produced, *iii)* what is the time pattern (temporary, extended or cyclical) and *iv)* which error messages are generated. However, this procedure gives rise to many false alarms as it is usual to encounter problems that do not represent a real concern. For instance, this may happen when few transfers are attempted so even a low number of errors imply a high failure rate, or when there are after-effects of a transient issue that had already been fixed. Also, sometimes unnecessary drill-down activity is performed for actual issues that were already known, as in the case of ongoing tickets or site downtimes, for which reporting is not required. As a result, many human resources are employed in repetitive tasks of little scientific interest that would enormously benefit from automation.

In addition to that, the site-centric strategy described above has some drawbacks. Firstly, monitoring focuses on spotting where issues occur, while understanding the actual root causes is typically demanded to site experts in a subsequent investigation. Secondly, problems generating few error messages are usually ignored. This is natural, and to some extent desirable, as having limited resources forces us to address bigger malfunctioning first. However, that could be a potential pitfall in cases where promptly fixing a minor issue may prevent the rising of a more significant and longer to solve defect.

All these problems could be tackled programmatically by standardizing the

logging output of all the services. In this way, neat error messages would point directly to the source of the problem, thus allowing complete automation. However, the distributed nature of the infrastructure hampers such an approach. In fact, the opportunistic gathering of computing resources that led to WLCG entails many local configurations that are not easy to address using only a static strategy. Therefore, all these considerations expose the need for an intelligent support tool for speeding up infrastructure management to meet the productivity requirements for the near future.



	CA	CERN	DE	ES	FR	IT	ND	RU	TW	UK	US
CA	100%	100%	89%	90%	99%	100%	94%	100%	100%	98%	99%
CERN	100%	0%	100%	100%	100%	100%	100%	100%	100%	99%	100%
DE	98%	100%	99%	97%	99%	99%	88%	100%	100%	79%	99%
ES	100%	99%	100%	100%	99%	100%	100%	100%	100%	71%	98%
FR	96%	99%	98%	97%	98%	99%	95%	99%	92%	79%	98%
IT	99%	100%	100%	76%	71%	100%	96%	100%	69%	72%	99%
ND	94%	87%	90%	94%	100%	99%	98%	100%	87%	64%	96%
NL	99%	100%	93%	96%	98%	100%	93%	100%	100%	100%	80%
RU	100%	100%	100%	98%	100%	99%	100%	100%	100%	39%	100%
TW	99%	100%	76%	100%	93%	100%	99%	100%	-	97%	100%
UK	83%	99%	82%	96%	96%	96%	77%	96%	93%	91%	86%
US	97%	100%	92%	98%	89%	93%	98%	100%	82%	65%	65%

Figure 8.2: **Transfer efficiency matrix (Grafana).** Transfer sources are shown as columns and destinations as rows. The drop-down menus at the top allow for custom filtering at the desired level of granularity.

8.2.1 Related works

Analyzing error messages encountered in large-scale distributed systems has become one of the crucial tasks for monitoring computing resources. Thus, a variety of tools for log and error message parsing has already been explored. Some of these approaches include longest common subsequence (Du & Li, 2016), frequent pattern mining (Vaarandi, 2003; Vaarandi & Pihelgas, 2015), iterative partitioning (Makanju, Zincir-Heywood, & Milios, 2009), parsing trees (P. He, Zhu, Zheng, & Lyu, 2017) and hierarchical clustering (Fu, Lou, Wang, & Li, 2009) (see Zhu et al., 2019 for a thorough discussion and comparison). However, the existing tools present some drawbacks. First, most methods require a crucial pre-processing phase that may need deep customization for specific data. This limitation hampers their adaptation to novel use cases as different systems may have diverse logging conventions, terminology and structure. Furthermore, they do not allow error messages to be linked with additional entities other than the textual information, meaning that messages cannot be clustered along with auxiliary data.

Grigorieva and Grin (2021) present a pipeline consisting of several stages specifically tailored for data processing workflows within WLCG. First, the error messages are tokenized[3] and cleaned from digits, punctuation and special characters. Then, a hashing algorithm replaces the parametric parts of the message with a placeholder, and the resulting patterns are exploited for the following elaborations. In this way, the total amount of data is reduced by 90-95%. After the above pre-processing, the vectorization[3] stage is based on *word2vec* (Mikolov et al., 2013) that computes a numerical representation for each token. The overall message representation is then retrieved by averaging over single word embeddings. The resulting representation is then reduced in dimension by means of principal components analysis (Wold, Esbensen, & Geladi, 1987), and a DBSCAN (Ester, Kriegel, Sander, Xu, et al., 1996) algorithm is adopted for the clustering stage. Finally, cluster descriptions are extracted by searching common textual patterns and key phrases for all messages belonging to the same cluster.

Another interesting approach is presented in Q. Lin, Zhang, Lou, Zhang, and Chen (2016), where the authors propose a convenient pipeline to group logs of failed jobs and exploit the knowledge coming from previous failures. After substituting placeholders instead of parametric parts in the raw messages, each log is summarized using the unordered set of the events (log lines) it contains. A vectorization stage is then performed based on Inverse-Document event Frequency (IDF) and contrast-based weighting. The resulting numerical representation undergoes an agglomerative hierarchical clustering algorithm that finds groups of similar logs. The resulting cluster centroids are then taken as representative log sequences of

[3]for more details see Section 9.1.2

their respective groups, and they are compared to a knowledge base of previous failures and corresponding solutions. If the sequence similarity to one of the known issues is above a given threshold, the corresponding actions are applied to solve the problem. Otherwise, the log sequence is passed to system experts for manual inspection and the reference dataset is successively updated. In this way, human resources are involved only in handling new issues, while previous knowledge is exploited for recurrent ones.

Another way to look at this problem is through the lenses of Natural Language Processing (NLP), where related tasks have been addressed by adopting various strategies. A direct approach would be to regard error categorization as a specific example of topic modeling (Hofmann, 1999; Papadimitriou, Raghavan, Tamaki, & Vempala, 2000). In brief, topic modeling resorts to a low-dimensional latent representation of textual data where each latent dimension may be interpreted as a separate topic. In the context of error categorization, the different topics can be seen as high-level descriptions of different failures, and the messages as particular instances of the related problems. Alternatively, popular *language models* can also be leveraged (Devlin, Chang, Lee, & Toutanova, 2018; Peters et al., 2018; Brown et al., 2020). They consist of numeric representations for textual information – also known as *embeddings* – that preserve syntactic, grammatical and semantic relations of the original data, but in a lower dimension. This means that words similar in terms of meaning and usage are projected near to each other. Therefore, these techniques can be adopted to get convenient error embeddings where related failures are close in the sense of some distance or similarity measure, and clustering algorithms can be exploited to retrieve error categories.

8.3 Contribution

The goal of this work is to discuss a complementary approach to current operations for grid monitoring based on a computer-aided strategy independent of experiment-specific settings. In particular, we propose an unsupervised machine learning pipeline to identify clusters of similar failures which is centered on error messages rather than site performances (Clissa et al., submitted). The underpinning idea is to retrieve groups of related errors and expose the results to on-duty shifters as suggestions of potential issues to investigate further.

Alongside the conceptual formulation, we provide both qualitative and quantitative evaluation of the proposed approach. In particular, the pipeline is tested on one full day of operations and the results are thoroughly inspected to assess the quality of the discovered groups and highlight the major pros and limitations of the adopted methods (see Section 10.1). Furthermore, a quantitative assessment is performed by comparing the previous results with the GGUS reported incidents

(see Section 10.2).

Finally, we provide a full, scalable implementation of described approach[4] developed in compliance with the Operational Intelligence software framework[5] to allow fast integration and testing by the whole LHC community.

[4] https://l.infn.it/opint-pyspark
[5] https://l.infn.it/opint-framework

Chapter 9

Methods

This section presents a detailed description of an approach for dealing with File Transfer Service failures. The main goal is to analyze FTS failed transfers and identify error categories as suggestions of potential issues to investigate further by human experts. In terms of desiderata, the objective is to develop a tool independent of experiment-specific configurations and workflows. Hence, we focus on errors produced by FTS rather than other services which are adopted only by some collaborations, e.g. Rucio for ATLAS. Likewise, a minimal experts' effort is required. For this reason, we embrace an unsupervised learning approach to force the model to learn autonomously from data without needing a costly labeling phase of previous failures. As a byproduct, this strategy also avoids incurring expectation bias from prior (perhaps suboptimal) operative categorizations, and it enables discovering new failure patterns.

The proposed approach (Di Girolamo et al., 2022, Section 2.19) is inspired by the pipeline described in Q. Lin et al. (2016). In particular, we adopt a 3-step workflow consisting of *i)* vectorization, *ii)* clustering and *iii)* description stages. The last step of the original pipeline, i.e. checking recurrence, is excluded since no knowledge base is available for FTS failures. Also, our work is conceptually similar to the strategy described in Grigorieva and Grin (2021). However, some major differences are present in the pre-processing, clustering and description stages, and they are discussed in detail in Section 9.1.

In practice, our pipeline is developed inside the Operational Intelligence software framework[1], and it is implemented trying to cope with the runtime restrictions for online processing despite not renouncing model performance. To achieve that, the *Spark* (Zaharia et al., 2010) processing framework is used through the `pyspark` language to leverage the advantages of distributed computations for large

[1] `https://l.infn.it/opint-pyspark`

data. In particular, the whole pipeline is executed through the **SWAN service** (Piparo et al., 2018) that allows CERN users to run `Jupyter Notebooks` connected to Spark clusters on the WLCG infrastructure. Also, a careful pipeline design is adopted to alternate online and offline elaborations. Specifically, the learning phase of the vectorization stage is performed once on a big set of data (see Section 9.1.2 for more details) and it is freezed and re-used as is – possibly updating it once in a while. The clustering stage, instead, is performed online every day so to always expose the latest results to the shifters on-duty (see Sections 9.1.3 and 9.1.4 and Chapter 10 for more details).

9.1 FTS errors categorization

The pipeline illustrated in this work comprises an initial pre-processing step followed by the *vectorization*, *clustering* and *description* stages. Although conceptually similar to the workflow described in Grigorieva and Grin (2021), our approach considers some substantial modifications concerning mainly the pre-processing and the clustering stages. Indeed, we apply minimal pre-processing to limit hard-coded feature engineering and let the vectorization stage figure out linguistic features of the error messages – e.g. grammar, syntax, lexicon and semantic – on its own. The rationale behind this choice is that the resulting representation should be more expressive, thus better modeling the semantic of the messages and easing the successive clustering phase.

Figure 9.1 reports a diagram that summarizes our workflow from the initial error message to the final outputs, and the next subsections provide a thorough description of each of the stages.

9.1.1 Pre-processing

The pre-processing phase is crucial to any data analysis workflow. Various best practices are suggested for data cleaning, and custom feature engineering is often adopted to feed models with the most relevant information in the most suitable format. Our approach tries to limit these elaborations to the bare minimum to avoid injecting too much prior knowledge into the system and probe the model's ability to learn by itself. The resulting pipeline is described below and summarized in Table 9.1.

As a first step, the raw error strings are transformed to lowercase and enriched by appending the source and destination hostnames. In particular, both hostnames are inserted at the end of each message with prepended `src_` or `dst_` prefixes to distinguish whether they were involved as source or destination, re-

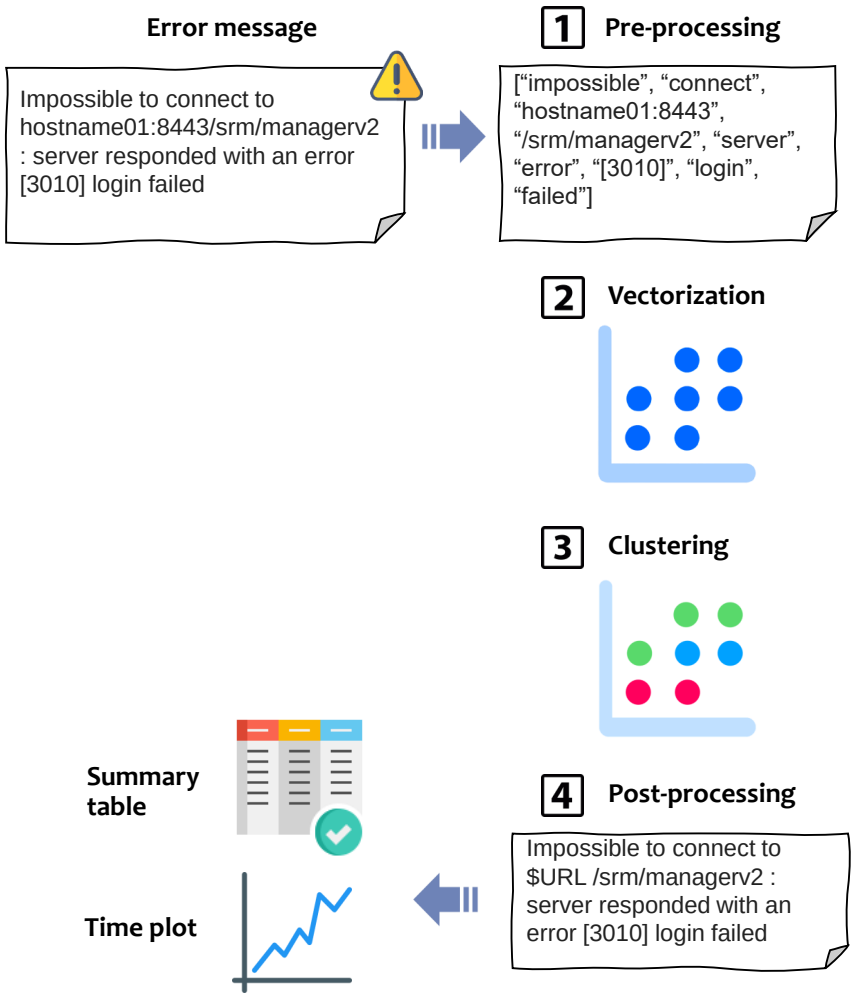


Figure 9.1: **Pipeline diagram.** The error message is first pre-processed and split into tokens (1). Then, the vectorization stage transforms the textual information into numeric data (2). The next step is clustering, where similar error messages are grouped (3). Finally, the messages are post-processed to get common patterns (4) and the resulting clusters are presented to the shifters in the form of a summary table and time evolution plots.

spectively. The resulting text then undergoes a process of quantization whereby the raw strings are decomposed into unitary pieces of information. This process is commonly referred to as tokenization and the resulting atomic units are called tokens. Various approaches have been proposed in the literature ranging from simply using words (Bengio, Ducharme, Vincent, & Jauvin, 2003; McCann, Bradbury, Xiong, & Socher, 2017) or characters (Ling et al., 2015; Dhingra, Zhou, Fitzpatrick, Muehl, & Cohen, 2016), to more complex strategies involving subwords (Gage, 1994; Sennrich, Haddow, & Birch, 2016), sentences (Kiros et al., 2015), documents (Le & Mikolov, 2014) and topics (Niu, Dai, Zhang, & Chen, 2015). In our case, we resort to whitespace tokenization for the sake of simplicity, which means individual words are used as tokens. Once tokens are obtained, they are stripped of leading and trailing punctuation (`":;,.-"`). After that, tokens corresponding to common English stopwords[2] or unuseful punctuation (`":-+"`) are discarded. Finally, the URL addresses are split into two components: the net location and the relative path of the requested resources. For instance, `httpg://<hostname>:<port>:/srm/managerv2` is decomposed as `httpg://<hostname>:<port>` and `srm/managerv2`. In this way, it is possible to exploit the compositional structure of the URL addresses to reduce the vocabulary of unique tokens. Also, this allows the model to disentangle the contribution of the single parts in different messages.

9.1.2 Vectorization

In the vectorization stage we leverage the **word2vec** language model (Mikolov et al., 2013) to compute message embeddings starting from pre-processed tokens. Although more recent and powerful alternatives as the ones described in Section 8.2.1 are available, they do not work well with short-text data (Albalawi, Yeap, & Benyoucef, 2020). Thus, the simpler yet effective word2vec model is used here.

The idea behind word2vec is to find a convenient mapping between tokens (words) and a vectorial space where token linguistic relations are preserved, thus producing a distributed representation of words. In this way, we obtain a numerical representation of textual data that quantitative algorithms can further process. In particular, two alternative implementations of the word2vec model are available (see Fig. 9.2). Both alternatives are based on a shallow (2-layer) neural network architecture and use a sliding window (context) of fixed size, w, around the current word, w_t. The continuous bag-of-words (CBOW) model sets the learning objective to predict the current word from the surrounding context. Conversely, the skip-gram model tries to guess the terms present in the context window starting from the current word. To achieve that, the input words are first represented as

[2] refer to `pyspark.ml.feature.StopWordsRemover` documentation for a full list

raw message	"DESTINATION OVERWRITE srm-ifce err: Communication error on send, err: [SE][srmRm][] httpg://hostname01.Site-4.ch:8443/srm/managerv2: CGSI-gSOAP running on fts-address-004.cern.ch reports Error initializing context GSS Major Status: Authentication Failed GSS Minor Status Error Chain: globus_gsi_gssapi: SSL handshake problems globus_gsi_callback_module: Could not verify credential globus_gsi_callback_module: Could not verify credential globus_gsi_callback_module: The certificate has been revoked: Serial number = -1 (0xFFFFFFFFFFF"
append hostnames	"DESTINATION OVERWRITE srm-ifce err: Communication error on send, err: [SE][srmRm][] httpg://hostname01.Site-4.ch:8443/srm/managerv2: CGSI-gSOAP running on fts-address-004.cern.ch reports Error initializing context GSS Major Status: Authentication Failed GSS Minor Status Error Chain: globus_gsi_gssapi: SSL handshake problems globus_gsi_callback_module: Could not verify credential globus_gsi_callback_module: Could not verify credential globus_gsi_callback_module: The certificate has been revoked: Serial number = -1 (0xFFFFFFFFFFF src_srmatlas.pic.es dst_hostname01.Site-4.ch"
tokenization	["destination", "overwrite", "srm-ifce", "err:", "communication", "error", "on", "send,", "err:", "[se][srmrm][]", "httpg://hostname01.Site-4.ch:8443:/srm/managerv2:", "gsi-gsoap", "running", "on", "fts-atlas-005.cern.ch", "reports", "error", "initializing", "context", "gss", "major", "status:", "authentication", "failed", "gss", "minor", "status", "error", "chain:", "globus_gsi_gssapi:", "ssl", "handshake", "problems", "globus_gsi_callback_module:", "could", "not", "verify", "credential", "globus_gsi_callback_module:", "could", "not", "verify", "credential", "globus_gsi_callback_module:", "the", "certificate", "has", "been", "revoked:", "serial", "number", "=", "-1", "(0xfffffffff", "src_srmatlas.pic.es", "dst_hostname01.Site-4.ch"]
remove punctuation	["destination", "overwrite", "srm-ifce", "err", "communication", "error", "on", "send", "err", "[se][srmrm][]", "httpg://hostname01.Site-4.ch:8443:/srm/managerv2", "cgsi-gsoap", "running", "on", "fts-atlas-005.cern.ch", "reports", "error", "initializing", "context", "gss", "major", "status", "authentication", "failed", "gss", "minor", "status", "error", "chain", "globus_gsi_gssapi", "ssl", "handshake", "problems", "globus_gsi_callback_module", "could", "not", "verify", "credential", "globus_gsi_callback_module", "could", "not", "verify", "credential", "globus_gsi_callback_module", "the", "certificate", "has", "been", "revoked", "serial", "number", "=", "1", "(0xfffffffff", "src_srmatlas.pic.es", "dst_hostname01.Site-4.ch"]
remove stopwords	["destination", "overwrite", "srm-ifce", "err", "communication", "error", "send", "err", "[se][srmrm][]", "httpg://hostname01.Site-4.ch:8443:/srm/managerv2,cgsi-gsoap", "running", "fts-atlas-005.cern.ch", "reports", "error", "initializing", "context", "gss", "major", "status", "authentication", "failed", "gss", "minor", "status", "error", "chain", "globus_gsi_gssapi", "ssl", "handshake", "problems", "globus_gsi_callback_module", "verify", "credential", "globus_gsi_callback_module", "verify", "credential", "globus_gsi_callback_module", "certificate", "revoked", "serial", "number", "=", "1", "(0xfffffffff", "src_srmatlas.pic.es", "dst_hostname01.Site-4.ch"]
url split	["destination", "overwrite", "srm-ifce", "err", "communication", "error", "send", "err", "[se][srmrm][]", "httpg://hostname01.Site-4.ch:8443", "/srm/managerv2", "cgsi-gsoap", "running", "fts-atlas-005.cern.ch", "reports", "error", "initializing", "context", "gss", "major", "status", "authentication", "failed", "gss", "minor", "status", "error", "chain", "globus_gsi_gssapi", "ssl", "handshake", "problems", "globus_gsi_callback_module", "verify", "credential", "globus_gsi_callback_module", "verify", "credential", "globus_gsi_callback_module", "certificate", "revoked", "serial", "number", "=", "1", "(0xfffffffff", "src_srmatlas.pic.es", "dst_hostname01.Site-4.ch"]

Table 9.1: **Message pre-processing pipeline.** The table illustrates the pre-processing steps (left) and the resulting data (right) for a sample error message. The raw error string is reported at the top, and the resulting pre-processed data at the bottom.

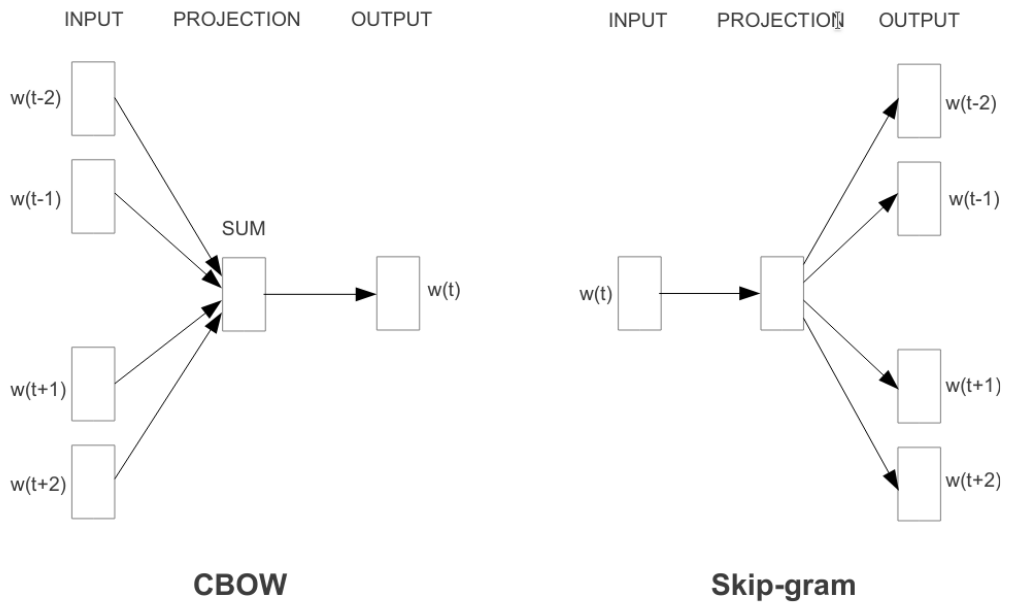


Figure 9.2: **Word2vec model architectures.** The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. The figure is borrowed from the original paper (Mikolov et al., 2013).

1-hot vectors of size V, where V is the number of terms present in the corpus (vocabulary size). During this encoding step, a minimum allowed token frequency, *min_count* is set to discard terms appearing less than such threshold. This trick limits the vocabulary size by excluding rare tokens, thus lowering the computational requirements. These input vectors pass through the first layer that projects them to a hidden space of customizable dimension, h. This mapping (embedding) is learned during the training phase such that terms often used together or in similar contexts should be projected nearby. Finally, the resulting representations are put through the next network layer to get the predicted context (in the case of CBOW) or current words (skip-gram).

This work adopts the word2vec version exploiting the skip-gram architecture and uses its `pyspark` implementation. Specifically, the model is trained on one month of data from 2020-10-01 to 2020-10-31. In total, nearly 28.6 M error messages are analyzed, corresponding to a vocabulary of 970 unique tokens. Once the model is trained, the resulting word embeddings are used to transform single tokens into numerical vectors, and they are then averaged to get the corresponding message representations. Regarding hyper-parameters, the window size, the

embedding size and the minimum count were the ones affecting the final representation the most. For this reason, a grid-search is conducted to compare alternative parametrizations. The optimal configuration is then chosen based on the compactness of the resulting groups in the following clustering stage. Specifically, the values of $w = 12$, $h = 300$ and $min\_count = 50$ seem to work best in our use case.

9.1.3 Clustering

The next step of the pipeline is the clustering stage. Cluster analysis examines the task of grouping a set of objects based on a given measure of distance or similarity. This is a well-studied topic typically conducted to discover underlying structures in data during exploratory analysis or pattern recognition.

One of the most intuitive and widely adopted strategies to tackle this problem is the so-called *k-means* algorithm (Lloyd, 1982). This technique is an iterative local search solution, and it refers to a particular formulation of the problem where the number of output groups, k, is supposedly known. The algorithm evolves in four simple steps, starting from this prior information and after establishing a suitable distance measure to express the similarity between data points. Figure 9.3 summarizes the k-means pipeline from raw data to the final clusters discovered applied to some toy data points. In the initialization phase, k arbitrary centers (also named *centroids*) are chosen uniformly at random from the data points (Fig. 9.3a). In the next step, the data are split into clusters by mapping each data point to the nearest center (Fig. 9.3b). Once the groups are formed, the centroids are recomputed as the center of mass of all points assigned to the corresponding cluster (Fig. 9.3c). Finally, the last two steps are repeated until a convergence criterion is met (Fig. 9.3d).

In this study, we resort to clustering for grouping messages including similar content. The resulting clusters are therefore interpreted as error categories. In particular, we adopt a slight variation of the above algorithm referred to as **k-means++** (Arthur & Vassilvitskii, 2007). The two techniques share the same workflow except for how the starting centroids are initialized. Specifically, the k-means algorithm assigns equal probability to all the data points and then samples k centers. For k-means++ instead, only the first centroid is chosen uniformly at random. The following $k - 1$ centers are sampled from data points with probability inversely proportional to the distance between each point and the closest predefined centroid. This careful seeding strategy ensures the initial centers are more spread across the data points, thus favoring better clustering results and a faster convergence. Despite more advanced clustering algorithms are available and may be applied to our use case, e.g. DBSCAN (Ester et al., 1996), HDBSCAN (McInnes, Healy, & Astels, 2017), BIRCH (T. Zhang, Ramakrishnan, & Livny,

(a) initialization (b) update clusters

(c) update centroids (d) final result

Figure 9.3: **K-Means clustering.** The algorithm performs an iterative search by alternately grouping observations around the current centroids (b) and updating the centers of the clusters (c).

1996), OPTICS (Ankerst, Breunig, Kriegel, & Sander, 1999), spectral clustering (Ng, Jordan, & Weiss, 2002) and so on, the choice of the a k-means algorithm is justified by its intuitive approach and good performance in practice in a wide range of applications (Von Luxburg et al., 2012). Also, a perhaps more profound and substantial motive is that the clustering strategy may be seen as a functional but not primary pipeline stage. Indeed, the learned language model determines the geometry of the embedded space, thus influencing the point cloud shapes of different error categories. For this reason, we embrace the idea that a simple clustering algorithm is preferable, and particular attention must be devoted to tuning the vectorization stage for easing the subsequent clustering, possibly even fostering the learning of an optimal representation for a specific clustering algorithm (Yang, Fu, Sidiropoulos, & Hong, 2017).

In order to demonstrate the approach, we report the analysis of FTS data from one full day of operation (2021-01-15), corresponding to roughly 1 M errors and 1.5 GB of data. To help the successive evaluation phase, only transfers between Tier-0, Tier-1s and Tier-2s are considered in the analysis. In practice, more hyper-parameter configurations are explored to test the effect of different choices. The first set of investigations regards the measure of similarity adopted to form the clusters. In this respect, we tested two alternatives, cosine similarity and the euclidean distance. The former consistently outperformed euclidean distance in all attempted experiments, both in terms of cluster geometrical properties and interpretability of the results. For this reason, only the the analysis involving **cosine similarity** is reported in the following.

Another crucial hyper-parameter is given by the assumed number of clusters, k. This represents the number of error categories in our case, which is not known in advance. Therefore a grid search for $k \in [12, 15, 20, 30]$ is exploited to retrieve a reasonable estimate from the data. For this purpose, two geometrical criteria are considered to compare results of different settings, namely the *Within cluster Sum of Squared Errors* (WSSE) and the *Average Silhouette Width* (ASW) (Rousseeuw, 1987). The WSSE measures the internal cluster variability, so the lower its value, the better the performance. In terms of its definition, the WSSE is based on the total sum of squared distances between the points of each group and the correspondent centroid:

$$\mathrm{WSSE}\left(\mathrm{dist}, k\right) = \sum_{j=1}^{k} \sum_{x_i \in C_j} \mathrm{dist}\left(x_i - \bar{x}_j\right) \tag{9.1}$$

where x_i is a generic data point, *dist* is a desired distance measure, and C_j and $\bar{x}_j$ indicate a generic cluster and its centroid, respectively. Although compactness is certainly a desirable property for the output groups, this metric is strongly affected

by the scale of the variables and the number of observations. Indeed, the clusters exhibit a greater variability as their points present higher values and/or the cluster size increases, thus causing the WSSE to explode. Also, being unbounded by its nature, i.e. WSSE $\in [0, +\text{inf}]$, the WSSE is of difficult interpretation on its own and it only makes sense when compared to other values.
A better metric is the so-called average silhouette width. In brief, the ASW is a measure of clustering performance that accounts for both internal homogeneity and external separation of the clusters. In particular, let $\bar{a}_i$ be the average distance of x_i from all the other points belonging to the same cluster C_I. Also, let b_i be the minimum average distance of x_i from the observations in all the other clusters $C_j, \forall j \neq I$. Then, the ASW is defined as:

$$\text{ASW}\left(\text{dist}, k\right) = \frac{1}{n} \sum_{i=1}^{n} \frac{b_i - \bar{a}_i}{\max\left(\bar{a}_i, b_i\right)} \tag{9.2}$$

where n is the total number of observations, i.e. error messages in our case. The average silhouette is much more intuitive to interpret than the WSSE as its value is bounded in the interval $[-1, +1]$. In practice, negative values of the ASW mean that points of one cluster are on average more similar to observations of other groups than the ones of their own cluster. A value of 0, instead, suggest that the groups are not really distinguishable, thus making the assignation of single observations to any of the clusters arbitrary. Finally, positive values close to 1 testify that the clustering produces nicely homogeneous groups that are also well separated. Given the more intuitive reading of ASW values, the latter is used in the following as the main figure of merit.

The results of the comparison between WSSE and ASW for different values of k are reported in Fig. 9.4. Both indicators tend to improve as the number of clusters increases. In particular, a value of $k = 30$ clusters seems to be optimal according to both criteria. Notably, however, the ASW indicator reaches very high values (around 0.9) even for lower k values. For this reason, the configuration having $\boldsymbol{k = 15}$ is preferred to limit the number of suggested issues and minimize the shifters effort.

9.1.4 Cluster description

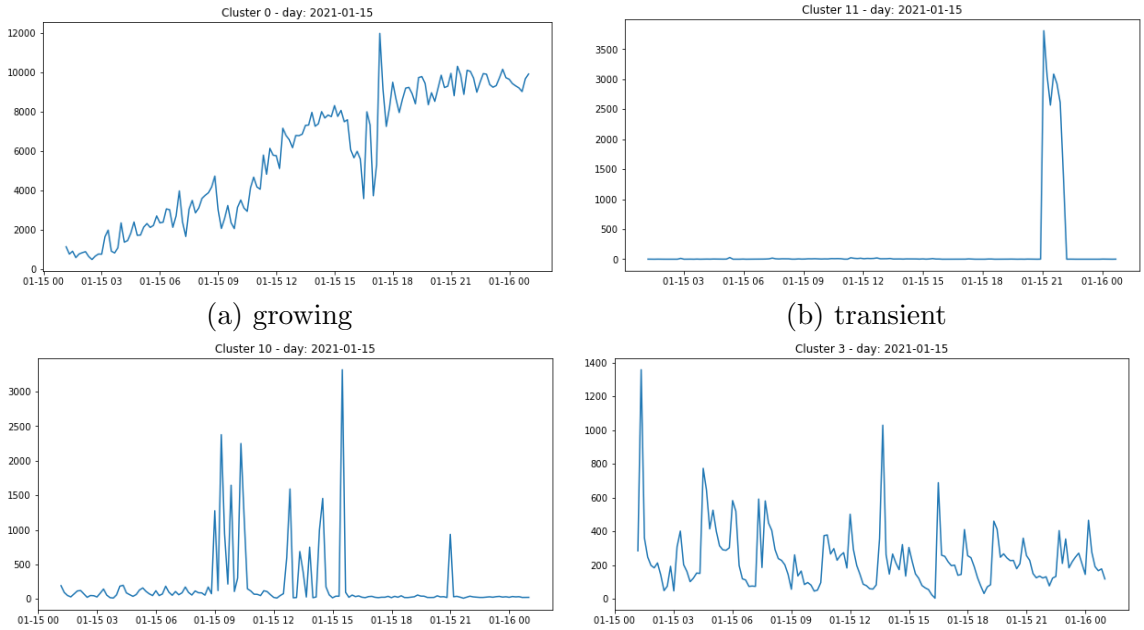
The last stage of the pipeline is cluster description. This step is fundamental to present the results in the most intelligible and immediate format for end-users. Indeed, given the unsupervised learning approach adopted, the interpretation of the clustering output resorts to the manual inspection of each group's content. This, in turn, potentially mean reading hundreds of error strings, comparing the

Figure 9.4: **Optimization of k.** The plot shows the value of the WSSE and ASW metrics as a function of the number of clusters, k. The hexagonal markers indicate the optimal values, which correspond to $k = 30$ for both indicators.

source and destination information, and spotting suspect time patterns. Therefore, producing a nice and compact visualization of the results is paramount to make the approach effective and avoid excessive manual checks by the operators. For this reason, the clustering results are summarized into two complementary outputs that are presented to the shifters.

First, the *summary table* represents the most important and informative visualization. This output is obtained by a first pre-aggregation of the clusters and is organized in a tabular format. The first three columns provide numeric summaries concerning the cluster size, the number of unique strings within each group, and the corresponding number of unique patterns. The latter is obtained from the raw strings by means of an *abstraction mechanism*[3] that removes the parametric parts – like file paths, IP addresses, URLs, checksum values, and so on – and replaces them by parameter-specific placeholders – e.g. `$FILE_PATH`, `$ADDRESS`, `$URL` and `$CHECKSUM`, respectively. The core part of this visualization is then represented by the *Top 3* section. Here, the three most frequent triplets of `<pattern>-<source>-<destination>` are reported in descending order for each cluster, alongside their cardinality and the percentage over the cluster size. Such information provides several precious insights for spotting the source of potential problems, e.g. whether a pattern is responsible for a large number of

[3]refer to the full implementation for more details: https://l.infn.it/opint-abstraction



(a) growing (b) transient

(c) resolved (d) cyclical

Figure 9.5: **Time evolution charts.** The figure illustrates several time patterns for the generated failures in 4 different clusters. Each plot reports the count of errors in bins of 10 minutes.

failures or if it accounts for a conspicuous fraction of the cluster. In addition, this representation allows us to investigate the contribution of source/destination pairs to each cluster. In this way, it is possible to discriminate failures based on both the nature of the problem and the location where they occurred. An example of summary table is reported in Figs. 10.1 and 10.2.

The second output of the pipeline consists of a time-series chart depicting the temporal evolution of the number of errors generated by each cluster (Fig. 9.5). This piece of information is crucial to discriminate between serious issues that require immediate actions (Figs. 9.5a and 9.5d) and transient (Fig. 9.5b) problems.

Overall, the idea behind our pipeline is to exploit the summary tables and the time plots for each cluster as suggestions of potential issues to investigate further. In this way, the shifters can have a first grasp of what kind of failures are observed and their corresponding amounts (Top-3 section), also having an indication of where they are happening (source/destination sites). Moreover, by looking at the time charts it is possible to immediately discard transient (Fig. 9.5b) or resolved (Fig. 9.5c) problems based on the evolution of the number of generated failures over time.

Chapter 10

Results

This section reports the results of the cluster analysis described in Section 9.1.3. The performance assessment is one of the trickiest parts when coming to unsupervised methods – and clustering in particular –, and it has long been debated (Von Luxburg et al., 2012; Guyon, Von Luxburg, & Williamson, 2009). The most straightforward approach is to resort to the within sum of squared errors and average silhouette width similarity measures leveraged during the training phase (see WSSE and ASW in Section 9.1.3). However, these metrics treat the data as points in the space and measure the resulting shape of the formed clusters. Hence, they rely on the assumptions that *i)* the "correct" output shape is known, *ii)* the similarity measure is informative about the desired morphology, and *iii)* the numeric representation of the data points is consistent with the avowed geometry. A general target for *i)* is commonly established as producing compact agglomerations of points, possibly well separated among each other. In the case of transfer errors, it is unclear what a proper output shape should be, but the general approach seems reasonable. Furthermore, the ASW indeed captures the desired morphology, which provides a convenient tool for *ii)*. Nevertheless, the previous building blocks can be leveraged only as long as the numeric representation of error strings complies with the above schema. This, in turn, presupposes that we know what a correct representation should be and that the adopted language model can reproduce an accurate mapping of the raw data into such embedding. Unfortunately, this element is unknown in advance and hardly explorable a posteriori unless recurring to human review and interpretation. Furthermore, the above construction is solely based on geometric arguments, and the actual meaning of the data points, i.e. whether the clustered messages share the same content, is not taken into account.

In light of the above concerns, our work splits the evaluation of performances into two complementary phases. First, a qualitative assessment explores the cluster contents and expresses the goodness of fit based on their interpretability, i.e. how

messages of the same cluster resemble each other's meaning. Second, a quantitative evaluation is addressed by cross-checking the clustering result against the GGUS reported incidents. In this way, a more direct measure of impact is given by reckoning the ability of our approach to mimic current operations.

10.1 Qualitative assessment: interpretability

This section presents a qualitative assessment of the clustering performances based on the interpretability of the discovered groups of messages. In particular, the discussion is articulated by simulating the shifter's perspective when reading the pipeline outputs. In the following, we report five cherry-picked examples to showcase our approach's major successes and failures. Specifically, we first illustrate a thorough examination of the biggest cluster discovered (see Fig. 10.1, cluster with id = 0). Then we highlight some strengths and limitations of our approach, bringing other exemplary cases as evidence. The same procedure and similar conclusions apply likewise to most groups. Thus, a complete dissertation is omitted here for conciseness[1].

The main output of our pipeline is the summary table illustrated in Figs. 10.1 and 10.2, which reports a succinct highlight of the cluster contents and represents the most substantial source of information. A reasonable reading approach is to start with the groups including more errors and gradually proceed with the smaller ones. In this case, the biggest cluster is shown in Fig. 10.1 in the first row with id = 0. Despite including almost 820k error strings (`# cluster size`), the actual number of different messages is only 117(`# strings`). This number further reduces to simply 14 unique patterns (`# patterns`) after the abstraction mechanism described in Section 9.1.4 is applied, which is way more manageable for manual inspection than the initial cluster size. A second insight is then provided by the *Top-3* section. Including the auxiliary information about the source and destination sites involved makes it evident as the failures are united by the same error template and destination site. This suggests that `Site-4` may have a problem and that its root cause is linked to the error pattern reported in the `message` column. Finally, the last piece of information to consider is the time evolution plot (see Fig. 9.5a). In this case, the cluster shows an increasing trend throughout the whole day of analysis. Specifically, the number of generated failures grows from less than 2000 errors at the beginning of the day to a value around five times higher, with an incremented boost from 9 a.m. onwards. By and large, all these factors clearly advise that a potential issue is happening at `Site-4` as it always appears as a destination. Also, the message information further suggests that the

[1] full results available at: https://l.infn.it/opint-results

failure is linked to a *revoked certificate* that cannot be verified. Finally, the time chart shows that the problem is escalating and need prompt intervention.

Despite providing only good proxies of the actual end goals – i.e. *root causes* and *solving actions* –, this rapid analysis already points to actionable insights regarding *where* and *what* faults occur and whether they represent a real concern. Notice that one can draw similar conclusions by looking separately at the site transfer efficiency and the most frequent unique strings or patterns. However, observing high failure rates for `Site-4` only answers to *where* the faults occur. Likewise, the information contained in the errors only relates to the *what* part of the question. Thus, both approaches would lead to partial conclusions and require additional investigations to reach the same result. Conversely, our approach addresses the two tasks together, thus letting the conclusion emerge rapidly and naturally. Remarkably, a further advantage is that one can leverage both site and pattern information for more precise indications. For instance, one could hypothesize that not only the `Site-4` is experiencing a problem, but the issue is limited to incoming connections. Indeed, `Site-4` is involved only as a destination, and the error patterns point to something related to `destination overwrite`. Therefore, the previous advantages show how shifting from the current site-centric focus to a hybrid strategy based on error messages and auxiliary information is beneficial.

ID	cluster size	# strings	# patterns	Top 3				
				message	n	%	source rcsite	destination rcsite
0	819465	117	14	destination overwrite srm-ifce err communication error on send err [se][srmrm][] $URL /srm/managerv2 cgsi-gsoap running on $ADDRESS reports error initializing context gss major status authentication failed gss minor status error chain globus_gsi_gssapi ssl handshake problems globus_gsi_callback_module could not verify credential globus_gsi_callback_module could not verify credential globus_gsi_callback_module the certificate has been revoked serial number = 1 (0xfffffffffff	85545	10.44%	Site-1	Site-4
				destination overwrite srm-ifce err communication error on send err [se][srmrm][] $URL /srm/managerv2 cgsi-gsoap running on $ADDRESS reports error initializing context gss major status authentication failed gss minor status error chain globus_gsi_gssapi ssl handshake problems globus_gsi_callback_module could not verify credential globus_gsi_callback_module could not verify credential globus_gsi_callback_module the certificate has been revoked serial number = 1 (0xfffffffffff	84453	10.31%	Site-2	Site-4
				destination overwrite srm-ifce err communication error on send err [se][srmrm][] $URL /srm/managerv2 cgsi-gsoap running on $ADDRESS reports error initializing context gss major status authentication failed gss minor status error chain globus_gsi_gssapi ssl handshake problems globus_gsi_callback_module could not verify credential globus_gsi_callback_module could not verify credential globus_gsi_callback_module the certificate has been revoked serial number = 1 (0xfffffffffff	77410	9.45%	Site-3	Site-4
6	9673	347	60	source srm_get_turl srm-ifce err connection timed out err [se][statusofgetrequest] [etimedout] \$URL /srm/managerv2 user timeout over	1838	19.00%	Site-22	Site-46
				transfer globus_ftp_client the server responded with an error 421 service busy connection limit exceeded please try again later closing control connection	522	5.40%	Site-33	Site-47
				transfer globus_ftp_client the server responded with an error 421 service busy connection limit exceeded please try again later closing control connection	300	3.10%	Site-29	Site-47
3	34183	1568	1537	error reported from srm_ifce 2 [se][ls][srm_invalid_path] no such file or directory	13118	38.38%	Site-12	Site-35
				error reported from srm_ifce 2 [se][ls][srm_invalid_path] no such file or directory	9333	27.30%	Site-12	Site-17
				error reported from srm_ifce 2 [se][ls][srm_invalid_path] no such file or directory	1707	4.99%	Site-12	Site-22

Figure 10.1: **Summary table: successes.** The figure illustrates the main achievements of the pipeline. Cluster 3 provides immediate indication of the error type, i.e. `message`, and where it occurs (green). The others also suggest the approach is actually learning to understand message parameters and message semantic (yellow, clusters 0 and 6).

In addition to the practical usage of our pipeline, the results illustrated in Figs. 10.1 and 10.2 expose interesting insights about what the models are actually learning. For instance, the substantial reduction observed passing from errors to patterns suggests that the pipeline has learned something similar to an abstraction mechanism. Indeed, the raw error strings of `cluster 0` differ only by the `$URL` and `$ADDRESS` parameters (see `message` column). Although one may argue that the same could be obtained using a flexible parsing strategy, the superiority of our approach is even more evident in `cluster 6` (Fig. 10.1). In this case, the clustering joins two patterns with a far less straightforward linkage. In fact, this result appears to resemble the human association that `connection timed out` (first pattern) may be linked to a `service busy connection limit exceeded` (second and third) problem. Notably, this is a much higher level of abstraction with respect to a smart parsing approach, and it goes way beyond what one could achieve based on good abstraction heuristics. Clearly, this property is remarkable and highly desirable in practice, as it testifies that the approach produces a good embedded representation and recognizes the similarity of messages sharing similar content. In particular, this holds not only up to some parametric parts but also in terms of their actual meaning. In turn, this observation corroborates the initial design choice of applying minimal pre-processing and letting the model learn by itself.

Another clear example of success is provided by the `cluster 3` (Fig. 10.1), where the visualization makes it immediate for the shifter to understand that the issue is related to a missing file (`no such file or directory`) at `Site-12`.

However, our pipeline comes also with some limitations (Fig. 10.2). For instance, the two patterns reported in `cluster 4` show a more vague connection that would require more in-depth investigation. As a matter of fact, they seem to be linked due to a generic `server responded with an error` which is a very generic incipit to several error strings. Apart from that, the error codes are different (`[3021]` VS `[3010]`, which may imply the clustering is too coarse and a more refined distinction is needed. Also, the messages point to seemingly extraneous issues (storage VS authentication). Such observations expose two limitations. On the one hand, tuning the pipeline to meet the desired level of granularity when separating different groups is extremely complex. On the other hand, this behavior may be due to the difficulty in comparing longer strings (first and second patterns) with short-text (third).

Another drawback is related to how outliers are handled. The k-means algorithm is bounded to the specified number of clusters, k, which sets the number of output groups irrespective of the underlying structure of the data. As a result, the outliers are often incorporated into the closer cluster. When the latter is big enough, they probably pass undetected as they are dispersed into un heap of other

messages. However, they may contaminate other clusters when the affected group has a comparable size, as in the case of second and third patterns in `cluster 2`.

ID	cluster size	# strings	# patterns	Top 3				
				message	n	%	source rcsite	destination rcsite
4	51370	10108	814	transfer globus_ftp_client the server responded with an error 500 command failed open/create [error] server responded with an error [3021] unable to get quota space quota not defined or exhausted $FILE_PATH disk quota exceeded	4912	9.56%	Site-7	Site-31
				transfer globus_ftp_client the server responded with an error 500 command failed open/create [error] server responded with an error [3021] unable to get quota space quota not defined or exhausted $FILE_PATH disk quota exceeded	3709	7.22%	Site-8	Site-31
				error on $IPv6 [error] server responded with an error [3010] login failed	2950	5.74%	Site-9	Site-33
2	15132	11	9	transfer globus_ftp_control gss_init_sec_context failed globus_gsi_callback_module could not verify credential globus_gsi_callback_module could not verify credential globus_gsi_callback_module the certificate has been revoked serial number = 1 (0xffffffffffffffff) subject=/c=bm/o=quovadis limited/cn=quovadis grid ica g2	2048	13.53%	Site-27	Site-42
				destination srm_put_turl error on the turl request [se][statusofputrequest] [srm_duplication_error] cannot srmput file because it already exists!	1431	9.46%	Site-28	Site-12
				destination srm_put_turl error on the turl request [se][statusofputrequest] [srm_duplication_error] cannot srmput file because it already exists!	914	6.04%	Site-15	Site-12

Figure 10.2: **Summary table: limitations.** The two clusters show evidence of contamination (red) due to generic partial matching (yellow, cluster 4) or outliers aggregation (cluster 2).

N. Clusters	ASW	WSSE	Perfect Match	Fuzzy Match	Partial Match	False Positives	False Negatives
15	0.89	17107	7	3	2	3	1

Table 10.1: **GGUS pre-validation.** Summary of the cross-check between clusters and incidents reported in GGUS. Most of the groups discovered are linked to reported issues, with only 3 false positives and 1 false negative.

10.2 Quantitative assessment: GGUS tickets

The drawback of unsupervised techniques lies in the inherent difficulty of the evaluation phase, as no ground truth is available for comparison (Von Luxburg et al., 2012). In order to overcome this limitation, we have conducted extensive testing using incidents reported in GGUS as a benchmark. In this way, we attempt to provide a quantitative assessment of the pipeline performances and a more direct measure of its potential impact when applied in practice. In particular, we explore the overlapping between discovered clusters and the reported issues in two directions expressing alternative perspectives to the problem. On one side, we evaluate the usefulness of our approach for the shifters, i.e. how clusters explain failures/tickets (*direct association*). On the other, we study the overall capacity of the pipeline to discover and highlight issues – i.e. how many failures/tickets are reflected in the clusters (*inverse association*). In the first case, the objective is to limit the effort of the operators by suggesting as few potential failures as possible, meanwhile still highlighting the major concerns for the infrastructure. Thus, the focus is on limiting false positives at the expense of neglecting minor issues. On the contrary, the second point of view requires a more comprehensive search aimed at isolating all the ongoing malfunctions, irrespectively of their current priority. Hence, this time the focus is on maximizing true positives. Table 10.1 reports a summary of the evaluation according to both perspectives..

Concerning the first angle, we consider GGUS issues reported in a skewed time window of 17 days (01-01 to 01-18) around the day of the analysis for a total of 20 tickets related to data transfer failures. Adopting this filtering strategy is convenient since it considers both previously known issues and delayed detections. The former is necessary because standard practice in current operations requests not to open new incident reports when related investigations are already ongoing. Hence, considering only tickets opened on the analysis day may lead to incorrect conclusions. Instead, the latter is convenient to account for a "grace period" if the operators do not promptly spot failures that are really happening during the analysis. Overall, a good level of agreement is observed between the 15 discov-

ered clusters and the 20 tickets. Specifically, the 7 *perfect matches* indicate cases whereby the reported message and the affected site coincide with the ones highlighted by the clusters. The 3 *fuzzy matches*, instead, refer to occasions whereby the agreement is less obvious, meaning that the cluster has evident connections with more than one ticket. Similarly, the 2 *partial matches* describe cases whereby either the message or the site coincide. The previous three statistics reveal that 12 out of the 15 suggested failures have led to fruitful investigations, thus implying a precision between 0.46 and 0.8 depending on the degree of nuisance one is willing to tolerate. Besides the above matches, 3 clusters highlight issues not reported on GGUS in the considered time window. These false positives indeed entail a futile effort for the operators and should be avoided, e.g. thwarting in-depth investigations if the temporal pattern is not escalating and/or the number of errors is not a concern. Nevertheless, in our case, posterior checks on the 3 false positives showed hints for real problems that went undetected or unreported by the operators, i.e. the error pattern seemed similar to other incidents opened to different sites.

For the second assessment, we investigate the relationship between clusters and tickets in the opposite direction, i.e. by looking at how many reported issues our approach captures. In this case, we consider a different baseline that provides a fairer detection performance evaluation. Indeed, it is reasonable to think that the failures observed during the analysis may be correlated to earlier tickets, thus justifying the adoption of a wide time window for the direct association. However, the same rationale does not necessarily apply when we reverse our perspective. In fact, there is no prior guarantee that a past ticket will generate new failures at a given moment in the future. Hence, considering all tickets undergoing investigations would potentially bias our measurement since specific past failures may not produce new malfunctions during the day of the analysis, thus resulting in untruthful false negatives. For this reason, in the case of the inverse association we limit our baseline to consider solely the tickets for which failures were really observed during the day of the analysis, thus reducing the initial 20 reports to only 9. Given this reference framework, the clusters successfully identify 8 out of 9 tickets, thus overlooking only a single issue.

To summarize, the previous results show that the approach presents promising perspectives given the complexity of the task and the completely unsupervised approach embraced. Although conducting an indisputable quantitative assessment is challenging – if not impossible with the available data –, the considerations expressed above furnish a reasonable proxy of the potential of our approach. Of course, a trade-off between the two perspectives is desirable in practice, for which more tuning is necessary with the help of shifters and site experts.

Chapter 11

Conclusions

The increasingly growing scale of modern computing infrastructures solicits more ingenious and automatic solutions to their management. This is particularly true concerning WLCG and the LHC experiments, whereby the upcoming upgrade will deliver ten times the actual volumes at a flat budget for infrastructure management.

Part II of this thesis discuss a data-driven pipeline to support DDM operations management for LHC experiments, with a particular focus on FTS transfer failures. The proposed approach consists of a pipeline that takes care of all the steps of a typical data science project, from the raw data to the final visualization. Also, some pre-production integration and testing has been made. In particular, the approach is already compatible – at least to some extent – with the production systems as it natively interacts with the raw data streams, and it complies with the timely execution requirements for online processing. In fact, the pipeline takes around 2.5/3 hours for one day of data, which is compatible with one or two applications per 8-hours shifts. This runtime is almost equally divided among the clustering stage – with a grid search for the optimization of k as described in Section 9.1.3 – and the post-processing/pre-aggregation needed for the visualization. Furthermore, no specific effort to optimize such runtimes was attempted, which suggests that some space for improvement is probably still available.

In terms of performance, our pipeline delivers promising results. The output clusters show an evident ability to capture both structural and semantic similarity between messages, as discussed in Section 10.1. Remarkably, this result is achieved despite applying minimal hard-coded feature engineering and exploiting simple baseline models for vectorization and clustering.

Interestingly, incorporating additional auxiliary information related to the source and destination hostnames seems to help unravel higher-level interactions between the nature of the issues and where they occur. This, in turn, provides a finer detail

when spotting problems that may aid the human operators to restore the proper functioning of the infrastructure faster.

The previous considerations are also corroborated by a quantitative assessment of the pipeline's potential impact when applied to daily workflows. This is done by comparing the outputs of our approach to the incidents reported in GGUS in a reasonable time window around the day of the analysis. In terms of the direct association between clusters and tickets, the performance varies from average to decent depending on how much nuisance one is willing to tolerate in the output. Regarding the inverse relationship, instead, the approach is close to perfection since it highlights 8 out of the 9 incidents observable on the day of the analysis.

Nonetheless, some adjustment and tuning would be helpful prior to full integration into production. First, the analyzed clusters show indications that additional tuning may be needed in some cases to guarantee a more suitable level of granularity. This task is highly application-specific and requires the direct involvement of shifters and site experts.

A second concern is related to the limited number of errors shown. Ideally, the perfect output for our use case would be one error pattern – or even a more human-readable description directly pointing to the source of the problem – per cluster for a small number of clusters (e.g. ≤ 6). In practice, however, the magnitude of the problem still refers to the actual number of failures. Even reducing it to the minimum, this is still bounded by the number of combinations between unique strings/patterns and source/destination locations, which is clearly overwhelming to handle for human operators. Therefore, the desired output is hardly deliverable as there is a trade-off between the clusters' internal homogeneity (number of patterns) and their number. For this reason, we reach a compromise by setting a higher value of k and displaying just a fixed portion of each cluster (3 patterns in the current implementation). However, limiting the visualized patterns potentially hinders serious faults of medium and small sizes. Moreover, the necessity to mask message parameters to get more informative and abstract descriptions prevents using their values for troubleshooting – e.g. when the failures are due to specific parameter values. In order to comply with the above requirements, a possible solution is a flexible and efficient implementation that allows the shifters to adjust the number of displayed patterns and enables interactive drill-down to investigate more closely the effect of parametric values. Nevertheless, guaranteeing a good balance constitutes an intrinsic challenge of our use case, and its resolution again requires a direct tuning by experts.

Furthermore, although it makes sense to cross-check clustering results with GGUS tickets for a quantitative evaluation, this comparison has drawbacks. On one side, GGUS incidents force to focus solely on reported failures, thus preventing the study of undetected issues and masking some omission policies due to external

factors – e.g. the site is in downtime or blacklisted, or the fault is known to be transient and therefore not reported. On the other side, the procedure is sensitive to the choice of the time window. Indeed the issues may have no match because they are reported before the selected period or due to delays in their discovery and reporting. All in all, the final assessment may result biased because of these factors, thus limiting the reaches of the drawn conclusions.

All of the previous adjustments demand additional in-depth studies, each requiring a lengthy manual review of the results due to the unsupervised approach. Also, most of the above solicit direct participation of system experts to guarantee the soundness of the results and proper tuning. Considering the several appointed investigations and the conspicuous number of alternative combinations, it becomes clear how the requested effort is not affordable and does not scale to the comparison of adversarial approaches. A possible solution we envision for future developments is represented by the collection of a reference dataset where to store labels for error categories, root causes, incident priority and solving actions. In this way, the evaluation of new experiments would become immediate and systematic. Also, this would make the investigation of novel techniques sustainable, enlarging the plethora of applicable approaches to supervised methods and enabling a coherent comparison of alternative algorithms. Perhaps more importantly, the derived measure of performance would be linked to the actual goal of the analysis, thus allowing a direct optimization of the models for the specific task of interest.

References

Aad, G., Abat, E., Abdallah, J., Abdelalim, A., Abdesselam, A., Abi, B., ... others (2020, apr). ATLAS data quality operations and performance for 2015–2018 data-taking. *Journal of Instrumentation*, *15*(04), P04003–P04003. Retrieved from https://doi.org/10.1088/1748-0221/15/04/p04003 doi: 10.1088/1748-0221/15/04/p04003

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from http://tensorflow.org/ (Software available from tensorflow.org)

Aberle, O., Béjar Alonso, I., Brüning, O., Fessia, P., Rossi, L., et al. (2020). *High-Luminosity Large Hadron Collider (HL-LHC): Technical design report.* Geneva: CERN. Retrieved from https://cds.cern.ch/record/2749422 doi: 10.23731/CYRM-2020-0010

Adobe. (2021). *The ultimate guide to facebook image sizes.* Author. Retrieved 2021-12-24, from https://www.adobe.com/express/discover/sizes/facebook (Blog post)

Alam, H. B., Pusateri, A. E., Kindzelski, A., Egan, D., Hoots, K., Andrews, M. T., ... others (2012). Hypothermia and hemostasis in severe trauma: a new crossroads workshop report. *Journal of Trauma and Acute Care Surgery*, *73*(4), 809–817.

Albalawi, R., Yeap, T. H., & Benyoucef, M. (2020). Using topic modeling methods for short-text data: A comparative analysis. *Frontiers in Artificial Intelligence*, *3*. Retrieved from https://www.frontiersin.org/article/10.3389/frai.2020.00042 doi: 10.3389/frai.2020.00042

Albrecht, J., Alves, A. A., Amadio, G., Andronico, G., Anh-Ky, N., Aphecetche,

L., ... others (2019). A roadmap for hep software and computing r&d for the 2020s. *Computing and software for big science*, *3*(1), 1–49.

ALICE Collaboration. (2008). The alice experiment at the cern lhc. *Journal of Instrumentation*, *3*(08), S08002.

Alzubaidi, L., Al-Amidie, M., Al-Asadi, A., Humaidi, A. J., Al-Shamma, O., Fadhel, M. A., ... Duan, Y. (2021). Novel transfer learning approach for medical imaging with limited labeled data. *Cancers*, *13*(7), 1590.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., & Sander, J. (1999, jun). Optics: Ordering points to identify the clustering structure. *SIGMOD Rec.*, *28*(2), 49–60. Retrieved from https://doi.org/10.1145/304181.304187 doi: 10.1145/304181.304187

Antoni, T., Bühler, W., Dres, H., Grein, G., & Roth, M. (2008, jul). Global grid user support—building a worldwide distributed user support infrastructure. *Journal of Physics: Conference Series*, *119*(5), 052002. Retrieved from https://doi.org/10.1088/1742-6596/119/5/052002 doi: 10.1088/1742-6596/119/5/052002

Arteta, C., Lempitsky, V., & Zisserman, A. (2016, 10). Counting in the wild. In (Vol. 9911, p. 483-498). doi: 10.1007/978-3-319-46478-7_30

Arthur, D., & Vassilvitskii, S. (2007). K-means++: the advantages of careful seeding. In *In proceedings of the 18th annual acm-siam symposium on discrete algorithms.*

Ashton, K., et al. (2009). That 'internet of things' thing. *RFID journal*, *22*(7), 97–114.

ATLAS Collaboration. (2008). The atlas experiment at the cern large hadron collider. *Journal of instrumentation*, *3*, S08003.

Baker, I. (2014, October). *Why are email files so large?* lifewire. Retrieved 2021-12-24, from https://medium.com/@raindrift/how-big-is-email-305bbdb69776 (Blog post)

Balazka, D., & Rodighiero, D. (2020). Big data and the little big bang: an epistemological (r) evolution. *Frontiers in big Data*, *3*, 31.

Barisits, M., Beermann, T., Berghaus, F., Bockelman, B., Bogado, J., Cameron, D., ... others (2019). Rucio: Scientific data management. *Computing and Software for Big Science*, *3*(1), 1–19.

Barr, J. (2021, March). *Celebrate 15 years of amazon s3 with 'pi week' livestream events.* Amazon blog. Retrieved 2021-12-24, from https://aws.amazon.com/blogs/aws/amazon-s3s-15th-birthday-it-is-still-day-1-after-5475-days-100-trillion-objects/ (Blog post)

Bellamy, R., Safar, P., Tisherman, S. A., Basford, R., Bruttig, S. P., Capone, A., ... others (1996). Suspended animation for delayed resuscitation. *Critical care medicine*, *24*(2), 24S–47S.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, *3*(Feb), 1137–1155.

Big data's fourth v. (2017, October). Spotless data blog. Retrieved 2021-12-18, from https://web.archive.org/web/20180731105912/https://spotlessdata.com/blog/big-datas-fourth-v (Blog post)

Bird, I. (2011). Computing for the large hadron collider. *Annual Review of Nuclear and Particle Science*, *61*, 99–118. Retrieved from https://doi.org/10.1146/annurev-nucl-102010-130059 doi: 10.1146/annurev-nucl-102010-130059

Bouma, H. R., Verhaag, E. M., Otis, J. P., Heldmaier, G., Swoap, S. J., Strijkstra, A. M., ... Carey, H. V. (2012). Induction of torpor: mimicking natural metabolic suppression for biomedical applications. *Journal of cellular physiology*, *227*(4), 1285–1290.

Bradford, J., Schaffer, M., & Talk, D. (2020). Advancing towards human torpor capabilities in support of future space settlements. In *2020 ieee aerospace conference* (p. 1-16). doi: 10.1109/AERO47225.2020.9172507

Breiman, L. (2001). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, *16*(3), 199–231.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... others (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Calafiura, P., Catmore, J., Costanzo, D., & Di Girolamo, A. (2020, Sep). *Atlas hl-lhc computing conceptual design report* (Tech. Rep.). Geneva: CERN. Retrieved from https://cds.cern.ch/record/2729668

Cao, L. (2017). Data science: a comprehensive overview. *ACM Computing Surveys*

(CSUR), *50*(3), 1–42.

Cao, Y., Liu, S., Peng, Y., & Li, J. (2020). Denseunet: densely connected unet for electron microscopy image segmentation. *IET Image Processing*, *14*(12), 2682–2689.

CERN. (n.d.). *Large hadron collider.* CERN Website. Retrieved 2021-12-24, from https://home.cern/science/accelerators/large-hadron-collider

CERN. (2017, July). *What data to record?* Author. Retrieved 2022-01-05, from https://home.cern/science/computing/storage

Cerri, M., Hitrec, T., Luppi, M., & Amici, R. (2021). Be cool to be far: Exploiting hibernation for space exploration. *Neuroscience & Biobehavioral Reviews*, *128*, 218-232. Retrieved from https://www.sciencedirect.com/science/article/pii/S0149763421002645 doi: https://doi.org/10.1016/j.neubiorev.2021.03.037

Cerri, M., Negrini, M., & Zoccoli, A. (2021). Study of enhanced radio-resistance induced by hibernation. *Il nuovo cimento C*, *44*(4-5), 1–4.

Cerri, M., Tinganelli, W., Negrini, M., Helm, A., Scifoni, E., Tommasino, F., ... Durante, M. (2016). Hibernation for space travel: Impact on radioprotection. *Life Sciences in Space Research*, *11*, 1-9. Retrieved from https://www.sciencedirect.com/science/article/pii/S2214552416300542 doi: https://doi.org/10.1016/j.lssr.2016.09.001

Chambers, J. M. (1993). Greater or lesser statistics: a choice for future research. *Statistics and Computing*, *3*(4), 182–184.

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

Chollet, F., et al. (2015). *Keras.* https://keras.io.

Ciresan, D., Giusti, A., Gambardella, L. M., & Schmidhuber. (2012, 01). Deep neural networks segment neuronal membranes in electron microscopy images. *Proceedings of Neural Information Processing Systems*, *25*.

Ciresan, D., Giusti, A., Gambardella, L. M., & Schmidhuber, J. (2013, 09). Mitosis detection in breast cancer histology images with deep neural networks. In (Vol. 16, p. 411-8). doi: 10.1007/978-3-642-40763-5_51

Cleveland, W. S. (2001). Data science: an action plan for expanding the technical

areas of the field of statistics. *International statistical review*, *69*(1), 21–26.

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Clissa, L. (2022). Survey of big data sizes in 2021. *arXiv preprint arXiv:2202.07659*.

Clissa, L., Lassnig, M., & Rinaldi, L. (submitted). Analyzing file transfer errors through machine learning. *Computing for Big Sciences*. (under review)

Clissa, L., Morelli, R., Squarcio, F., Hitrec, T., Luppi, M., Rinaldi, L., ... Zoccoli, A. (2021). *Fluorescent neuronal cells.* Retrieved from http://amsacta.unibo.it/6706/

CMS Collaboration. (2008). The cms experiment at the cern lhc. *JInst*, *3*, S08004.

Cohen, J., Boucher, G., Glastonbury, C., Lo, H., & Bengio, Y. (2017, 10). Count-ception: Counting by fully convolutional redundant counting. In (p. 18-26). doi: 10.1109/ICCVW.2017.9

da Conceição, E. P. S., Morrison, S. F., Cano, G., Chiavetta, P., & Tupone, D. (2020). 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.

Davenport, T., & Patil, D. J. (2012, October). *Data scientist: The sexiest job of the 21st century.* Retrieved 2021-12-05, from https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century

Dean, B. (2021a, April). *Dropbox usage and revenue stats.* Backlinko. Retrieved 2021-12-24, from hhttps://backlinko.com/dropbox-users (Blog post)

Dean, B. (2021b, September). *How many people use youtube in 2021?* Backlinko. Retrieved 2021-12-24, from https://backlinko.com/youtube-users#youtube-statistics (Blog post)

Decker, L., Leite, D., Giommi, L., & Bonacorsi, D. (2020). Real-time anomaly detection in data centers for log-based predictive maintenance using an evolving fuzzy-rule-based approach. In *2020 ieee international conference on fuzzy systems (fuzz-ieee)* (p. 1-8). doi: 10.1109/FUZZ48607.2020.9177762

Decker, L., Leite, D., Viola, F., & Bonacorsi, D. (2020). Comparison of evolving granular classifiers applied to anomaly detection for predictive maintenance

in computing centers. In *2020 ieee conference on evolving and adaptive intelligent systems (eais)* (p. 1-8). doi: 10.1109/EAIS48028.2020.9122779

Dedić, N., & Stanier, C. (2016). Towards differentiating business intelligence, big data, data analytics and knowledge discovery. In *International conference on enterprise resource planning systems* (pp. 114–122).

De Kunder, M. (2021, December). *Daily estimated size of the world wide web.* WorldWideWebSize.com. Retrieved 2021-12-24, from https://www.worldwidewebsize.com/ (Website)

dell´Agnello, Luca, Boccali, Tommaso, Cesini, Daniele, Chiarelli, Lorenzo, Chierici, Andrea, Dal Pra, Stefano, ... Zani, Stefano (2019). Infn tier-1: a distributed site. *EPJ Web Conf.*, *214*, 08002. Retrieved from https://doi.org/10.1051/epjconf/201921408002 doi: 10.1051/epjconf/201921408002

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 ieee conference on computer vision and pattern recognition* (p. 248-255). doi: 10.1109/CVPR.2009.5206848

Dentico, D., Amici, R., Baracchi, F., Cerri, M., Del Sindaco, E., Luppi, M., ... Zamboni, G. (2009). C-fos expression in preoptic nuclei as a marker of sleep rebound in the rat. *European Journal of Neuroscience*, *30*(4), 651–661. doi: 10.1111/j.1460-9568.2009.06848.x

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dhar, V. (2013, dec). Data science and prediction. *Commun. ACM*, *56*(12), 64–73. Retrieved from https://doi.org/10.1145/2500499 doi: 10.1145/2500499

Dhingra, B., Zhou, Z., Fitzpatrick, D., Muehl, M., & Cohen, W. W. (2016). Tweet2vec: Character-based distributed representations for social media. *arXiv preprint arXiv:1605.03481*.

Di Girolamo, Alessandro, Legger, Federica, Paparrigopoulos, Panos, Klimentov, Alexei, Schovancová, Jaroslava, Kuznetsov, Valentin, ... Padolski, Sergey (2020). Operational intelligence for distributed computing systems for exas-

cale science. *EPJ Web Conf.*, *245*, 03017. Retrieved from https://doi.org/10.1051/epjconf/202024503017 doi: 10.1051/epjconf/202024503017

Di Girolamo, A., Legger, F., Paparrigopoulos, P., Schovancová, J., Beermann, T., Boehler, M., ... Tuckus, N. (2022). Preparing distributed computing operations for the hl-lhc era with operational intelligence. *Frontiers in Big Data*, *4*, 115. Retrieved from https://www.frontiersin.org/article/10.3389/fdata.2021.753409 doi: 10.3389/fdata.2021.753409

Diotalevi, T., Bonacorsi, D., Falabella, A., Giommi, L., Martelli, B., Michelotto, D., ... Rossi Tisbeni, S. (2019). Collection and harmonization of system logs and prototypal Analytics services with the Elastic (ELK) suite at the INFN-CNAF computing centre. In *Proceedings of international symposium on grids & clouds 2019 — pos(isgc2019)* (Vol. 351, p. 027). doi: 10.22323/1.351.0027

Djuraskovic, O. (2021, October). *Google search statistics and facts 2021 (you must know).* First Site Guide. Retrieved 2021-12-24, from https://firstsiteguide.com/google-search-stats/ (Blog post)

Domo. (2021, September). *Data never sleeps 9* (Tech. Rep.). Retrieved 2021-12-24, from https://www.businesswire.com/news/home/20210929005835/en/Domo-Releases-Ninth-Annual-%E2%80%9CData-Never-Sleeps%E2%80%9D-Infographic

Donoho, D. (2017). 50 years of data science. *Journal of Computational and Graphical Statistics*, *26*(4), 745–766.

Du, M., & Li, F. (2016). Spell: Streaming parsing of system event logs. In *2016 ieee 16th international conference on data mining (icdm)* (pp. 859–864).

Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd* (Vol. 96, pp. 226–231).

Faustino, G. M., Gattass, M., Rehen, S., & de Lucena, C. J. P. (2009). Automatic embryonic stem cells detection and counting method in fluorescence microscopy images. In *2009 ieee international symposium on biomedical imaging: From nano to macro* (p. 799-802). doi: 10.1109/ISBI.2009.5193170

Friedman, J., Hastie, T., Tibshirani, R., et al. (2009). *The elements of statistical*

learning (Vol. 1) (No. 10). Springer series in statistics New York. (Second Edition)

Fu, Q., Lou, J.-G., Wang, Y., & Li, J. (2009). Execution anomaly detection in distributed systems through unstructured log analysis. In *2009 ninth ieee international conference on data mining* (pp. 149–158).

Fukushima, K., & Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets* (pp. 267–285). Springer.

Gage, P. (1994). A new algorithm for data compression. *C Users Journal*, *12*(2), 23–38.

Ganslo\ßer, U., & Jann, G. (2019). Thermoregulation in animals: Some fundamentals of thermal biology. In B. Fath (Ed.), *Encyclopedia of ecology (second edition)* (Second Edition ed., p. 328-336). Oxford: Elsevier. Retrieved from https://www.sciencedirect.com/science/article/pii/B9780124095489111704 doi: https://doi.org/10.1016/B978-0-12-409548-9.11170-4

Gelman, A. (2013). Statistics is the least important part of data science. *Blog post on Statistical Modeling, Causal Inference, and Social Science*, *14*.

Gillis, R., Adams, G., Besong, D., Machova, E., Ebringerova, A., Harding, S., & Patel, T. (2016). Phosphorylated tau protein in the myenteric plexus of the ileum and colon of normothermic rats and during synthetic torpor. *European Biophysics Journal*.

Giommi, L., Bonacorsi, D., Diotalevi, T., Rinaldi, L., Morganti, L., Falabella, A., ... Tisbeni, S. (2019). Towards Predictive Maintenance with Machine Learning at the INFN-CNAF computing centre. In *Proceedings of international symposium on grids & clouds 2019 — pos(isgc2019)* (Vol. 351, p. 003). doi: 10.22323/1.351.0003

Giordano, Domenico, Paltenghi, Matteo, Metaj, Stiven, & Dvorak, Antonin. (2021). Anomaly detection in the cern cloud infrastructure. *EPJ Web Conf.*, *251*, 02011. Retrieved from https://doi.org/10.1051/epjconf/202125102011 doi: 10.1051/epjconf/202125102011

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. (http://www.deeplearningbook.org)

Grandi, C. (2017, June). *Computing for hep experiments.* CERN Indico. Retrieved 2021-12-24, from https://indico.cern.ch/event/605204/contributions/2440577/attachments/1471783/2277732/Calcolo-HEP-Perugia-20170605.pdf (Slides from invited talk)

Greenspan, H., van Ginneken, B., & Summers, R. M. (2016). 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. doi: 10.1109/TMI.2016.2553401

Grigorieva, M., & Grin, D. (2021, April). Clustering error messages produced by distributed computing infrastructure during the processing of high energy physics data. *International Journal of Modern Physics A*, *36*(10), 2150070-130. doi: 10.1142/S0217751X21500706

Grimes, S. (2013, August). *Big data: Avoid 'wanna v' confusion.* InformationWeek. Retrieved 2021-12-18, from https://www.informationweek.com/big-data-analytics/big-data-avoid-wanna-v-confusion

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

Guyon, I., Von Luxburg, U., & Williamson, R. C. (2009). Clustering: Science or art. In *Nips 2009 workshop on clustering theory* (pp. 1–11).

Hampton, M. (2021, June). *Why is aws s3 object count is measured iin units used to measure file size?* serverfault. Retrieved 2021-12-24, from https://www.ringingliberty.com/2021/06/30/why-is-aws-s3-object-count-is-measured-iin-units-used-to-measure-file-size/ (Website)

Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, *143*(1), 29–36.

Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., ... Larochelle, H. (2017). Brain tumor segmentation with deep neural networks. *Medical Image Analysis*, *35*, 18-31.

He, K., Zhang, X., Ren, S., & Sun, J. (2016, 10). Identity mappings in deep residual networks. In (Vol. 9908, p. 630-645). doi: 10.1007/978-3-319-46493-0_38

He, P., Zhu, J., Zheng, Z., & Lyu, M. R. (2017). Drain: An online log parsing approach with fixed depth tree. In *2017 ieee international conference on web services (icws)* (pp. 33–40).

Hilbert, M., & López, P. (2011). The world's technological capacity to store, communicate, and compute information. *science*, *332*(6025), 60–65.

Hitrec, T., Luppi, M., Bastianini, S., Squarcio, F., Berteotti, C., Martire, V. L., ... Cerri, M. (2019, oct). Neural control of fasting-induced torpor in mice. *Scientific Reports*, *9*(1). doi: 10.1038/s41598-019-51841-2

Hitrec, T., Squarcio, F., Cerri, M., Martelli, D., Occhinegro, A., Piscitiello, E., ... Luppi, M. (2021). Reversible tau phosphorylation induced by synthetic torpor in the spinal cord of the rat. *Frontiers in neuroanatomy*, *15*, 3.

Hochreiter, S. (1998). 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.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international acm sigir conference on research and development in information retrieval* (pp. 50–57).

Howard, J., & Gugger, S. (2020, Feb). Fastai: A layered api for deep learning. *Information*, *11*(2), 108. Retrieved from http://dx.doi.org/10.3390/info11020108 doi: 10.3390/info11020108

Huber, P. J. (2012). *Data analysis: What can be learned from the past 50 years* (Vol. 874). John Wiley & Sons.

Indig, K. (2020, July). *Google's index is smaller than we think - and might not grow at all.* kevin-indig.com. Retrieved 2021-12-24, from https://www.kevin-indig.com/googles-index-is-smaller-than-we-think-and-might-not-grow-at-all/ (Blog post)

INFN-CNAF Collaboration. (2019). *Annual report 2019* (Tech. Rep.). INFN-CNAF. Retrieved from https://www.cnaf.infn.it/wp-content/uploads/2020/05/cnaf-annual-report-2019.pdf

Jain, A. (2016, September). *The 5 v's of big data.* Watson Health Perspectives. Retrieved 2021-12-18, from https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data/ (Blog post)

Jiang, H., Ma, H., Qian, W., Gao, M., & Li, Y. (2018). An automatic detection sys-

tem of lung nodule based on multigroup patch-based deep learning network. *IEEE Journal of Biomedical and Health Informatics*, *22*(4), 1227-1237. doi: 10.1109/JBHI.2017.2725903

Jimenez-del Toro, O., Otálora Montenegro, J., Andersson, M., Euren, K., Hedlund, M., Rousson, M., ... Atzori, M. (2017, 01). Analysis of histopathology images. In (p. 281-314). doi: 10.1016/B978-0-12-812133-7.00010-7

Karavakis, E., Manzi, A., Rios, M. A., Keeble, O., Cabot, C. G., Simon, M., ... Angelogiannopoulos, A. (2020). Fts improvements for lhc run-3 and beyond. In *Epj web of conferences* (Vol. 245, p. 04016).

Kingma, D. P., & Ba, J. (2017). *Adam: A method for stochastic optimization.*

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems* (pp. 3294–3302).

Kitchin, R., & McArdle, G. (2016). What makes big data, big data? exploring the ontological characteristics of 26 datasets. *Big Data & Society*, *3*(1), 2053951716631130.

Korbar, B., Olofson, A., Miraflor, A., Nicka, K., Suriawinata, M., Torresani, L., ... Hassanpour, S. (2017, 03). Deep-learning for classification of colorectal polyps on whole-slide images. *Journal of Pathology Informatics*, *8*. doi: 10.4103/jpi.jpi_34_17

Kraus, O., Ba, J., & Frey, B. (2016, 06). Classifying and segmenting microscopy images with deep multiple instance learning. *Bioinformatics*, *32*, i52-i59. doi: 10.1093/bioinformatics/btw252

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012, 01). Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, *25*. doi: 10.1145/3065386

Kshemkalyani, A. D., & Singhal, M. (2011). *Distributed computing: principles, algorithms, and systems.* Cambridge University Press.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188–1196).

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., ... Shi, W. (2017, 07). Photo-realistic single image super-resolution using a generative adversarial network. In (p. 105-114). doi: 10.1109/CVPR.2017

.19

Leite, D., Decker, L., Santana, M., & Souza, P. (2020). Egfc: Evolving gaussian fuzzy classifier from never-ending semi-supervised data streams – with application to power quality disturbance detection and classification. In *2020 ieee international conference on fuzzy systems (fuzz-ieee)* (p. 1-9). doi: 10.1109/FUZZ48607.2020.9177847

Lempitsky, V., & Zisserman, A. (2010). Learning to count objects in images. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, & A. Culotta (Eds.), *Advances in neural information processing systems* (Vol. 23). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2010/file/fe73f687e5bc5280214e0486b273a5f9-Paper.pdf

LHCb Collaboration. (2008). The lhcb detector at the lhc. *Journal of instrumentation*, *3*(08), S08005.

Lichtman, J. W., & Conchello, J.-A. (2005). Fluorescence microscopy. *Nature methods*, *2*(12), 910–919.

Lin, Q., Zhang, H., Lou, J.-G., Zhang, Y., & Chen, X. (2016). Log clustering based problem identification for online service systems. In *2016 ieee/acm 38th international conference on software engineering companion (icse-c)* (pp. 102–111).

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., ... Dollár, P. (2015). *Microsoft coco: Common objects in context.*

Ling, W., Luís, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., ... Trancoso, I. (2015). Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, *28*(2), 129–137.

Lohr, S. (2013, February). *The origins of 'big data': An etymological detective story.* The New York Times. Retrieved 2021-12-18, from https://bits.blogs.nytimes.com/2013/02/01/the-origins-of-big-data-an-etymological-detective-story/ (Blog post)

Lomonaco, V., Pellegrini, L., Cossu, A., Carta, A., Graffieti, G., Hayes, T. L., ... Maltoni, D. (2021). *Avalanche: an end-to-end library for continual learning.*

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for

semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3431–3440).

Lundervold, A. S., & Lundervold, A. (2019). An overview of deep learning in medical imaging focusing on mri. *Zeitschrift fur Medizinische Physik*, *29*(2), 102–127. Retrieved from https://doi.org/10.1016/j.zemedi.2018.11.002 doi: 10.1016/j.zemedi.2018.11.002

Luppi, M., Cerri, M., Di Cristoforo, A., Hitrec, T., Dentico, D., Del Vecchio, F., ... Amici, R. (2019). c-fos expression in the limbic thalamus following thermoregulatory and wake–sleep changes in the rat. *Experimental Brain Research*, *237*(6), 1397–1407. Retrieved from http://dx.doi.org/10.1007/s00221-019-05521-2 doi: 10.1007/s00221-019-05521-2

Makanju, A. A., Zincir-Heywood, A. N., & Milios, E. E. (2009). Clustering event logs using iterative partitioning. In *Proceedings of the 15th acm sigkdd international conference on knowledge discovery and data mining* (pp. 1255–1264).

Mashey, J. R. (1998, April). *Big data ...and the next wave of infrastress.* USENIX. Retrieved 2021-12-18, from https://static.usenix.org/event/usenix99/invited_talks/mashey.pdf (Slides from invited talk)

Masin, L., Claes, M., Bergmans, S., Cools, L., Andries, L., Davis, B. M., ... De Groef, L. (2021). A novel retinal ganglion cell quantification tool based on deep learning. *Scientific reports*, *11*(1), 1–13.

Mason, S. J., & Graham, N. E. (2002). Areas beneath the relative operating characteristics (roc) and relative operating levels (rol) curves: Statistical significance and interpretation. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, *128*(584), 2145–2166.

McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). Learned in translation: Contextualized word vectors. *arXiv preprint arXiv:1708.00107*.

McInnes, L., Healy, J., & Astels, S. (2017). hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, *2*(11), 205.

Meraj, T., Rauf, H. T., Zahoor, S., Hassan, A., Lali, M. I., Ali, L., ... Shoaib, U. (2020). Lung nodules detection using semantic segmentation and classification with optimal features. *Neural Computing and Applications*, 1–14.

Merkus, H. G. (2009). *Particle size measurements: fundamentals, practice, quality* (Vol. 17). Springer Science & Business Media.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Minarini, F., & Decker, L. (2020). Time-series anomaly detection applied to log-based diagnostic system using unsupervised machine learning approach. In *Conference of open innovations association, fruct* (pp. 343–348).

Morelli, R., Clissa, L., Amici, R., Cerri, M., Hitrec, T., Luppi, M., ... Zoccoli, A. (2021). Automating cell counting in fluorescent microscopy through deep learning with c-ResUnet. *Scientific Reports*, *11*(1), 22920. doi: 10.1038/s41598-021-01929-5

Mullen Jr, J. F., Tanner, F. R., & Sallee, P. A. (2019). Comparing the effects of annotation type on machine learning detection performance. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition workshops* (pp. 0–0).

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Icml*.

Naur, P. (1974). *Concise survey of computer methods*. Petrocelli Books.

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems* (pp. 849–856).

Niu, L., Dai, X., Zhang, J., & Chen, J. (2015). Topic2vec: learning distributed representations of topics. In *2015 international conference on asian language processing (ialp)* (pp. 193–196).

Onay, C., & Öztürk, E. (2018). A review of credit scoring research in the age of big data. *Journal of Financial Regulation and Compliance*.

Papadimitriou, C. H., Raghavan, P., Tamaki, H., & Vempala, S. (2000). Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences*, *61*(2), 217–235.

Perry, N. (2021, June). *How much data does netflix use?* digitaltrends. Retrieved 2021-12-24, from https://www.digitaltrends.com/movies/how-much-data-does-netflix-use/ (Blog post)

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettle-

moyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Piparo, D., Tejedor, E., Mato, P., Mascetti, L., Moscicki, J., & Lamanna, M. (2018). Swan: A service for interactive analysis in the cloud. *Future Generation Computer Systems*, *78*, 1071–1078.

Priceonomics. (2015, oct). *What's the difference between data science and statistics?* Author. Retrieved 2022-01-29, from https://priceonomics.com/whats-the-difference-between-data-science-and/

Puspitasari, A., Cerri, M., Takahashi, A., Yoshida, Y., Hanamura, K., & Tinganelli, W. (2021). Hibernation as a tool for radiation protection in space exploration. *Life*, *11*(1), 54.

Qamar, S., Jin, H., Zheng, R., Ahmad, P., & Usama, M. (2020). A variant form of 3d-unet for infant brain segmentation. *Future Generation Computer Systems*, *108*, 613–623.

Rahnemoonfar, M., & Sheppard, C. (2017, Apr). Deep count: Fruit counting based on deep simulated learning. *Sensors*, *17*(4), 905. Retrieved from http://dx.doi.org/10.3390/s17040905 doi: 10.3390/s17040905

Raza, S. e. A., Cheung, L., Epstein, D., Pelengaris, S., Khan, M., & Rajpoot, N. (2017, 04). Mimo-net: A multi-input multi-output convolutional neural network for cell segmentation in fluorescence microscopy images. In (p. 337-340). doi: 10.1109/ISBI.2017.7950532

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016, 06). You only look once: Unified, real-time object detection. In (p. 779-788). doi: 10.1109/CVPR.2016.91

Riccio, D., Brancati, N., Frucci, M., & Gragnaniello, D. (2018, 03). A new unsupervised approach for segmenting and counting cells in high-throughput microscopy image sets. *IEEE Journal of Biomedical and Health Informatics*, *PP*, 1-1. doi: 10.1109/JBHI.2018.2817485

Ritch, M. D., Hannon, B. G., Read, A. T., Feola, A. J., Cull, G. A., Reynaud, J., ... Ethier, C. R. (2020). Axonet: A deep learning-based tool to count retinal ganglion cell axons. *Scientific reports*, *10*(1), 1–13.

Ronneberger, O., Fischer, P., & Brox, T. (2015, 10). U-net: Convolutional networks for biomedical image segmentation. In (Vol. 9351, p. 234-241). doi:

10.1007/978-3-319-24574-4_28

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*, 53-65. Retrieved from https://www.sciencedirect.com/science/article/pii/0377042787901257 doi: https://doi.org/10.1016/0377-0427(87)90125-7

Sagiroglu, S., & Sinanc, D. (2013). Big data: A review. In *2013 international conference on collaboration technologies and systems (cts)* (pp. 42–47).

Sahiner, B., Heang-Ping Chan, Petrick, N., Datong Wei, Helvie, M. A., Adler, D. D., & Goodsitt, M. M. (1996). 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. doi: 10.1109/42.538937

Satopaa, V., Albrecht, J., Irwin, D., & Raghavan, B. (2011). Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops* (p. 166-171). doi: 10.1109/ICDCSW.2011.20

Schovancová, J. (2019, April). *Atlas computing operations* (Tech. Rep.). Retrieved 2022-01-10, from https://indico.cern.ch/event/809227/contributions/3370897/attachments/1820938/2978308/20190401-ATLAS_Computing_Operation_partial_view.pdf

Segui, S., Pujol, O., & Vitria, J. (2015, jun). Learning to count with deep object features. In *2015 ieee conference on computer vision and pattern recognition workshops (cvprw)* (p. 90-96). Los Alamitos, CA, USA: IEEE Computer Society. Retrieved from https://doi.ieeecomputersociety.org/10.1109/CVPRW.2015.7301276 doi: 10.1109/CVPRW.2015.7301276

Sennrich, R., Haddow, B., & Birch, A. (2016, August). Neural machine translation of rare words with subword units. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 1715–1725). Berlin, Germany: Association for Computational Linguistics. Retrieved from https://aclanthology.org/P16-1162 doi: 10.18653/v1/P16-1162

Silver, N. (2013). *What i need from statisticians.* Stats & Data Science View.

Retrieved 2022-01-29, from https://www.statisticsviews.com/article/nate-silver-what-i-need-from-statisticians/

Simard, P., Steinkraus, D., & Platt, J. (2003, 01). Best practices for convolutional neural networks applied to visual document analysis. In (p. 958-962). doi: 10.1109/ICDAR.2003.1227801

Snijders, C., Matzat, U., & Reips, U.-D. (2012). " big data": big gaps of knowledge in the field of internet science. *International journal of internet science*, *7*(1), 1–5.

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

Solé, M., Muntés-Mulero, V., Rana, A. I., & Estrada, G. (2017). Survey on models and techniques for root-cause analysis. *arXiv preprint arXiv:1701.08546*.

Statista, R. D. (2021, October). *Average daily spam volume worldwide from october 2020 to september 2021.* Statista. Retrieved 2021-12-24, from https://www.statista.com/statistics/1270424/daily-spam-volume-global/

Stokes, G. G. (2010). *Memoir and scientific correspondence of the late sir george gabriel stokes, bart.: Selected and arranged by joseph larmor* (Vol. 2). Cambridge University Press.

Su, Y., Li, D., & Chen, X. (2021). Lung nodule detection based on faster r-cnn framework. *Computer Methods and Programs in Biomedicine*, *200*, 105866.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.* MIT press. (Second Edition)

Teague, J., Karamalegos, S., Rebecca, H., Peck, J., & Pollard, B. (2021, July). *Web almanac* (Tech. Rep.). Retrieved 2021-12-24, from https://almanac.httparchive.org/en/2021/page-weight

Tinganelli, W., Hitrec, T., Romani, F., Simoniello, P., Squarcio, F., Stanzani, A., ... others (2019). Hibernation and radioprotection: gene expression in the liver and testicle of rats irradiated under synthetic torpor. *International journal of molecular sciences*, *20*(2), 352.

Tisbeni, S. R. (2019). *Big data analytics towards predictive maintenance at the infn-cnaf computing centre* (Doctoral dissertation). Retrieved from http://amslaurea.unibo.it/18430/

Tkachenko, M., Malyuk, M., Shevchenko, N., Holmanyuk, A., & Liubimov, N. (2020-2021). *Label Studio: Data labeling software.* Retrieved from https://github.com/heartexlabs/label-studio (Open source software available from https://github.com/heartexlabs/label-studio)

Todd, B., Ponce, L., Apollonio, A., & Walsh, D. J. (2018, Dec). *Lhc availability 2018: Proton physics* (Tech. Rep.). Retrieved from https://cds.cern.ch/record/2650574

Tschabitscher, H. (2021, June). *Why are email files so large?* lifewire. Retrieved 2021-12-24, from https://www.lifewire.com/what-is-the-average-size-of-an-email-message-1171208 (Blog post)

Tukey, J. W. (1962). The future of data analysis. *The annals of mathematical statistics*, *33*(1), 1–67.

Uddin, M. F., Gupta, N., et al. (2014). Seven v's of big data understanding big data to extract value. In *Proceedings of the 2014 zone 1 conference of the american society for engineering education* (pp. 1–5).

Vaarandi, R. (2003). A data clustering algorithm for mining patterns from event logs. In *Proceedings of the 3rd ieee workshop on ip operations & management (ipom 2003)(ieee cat. no. 03ex764)* (pp. 119–126).

Vaarandi, R., & Pihelgas, M. (2015). Logcluster-a data clustering and pattern mining algorithm for event logs. In *2015 11th international conference on network and service management (cnsm)* (pp. 1–7).

Vandenberghe, M., Scott, M., Scorer, P., Soderberg, M., Balcerzak, D., & Barker, C. (2017, 05). Relevance of deep learning to facilitate the diagnosis of her2 status in breast cancer open. *Scientific Reports*, *7*. doi: 10.1038/srep45938

Van den Bosch, A., Bogers, T., & De Kunder, M. (2016). Estimating search engine index size variability: a 9-year longitudinal study. *Scientometrics*, *107*(2), 839–856.

van Dyk, D., Fuentes, M., Jordan, M. I., Newton, M., Ray, B. K., Lang, D. T., & Wickham, H. (2015, October). *Asa statement on the role of statistics in data science.* Retrieved 2021-12-05, from https://magazine.amstat.org/blog/2015/10/01/asa-statement-on-the-role-of-statistics-in-data-science/

Vera, P., James, M., & Dan, S. (2019, August). *What is the file size of a one hour*

youtube video? Quora. Retrieved 2021-12-24, from https://www.quora.com/What-is-the-file-size-of-a-one-hour-YouTube-video

Vijayanarasimhan, S., & Grauman, K. (2009). What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *2009 ieee conference on computer vision and pattern recognition* (p. 2262-2269). doi: 10.1109/CVPR.2009.5206705

Von Luxburg, U., Williamson, R. C., & Guyon, I. (2012). Clustering: Science or art? In *Proceedings of icml workshop on unsupervised and transfer learning* (pp. 65–79).

Wang, L., Von Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J., & Fu, C. (2010). Cloud computing: a perspective study. *New generation computing*, *28*(2), 137–146.

Weng, J. J., Ahuja, N., & Huang, T. S. (1993). Learning recognition and segmentation of 3-d objects from 2-d images. In *1993 (4th) international conference on computer vision* (pp. 121–128).

Withers, P., & Cooper, C. E. (2019). Dormancy. In B. Fath (Ed.), *Encyclopedia of ecology (second edition)* (Second Edition ed., p. 309-314). Oxford: Elsevier. Retrieved from https://www.sciencedirect.com/science/article/pii/B9780124095489111674 doi: https://doi.org/10.1016/B978-0-12-409548-9.11167-4

WLCG. (2019). *Lhc run 2 (2014-2018).* Author. Retrieved 2022-01-07, from https://wlcg-public.web.cern.ch/about/

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, *2*(1-3), 37–52.

Wolf, S., Schott, L., Kothe, U., & Hamprecht, F. (2017). Learned watershed: End-to-end learning of seeded segmentation. In *Proceedings of the ieee international conference on computer vision* (pp. 2011–2019).

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67–82.

Wu, C. F. J. (1997). *Statistics = data science?* Georgia Tech. Retrieved 2022-01-29, from www2.isye.gatech.edu/~jeffwu/presentations/datascience.pdf (Slides from inauguration as Carver Professor of Statistics at University of Michigan)

Xie, J., Kiefel, M., Sun, M.-T., & Geiger, A. (2016, June). Semantic instance annotation of street scenes by 3d to 2d label transfer. In *Proceedings of the ieee conference on computer vision and pattern recognition (cvpr)*.

Xie, W., Noble, J. A., & Zisserman, A. (2018). Microscopy cell counting and detection with fully convolutional regression networks. *Computer methods in biomechanics and biomedical engineering: Imaging & Visualization*, *6*(3), 283–292.

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

Yang, B., Fu, X., Sidiropoulos, N. D., & Hong, M. (2017). Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning* (pp. 3861–3870).

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., Stoica, I., et al. (2010). Spark: Cluster computing with working sets. *HotCloud*, *10*(10-10), 95.

Zeiler, M. D., Krishnan, D., Taylor, G. W., & Fergus, R. (2010). Deconvolutional networks. In *2010 ieee computer society conference on computer vision and pattern recognition* (pp. 2528–2535).

Zhang, T., Ramakrishnan, R., & Livny, M. (1996, jun). Birch: An efficient data clustering method for very large databases. *SIGMOD Rec.*, *25*(2), 103–114. Retrieved from https://doi.org/10.1145/235968.233324 doi: 10.1145/235968.233324

Zhang, Z., Liu, Q., & Wang, Y. (2018). Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, *15*, 749-753.

Zhang, Z., & Zhang, D. (2021). What is data science? an operational definition based on text mining of data science curricula. *Journal of Behavioral Data Science*, *1*(1), 1–16.

Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., & Lyu, M. R. (2019). Tools and benchmarks for automated log parsing. In *2019 ieee/acm 41st international conference on software engineering: Software engineering in practice (icse-seip)* (pp. 121–130).