

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *08/11/2023* **par :**

Kévin CORTACERO

Cartesian Genetic Programming for Biomedical Image Segmentation

---

JURY

Dr. Una-May O'REILLY	Rapporteur
Prof. Cédric WEMMERT	Rapporteur
Prof. Nicolas GAUDENZIO	Examinateur
Prof. Wolfgang BANZHAF	Examinateur
Prof. Sylvain CUSSAT-BLANC	Directeur de Thèse
Prof. Salvatore VALITUTTI	Directeur de Thèse

École doctorale et spécialité :
BSB : Immunologie

Unité de Recherche :
Centre de Recherches en Cancérologie de Toulouse (UMR 1037)

Directeur(s) de Thèse :
Prof. Sylvain CUSSAT-BLANC et *Prof. Salvatore VALITUTTI*

Rapporteurs :
Dr. Una-May O'REILLY et *Prof. Cédric WEMMERT*

Acknowledgments

First of all, I would like to thank the people who made this project possible. Starting with my two supervisors, Salvatore and Sylvain, whom I trusted and who learned to trust me throughout my thesis. Thank you for your support, your advice and your availability. I enjoyed being supervised by both of you. I would also like to thank the members (past and present) of my two research teams. With a special mention for Brienne, who manages to understand and beautify the work we made and read multiple times this thesis. It is a collaboration I immensely enjoyed and which I think will continue. This thesis also benefited from the insight of Dr. Jean-Philippe Girard and Prof. Wolfgang Banzhaf, whom I thank for their help and advice. And a big thank you to Wolfgang for spending time trying to discuss my ideas and meditate around GP, Life and Time. Finally, I would like to thank my referees Cédric Wemmert and Una-May O'Reilly who agreed to read and review my work.

I would also like to take this opportunity to thank once again all my former masters and mentors: Etienne (IRAP), Gilles (IRIT), Yiannis and Tobi (PRL), Rachid and Yoan (LAAS). There also a bit of you in this thesis.

And above and beyond my work, I would like to thank my immediate family most of all. Maman, Papa, et Vincent, Marjorie, merci pour toutes les valeurs que vous avez su me transmettre et de toute la bonne énergie qui me suit et m'anime depuis 27 ans. Egalement merci Aubin, Swan et Angèle, vous avez la vie devant vous et je vous souhaite plein de bonheur et de réussite dans tous vos projets. Enfin, Justine, merci absolument pour tout ce que tu apportes à ma vie, de ton amour et ton soutien quelles que soient les épreuves. Tu es ma force, ma motivation, je suis très fier de tout ce que tu entreprends, ne change jamais ♡.

I dedicate this thesis manuscript to Julian Francis Miller, also known as the “Father of Cartesian Genetic Programming”, who sadly passed away last year. Although I never had the privilege to meet him in person, I hope to honor Julian by ensuring a bright future for his work and visionary conception of CGP.

Wherever you may be, may this work and upcoming developments make you proud of the legacy you left to us...

Farewell, dear colleague

Kévin Cortacero
Toulouse, France, September 2023

Résumé

Au cours de la dernière décennie, l'Intelligence Artificielle (IA), et en particulier l'Apprentissage Profond (DL pour "Deep Learning"), a connu une croissance exponentielle. Bien que l'accent ait principalement été mis sur l'amélioration de la précision des modèles, l'explicabilité, la frugalité et la réduction de la taille des jeux de données sont des défis essentiels qui doivent être abordés avec une urgence croissante. Bien que la précision demeure primordiale, cette thèse propose un rééquilibrage de ces critères, dans le but d'atteindre une performance de modèle solide dans un contexte de transparence totale et d'apprentissage avec peu d'annotaions.

Cette thèse se concentre sur le domaine de l'imagerie biomédicale et la segmentation des entités biologiques à l'aide de la vision par ordinateur. Nous introduisons ici *Kartezio*, une approche basée sur la Programmation Génétique Cartésienne (CGP) qui produit des algorithmes de traitement d'image transparents et interprétables grâce à l'assemblage et la paramétrisation itérative des fonctions. Les pipelines ainsi créés démontrent une performance comparable aux approches DL pour les tâches de segmentation d'image, avec moins de données d'entrainement. Cette méthode frugale d'apprentissage confère une grande flexibilité, comme démontré à travers une série de cas d'étude. De plus, cette thèse explore plusieurs approches pour extraire des informations significatives de la structure des algorithmes générés, offrant des stratégies pour diriger l'optimisation de l'algorithme à l'avenir. Enfin, cette étude étend le travail publié à un nombre illimité de types pouvant être intégrés au sein d'un même modèle multimodal (MAGE).

Grâce à cette exploration détaillée, ce travail ne présente pas seulement une méthodologie, mais propose également une alternative ou un complément aux paradigme dominant du DL qualifié de "boîte noire". Cela suggère que, malgré la complexité des solutions de pointe, nous avons peut-être précédemment sous-estimé des technologies frugales et transparentes telles que CGP, en particulier dans le domaine du traitement d'image. Dans l'imagerie biomédicale, où les prédictions des modèles peuvent avoir des implications profondes, l'idée d'utiliser des techniques modernes d'IA ne devrait pas être mise de côté en raison de contraintes de calcul ou de données, ni d'un manque de transparence. En présentant les modèles CGP à travers plusieurs applications pratiques, cette thèse plaide en faveur de l'adoption de l'Intelligence Artificielle Explicable (XAI) dans le domaine biomédical.

Mots-clés : Intelligence Artificielle Explicable, Apprentissage Automatique, Programmation Génétique Cartésienne, Traitement d'Image, Segmentation d'Instances.

Abstract

Over the past decade, Artificial Intelligence (AI), and in particular Deep Learning (DL), has experienced exponential growth. Although the main focus has been on improving the accuracy of models, explainability, frugality and dataset size reduction are fundamental challenges that need to be addressed with increasing urgency. While precision remains paramount, this thesis research proposes a rebalancing of these metrics, in order to achieve strong model performance in the context of full transparency and Few-Shot learning.

This thesis focuses upon the field of biomedical imaging and the segmentation of biological entities using computer vision. Herein, we introduce *Kartezio*, a modular Cartesian Genetic Programming (CGP)-based computational strategy that produces transparent and interpretable image processing algorithms through the iterative assembly and parameterization of functions. The pipelines thus created demonstrate comparable performance to state-of-the-art DL approaches on image segmentation tasks, while requiring drastically smaller training datasets. This Few-Shot Learning and frugal method confers tremendous flexibility, as demonstrated herein through a series of biomedical Use Cases. In addition, this thesis explores several approaches to extract meaningful information from the structure of the generated algorithms, offering strategies to direct optimization of the algorithm in the future. Finally, this study proposes extensions of the published work wherein unlimited types of data can be integrated into a single multimodal evolutionary approach (MAGE).

Through this detailed exploration, this work not only introduces a methodology but also offers an alternative or complement to dominant "black-box" DL paradigms. This suggests that, despite the complexity of state-of-the-art DL solutions, we may have previously underestimated frugal and transparent strategies such as CGP, particularly in the field of Image Processing. In biomedical imaging, where model predictions can have profound implications, the idea of using modern AI techniques should not be sidelined either due to computational or data constraints, or a lack of transparency. By showcasing CGP models through several practical applications, this thesis argues for the adoption of eXplainable Artificial Intelligence (XAI) in the biomedical domain.

Keywords: Explainable Artificial Intelligence, Machine Learning, Cartesian Genetic Programming, Image Processing, Instance Segmentation

List of publications

This thesis consists of a major article "Kartezio: A Darwinian Designer of Explainable Algorithms for Biomedical Image Segmentation" (**I.**), published in ***Nature Communications*** journal. This work is presented in **Chapter 3** and has been patented in collaboration with Inserm Transfert ("K. Cortacero et al. filed patent; EP 22307041.8").
The initial proof-of-concept of this project was published in ***Science Advances*** (**II.**), in which the Image Processing pipelines were adapted using Cartesian Genetic Programming.
Finally, at the end of this project, I contributed to the next developments of Kartezio with Yuri Lavinas by Lexicase Selection integration (**III.**).
As an earlier scientific work, I had the chance to publish a part of my Master 1 work on eye blinking detection with Tobias Fischer (**IV.**).

I. Cortacero, K. *et al.* Evolutionary design of explainable algorithms for biomedical image segmentation. *Nature communications* **14,** 7112 (2023)

II. Filali, L. *et al.* Ultrarapid lytic granule release from CTLs activates Ca2+-dependent synaptic resistance pathways in melanoma cells. *Science Advances* **8,** eabk3234 (2022)

III. Lavinas, Y. *et al. Evolving Graphs with Cartesian Genetic Programming with Lexicase Selection* in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (2023), 1920–1924

IV. Cortacero, K. *et al. RT-BENE: A dataset and baselines for real-time blink estimation in natural environments* in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops* (2019)

Contents

1 Introduction **21**
- 1.1 Thesis context . . . 21
  - 1.1.1 Oncology and Immunology . . . 23
  - 1.1.2 Computer Vision and Image Processing . . . 24
  - 1.1.3 Evolutionary Computation and Genetic Programming . 25
- 1.2 Main challenges . . . 26
  - 1.2.1 Image Data Diversity . . . 27
  - 1.2.2 Few Annotated Images . . . 28
  - 1.2.3 Frugality and low requirements . . . 28
  - 1.2.4 Explainable Artificial Intelligence . . . 29

2 Background **31**
- 2.1 Immunological Context: The Battlefield . . . 32
  - 2.1.1 The innate immune system . . . 33
  - 2.1.2 The adaptive immune system . . . 35
- 2.2 Fundamentals of Image Processing . . . 40
  - 2.2.1 Arithmetic Operators for Images . . . 41
  - 2.2.2 Logical Operators for Images . . . 44
  - 2.2.3 Convolution filter . . . 44
  - 2.2.4 Blurring and Sharpening . . . 46
  - 2.2.5 Derivative and Edge Detection . . . 48
  - 2.2.6 Mathematical Morphology . . . 55
  - 2.2.7 Features Descriptors . . . 61
- 2.3 Deep Learning . . . 64
  - 2.3.1 Convolutional Neural Network Model . . . 66
  - 2.3.2 Fully Convolutional Network . . . 67
  - 2.3.3 Residual Neural Network Model (ResNet) . . . 70
  - 2.3.4 Object Detection Algorithms . . . 73
  - 2.3.5 Embedding Models . . . 75
  - 2.3.6 Sequential Models . . . 76
  - 2.3.7 Transformers . . . 77

2.3.8 Biomedical Segmentation Models 79
2.3.9 Limitations 82
2.4 Objectives and direction of the thesis 83

3 Kartezio: A Darwinian Designer of Explainable Algorithms for Biomedical Image Segmentation 87
3.1 Genesis of the project 88
3.2 Cartesian Genetic Programming 92
3.2.1 Evolutionary Computation 92
3.2.2 Genetic Programming 93
3.2.3 Cartesian Genetic Programming Fundamentals 95
3.2.4 Cartesian Genetic Programming Models 100
3.2.5 Cartesian Genetic Programming for Image Processing . 102
3.3 Main Text (as submitted) 104
3.3.1 Abstract 105
3.3.2 Introduction 106
3.3.3 Results 109
3.3.4 Discussion 133
3.3.5 Material and Methods 143
3.4 Extended Data 163
3.4.1 Extended Data Figure 163
3.4.2 Extended Data Table 169
3.4.3 Extended Data Movie 169
3.5 Conclusion 173

4 Genotype Introspection 175
4.1 Hyperparameters Study 177
4.1.1 Observations and Perspectives 177
4.2 Function Distribution 178
4.2.1 Function Histogram 181
4.2.2 Bi-grams 184
4.2.3 Function location 184
4.3 Preliminary conclusion 186

5 Enhancing Kartezio: Multimodal Adaptive Graph Evolution 189
5.1 Multimodality for Cartesian Genetic Programming 190
5.1.1 Introduction and Context 190
5.1.2 Abstract 193
5.1.3 Introduction 193
5.1.4 Background 196
5.1.5 MAGE 201

5.1.6 Multimodal function synthesis . . . 207
5.1.7 Medical Image Segmentation . . . 213
5.2 Conclusion . . . 216

6 Discussion and Conclusion **221**
6.1 Perspectives for Computer Science . . . 223
6.1.1 Post-Evolution recombination research . . . 224
6.1.2 Evolution of Convolution kernels . . . 230
6.1.3 Towards If-Then-Else statements for CGP . . . 231
6.1.4 Cartesian Genetic Programming for Neural Architecture Search . . . 231
6.1.5 Cartesian Genetic Programming for generalist Deep Learning models . . . 232
6.1.6 Specialist Artificial Neural Networks as primitives . . . 233
6.1.7 Evolution as a Chess Game . . . 233
6.1.8 Genotype Correction . . . 236
6.2 The impact of Kartezio on Biomedical Image Analysis . . . 237
6.2.1 Explainable AI . . . 237
6.2.2 Few-Shot Learning . . . 240
6.2.3 Customization . . . 241
6.2.4 Versatility . . . 241
6.2.5 Frugality . . . 242
6.3 Conclusion . . . 243

References **245**

List of Figures

1.1 The context of this interdisciplinary Thesis 22
1.2 Learning Performance Versus Explainability Trade-Off 29
1.3 Evolution and Adoption of XAI search over years 30

2.1 Simplified T cell response against Tumor cells 39
2.2 An Immunofluorescence image, composed of 3 channels 42
2.3 Basic Arithmetic Operations and the Overflow effect 43
2.4 More advanced Arithmetic Operations 43
2.5 Basic Logical filters 45
2.6 Image Convolution Illustration 45
2.7 Blurs and Sharpen Operations 48
2.8 Edge Detection (1) and Gradient 51
2.9 Edge Detection (2) 54
2.10 Erode and Dilate filters 57
2.11 Open and Close filters 59
2.12 Watershed Transform Illustration 61
2.13 The Hough Circle Transform 62
2.14 CNN Standard Model: Convolution+ReLU, and Pooling . . . 68
2.15 CNN Features Visualization 68
2.16 Fully Convolutional Network Model 69
2.17 U-net Model Architecture 70
2.18 ResNet Model blocks 72
2.19 Mask Region-CNN Model Architecture 74
2.20 llustration of LSTM and GRU 78
2.21 Unfolded architecture of bidirectional LSTM with 3 consecutive steps 78

3.1 Comparison of GP program structure 95
3.2 Article Figure 1 | Architecture of a Kartezio-generated model: a fully explainable and transparent algorithm for Instance Segmentation 110

3.3 Article Figure 2 | Kartezio-generated models achieve comparable accuracy as state-of-the-art Deep Learning approaches with fewer training requirements . . . 111
3.4 Segmentation of tumor nodules within IHC-stained tissue specimens . . . 119
3.5 Article Figure 4 | Analysis of extracellular particles using Kartezio-mediated instance segmentation . . . 123
3.6 Article Figure 5 | 3D analysis of lytic molecule content in individual CTLs within a polyclonal population . . . 127
3.7 Article Figure 6 | A hybrid Kartezio/AI approach to detect CTL/target cell conjugates . . . 130
3.8 Extended Data Figure 1 | Conceptual summary of Kartezio . . 164
3.9 Extended Data Figure 2 | Illustrative images from each Use Case . . . 166
3.10 Extended Data Figure 3 | Automatically generated Python classes by Kartezio from given genotype . . . 167
3.11 Extended Data Figure 4 | Tumor nodule datasets and heatmap threshold analysis . . . 169

4.1 Visualization of Kartezio-generated graph . . . 176
4.2 Fitness curve along Evolution . . . 176
4.3 Impact of Four Common Hyperparameters on Kartezio . . . 179
4.4 Tendency of one metric against another . . . 179
4.5 Relative AP50 visualization with 2 fixed parameters . . . 180
4.6 Relative Graph Length visualization with 2 fixed parameters . 180
4.7 Histograms showing function frequency among active graphs . 182
4.8 Function histogram evolution over iterations . . . 183
4.9 Function bi-grams visualization . . . 185
4.10 Function position in partitioned Genotype . . . 187

5.1 Multimodal Adaptive Graph Evolution illustrative model . . . 194
5.2 Comparison of the convergence of MAGE and MT-CGP on the PSB2 suite. Fitness is based on error (lower is better). Dashed lines indicate training data; solid lines indicate test data. . . . 211
5.3 An example of instance segmentation, with the labels provided by CGP and MAGE. . . . 214
5.4 An example of evolved mono-typed CGP graph for Instance Segmentation . . . 217
5.5 An example of evolved MAGE graph for Instance Segmentation 218

6.1 Illustrative example of Post-Recombination between 2 genotypes 225
6.2 Improvement after Post-Training recombination 229
6.3 The decision-tree of a chess study 234
6.4 CGP-based methods and XAI 239

List of Tables

3.1 Panel of CGP variants 100
3.2 Table 1 | Parameters 144
3.3 Extended Data Table 1 | Functions included in the default library $\mathcal{L}$ accessible by Kartezio 170
3.4 Extended Data Table 2 | Fitness scores achieved by Kartezio with descending numbers of training images 171
3.5 Extended Data Table 3 | Summary of Use Case dataset preparation and analysis 171
3.6 Extended Data Table 4 | Summary of Use Case model parameterization 172
3.7 Extended Data Table 5 | Summary of Use Case fitness scores . 172

4.1 Studied Hyperparameters 178

5.1 Function Libraries used for MAGE and MT-CGP 209
5.2 Experimental Parameters 209
5.3 Solved: the number of runs which fully solved the given PSB2 problem, out of 20. Generation: the average generation at which the problem was solved. 214
5.4 Results on Cell Image Library dataset, a comparison between mono-typed CGP and MAGE model (10 runs). 218

6.1 Parameters and Results summary 230

Acronyms

ACGP Amplitude oriented Cartesian Genetic Programming.

ADF Automatically Defined Function.

AE Auto Encoder.

AI Artificial Intelligence.

ALE Atari Learning Environment.

ALIF-CGP Automatic Learning of Image Filters using Cartesian Genetic Programming.

ANN Artificial Neural Network.

APC Antigen Presenting Cell.

BBO Biogeography-Based Optimization.

BCGP Balanced Cartesian Genetic Programming.

BCR B Cell Receptor.

BERT Bidirectional Encoder Representations from Transformers.

CGP Cartesian Genetic Programming.

CGP-IP Cartesian Genetic Programming for Image Processing.

ChatGPT Chat Generative Pre-trained Transformer.

CNN Convolutional Neural Network.

CPU Central Processing Unit.

CRCT Cancer Research Center of Toulouse.

CS Computer Science.

CTL Cytotoxic T Lymphocyte.

CV Computer Vision.

DAG Directed Acyclic Graph.

dCGP Differentiable Cartesian Genetic Programming.

DL Deep Learning.

DNN Deep Neural Network.

DRL Deep Reinforcement Learning.

DT Decision Tree.

EA Evolutionary Algorithm.

EC Evolutionary Computation.

ES Evolution Strategy.

FC Fully Connected.

FCN Fully Convolutional Network.

FMCGP Frameshift Mutation Cartesian Genetic Programming.

FPGA Field-Programmable Gate Array.

GA Genetic Algorithm.

GAN Generative Adversarial Network.

GMM Gaussian Mixture Model.

GP Genetic Programming.

GPT Generative Pre-trained Transformer.

GPU Graphical Processing Unit.

GRU Gated Recurrent Unit.

HOG Histogram of Oriented Gradients.

IF immunofluorescence.

IFN-γ Interferon-γ.

IHC Immunohistochemistry.

IoU Intersection over Union.

IP Image Processing.

IRIT Institute of Research in Computer Science of Toulouse.

IS Instance Segmentation.

IUCT University Cancer Institute of Toulouse.

kNN k Nearest Neighbors.

LGP Linear Genetic Programming.

LLM Large Language Model.

LSTM Long Short Term Memory.

MAGE Multimodal Adaptive Graph Evolution.

MAS Multi-Agent System.

Mask R-CNN Mask Region-CNN.

MCTS Monte-Carlo Tree Search.

MCW Marker-Controlled Watershed.

MFI Mean Fluorescence Intensity.

ML Machine Learning.

MRI Magnetic Resonance Imaging.

MT-CGP Mixed-Types Cartesian Genetic Programming.

NAS Neural Architecture Search.

NK Natural Killer.

NLP Natural Language Processing.

NN Neural Network.

PCA Principal Component Analysis.

PSO Particle Swarm Optimization.

RCGP Recurrent Cartesian Genetic Programming.

ReLU Rectified Linear Unit.

REVA Real Expression Artificial Life.

RGB Red, Green, Blue.

RNN Recurrent Neural Network.

ROI Region Of Interest.

SE Structuring Element.

SGD Stochastic Gradient Descent.

SMAPs Supra Molecular Attack Particles.

SMCGP Self-Modifying Cartesian Genetic Programming.

SS Semantic Segmentation.

STGP Strongly-Typed Genetic Programming.

SVM Support Vector Machine.

TCR T Cell Receptor.

TGP Tree-based Genetic Programming.

TIRFM Total Internal Reflection Fluorescence Microscopy.

TNF-α Tumor Necrosis Factor-α.

UMAP Uniform Manifold Approximation and Projection.

VAE Variational Auto Encoder.

ViT Vision Tranformer.

WSI Whole Slide Image.

XAI eXplainable Artificial Intelligence.

Chapter 1

Introduction

1.1 Thesis context

As one of the major cancer research hubs in France, the Oncopole de Toulouse has been the incubator for the pioneering work presented in this thesis. Cancer Research Center of Toulouse (CRCT) plays an important role in this endeavor, fostering innovative and practical research solutions in its relentless commitment to combating cancer. Close collaboration with University Cancer Institute of Toulouse (IUCT) ensures the transposition of our research findings into clinical settings, thereby improving patient care. Our laboratory, nestled within this dynamic ecosystem, is uniquely positioned to marry advanced computational methods with impactful medical research, embodying the power of such synergistic collaborations. The genesis of this work can be traced to the collaborative synergy of the *Molecular Dynamics of Lymphocyte Interactions* (*DynAct*) team at CRCT, in conjunction with a strong and growing collaboration with the Institute of Research in Computer Science of

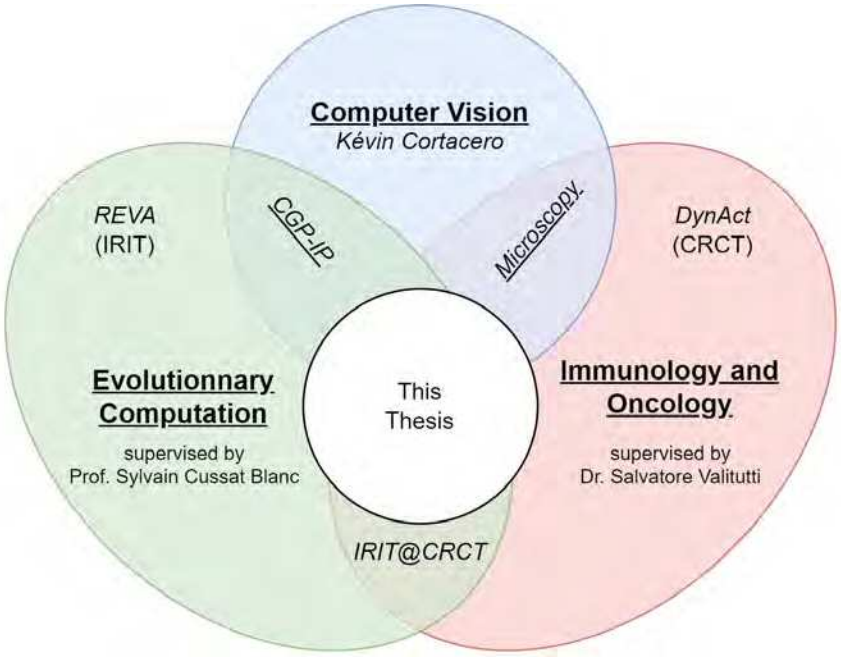


Figure 1.1: **The context of this interdisciplinary Thesis**
Thesis at the interface between Computer Science (Computer Vision and Evolutionary Computation) and Biology (Immunology and Oncology)

Toulouse (IRIT). The DynAct team, made up of an interdisciplinary mix of biologists, oncologists, clinicians, and computer scientists, is at the heart of this innovative endeavor. This collaborative effort has also benefited from a strengthening partnership with the Real Expression Artificial Life (REVA) Team, a member of IRIT, one of the most significant Computer Science (CS) laboratories in France. This partnership brings to the fore a rich exchange of knowledge, cutting-edge research, and innovative methodologies. The merging of REVA's computational expertise with CRCT's medical proficiency facilitates a comprehensive approach to our research, empowering us to tackle challenges from multiple perspectives. Collaboration between the two laboratories has grown enormously in recent years, with the integration of a REVA branch directly into the CRCT (under the name of *IRIT@CRCT*) The convergence of these multidisciplinary entities creates an ecosystem that fosters academic innovation, technological advancement, and practical healthcare solutions. Within this dynamic environment begins the research into Cartesian Genetic Programming (CGP) for explainable biomedical Image Processing (IP) with this work (see **Figure 1.1**).

1.1.1 Oncology and Immunology

Research within the DynAct team focuses primarily upon two different types of biomedical images: histological slides and *in vitro* immunofluorescence (IF) images. These image types offer complementary information on the complex interactions between cancer and the immune system. Histological slides (also known as Whole Slide Images (WSIs)), through their ability to present de-

tailed morphological structures of tissues, allow us to observe the physical architecture of cancer and the surrounding microenvironment. They provide visual insight into cancer's invasive nature, revealing how it shapes and is shaped by the tissue landscape in which it resides. *In vitro* IF images offer a different, yet equally profound perspective. Through the use of fluorescent labels, IF imaging allows us to observe the expression and spatial distribution of specific molecules within cells, thereby casting light on the molecular level of immune responses in the cancer landscape. This panel of microscopy techniques enables the team to study the entities involved in immune defense in cancer patients, at different scales. The large quantity of images acquired must then be analyzed to extract quantitative information or generate new hypotheses. This is where 2 factors come into play: **i)** human bias, which can introduce irregularities into the annotation/quantification process (fatigue, bias linked to knowledge of the condition, different days...); **ii)** the massive amount of time it takes to annotate thousands of entities accurately and consistently by hand. This is where automatic analysis is particularly relevant, especially in the case of medical data. In the image domain, the main challenge is to detect, count or even segment objects of interest, which is usually the first step.

1.1.2 Computer Vision and Image Processing

The realm of CS and IP, a critical foundation of our research, is an arena that is continually evolving and expanding. It encompasses a wide range of techniques, algorithms, and methods that are intended to analyze, interpret, and

enhance digital images. Its application in biomedical research, particularly in the context of our work with histological slides and *in vitro* IF images, is profoundly influential. In the world of CS, algorithms are the engines that power data interpretation and decision-making. In the context of IP, they transform vast arrays of pixel data into comprehensible information. These algorithms range from classical IP techniques, such as filtering and edge detection, to more advanced methodologies that incorporate Artificial Intelligence (AI) and Machine Learning (ML). IP techniques are indispensable tools for interpreting biomedical image data. They enable researchers to extract critical information from images, such as the spatial arrangement of cells, the morphological characteristics of tissues, or the distribution of specific molecular markers as described in previous section.

1.1.3 Evolutionary Computation and Genetic Programming

The REVA Team, with its robust expertise in Evolutionary Algorithm (EA) and their applications to biomedical data and challenges, led me to Evolutionary Computation (EC) domain, a sub-field within AI, encapsulates a family of algorithms inspired by biological evolution principles. These include Genetic Algorithm (GA), Evolution Strategy (ES), Particle Swarm Optimization (PSO) and Genetic Programming (GP). Among these, the focus of this thesis lies in harnessing the capacity of GP and CGP to evolve IP pipelines. CGP has in fact been the subject of work by REVA member Dennis G. Wilson, notably with his model agent on Atari Learning Environment

(ALE) [5].

GP and CGP provide a robust and flexible framework for evolving computer programs. In our context, they allow us to optimize and tailor IP pipelines for analyzing histological slides and *in vitro* IF images. The transparent and explainable nature of the algorithms produced is of immense value, enabling us to not just extract information from biomedical images but also understand the underlying decision-making process.

The collaboration with the REVA team at IRIT deepens our understanding of these EC methods and their potential applications in biomedical image analysis. The major work of this thesis centers around the use of GP and CGP to evolve IP pipelines, a significant endeavor aiming to transform how we interpret and understand biomedical images. By successfully navigating this exploration, we aspire to uncover novel insights in cancer biology and immunology and foster advances in diagnostic tools and therapeutic strategies.

1.2 Main challenges

This thesis is set against the backdrop of several pressing challenges in the field of biomedical image analysis. While these challenges bring complexity, they also offer unique opportunities for innovation and discovery. The diverse nature of image data, the scarcity of annotated images, the high computational requirements of contemporary methods, and the need for explainable algorithms - these form the quartet of challenges that our research seeks to address. The diversity inherent in biomedical image data is a sig-

nificant barrier to consistent analysis. Annotated images, vital for training accurate algorithms, are often in short supply due to the laborious process involved in their creation. Additionally, conventional IP and ML methods often necessitate substantial computational resources, presenting a barrier for resource-constrained settings. Furthermore, explicability, known as eXplainable Artificial Intelligence (XAI) [6, 7], or the ability to understand the reasoning behind an algorithm, is an imperative often overlooked in the face of the "black-box" nature of many contemporary existing techniques such as Deep Learning (DL).

1.2.1 Image Data Diversity

Diversity in image data presents both a challenge and an opportunity. Biomedical images can vary in numerous ways, particularly in terms of the tools used to acquire them: radiology, Magnetic Resonance Imaging (MRI), ultrasound scan, ligth and electron microscopy, etc. In the context of this thesis, we are only interested in the case of microscopy, and more specifically in two types: bright-field histological slides and fluorescent images. In addition to the type of microscopy, there are a number of other parameters that can be used to extend the range of possibilities: scale (organ, cell, particle), type of sample (*in vitro*, *in-vivo*), dimensionality (2D, 3D, time), resolution, staining techniques... This diversity means that an IP pipeline optimized for one type of image might not perform as well when applied to another. The challenge, therefore, lies in developing flexible, adaptable algorithms that can effectively handle this diversity and extract meaningful information from various

types of images. With the help of CGP, this thesis endeavors to evolve such versatile IP pipelines, turning the challenge of image data diversity into an opportunity for innovative computational solutions.

1.2.2 Few Annotated Images

In the field of biomedical research, one of the most significant challenges is the scarcity of annotated images. Annotated images serve as a guide for training algorithms, providing labels that help the algorithm understand what to look for in an image. However, generating these annotated images is a laborious and time-consuming task, often requiring the expertise of a trained professional. Consequently, internal datasets in biomedical image analysis are often smaller than those in other fields of ML. This thesis addresses this challenge by harnessing the power of existing IP filters known for their ability to produce robust solutions when assembled and tuned by human experts.

1.2.3 Frugality and low requirements

Another significant challenge lies in the high computational requirements of many existing IP and ML methods. Training complex models often requires a substantial amount of computational power and memory, which can be a limiting factor, particularly in resource-constrained settings. The goal of this thesis is to tackle this issue by leveraging the efficiency of CGP. By evolving IP pipelines that deliver comparable performances with less data and lower energy consumption, the project aims to create frugal yet effective computational solutions.

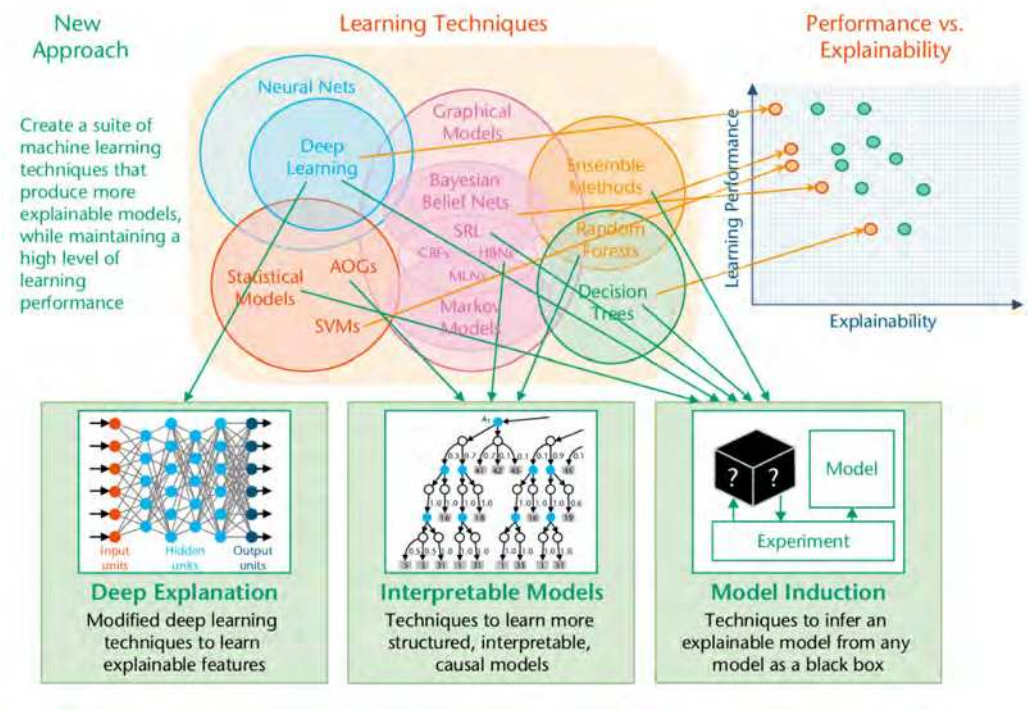


Figure 1.2: **Learning Performance Versus Explainability Trade-Off**
"Several Categories of Learning Techniques and Strategies for Developing Explainable Models"
Source: Figure 2 [6]

1.2.4 Explainable Artificial Intelligence

Explicability, or the ability of a ML model to make its decision-making process understandable to humans, is a vital requirement in biomedical research. This branch of AI and ML has been defined and named XAI by the *DARPA* project in 2017 [6, 7]. As depicted in **Figure 1.2**, XAI is associated with the trade-off between learning performance and explicability, often with difficulty in achieving both. Traditional DL methods, while powerful, often act as "black boxes", providing little insight into how they reach their conclusions. This lack of transparency can be a significant drawback in a field where un-

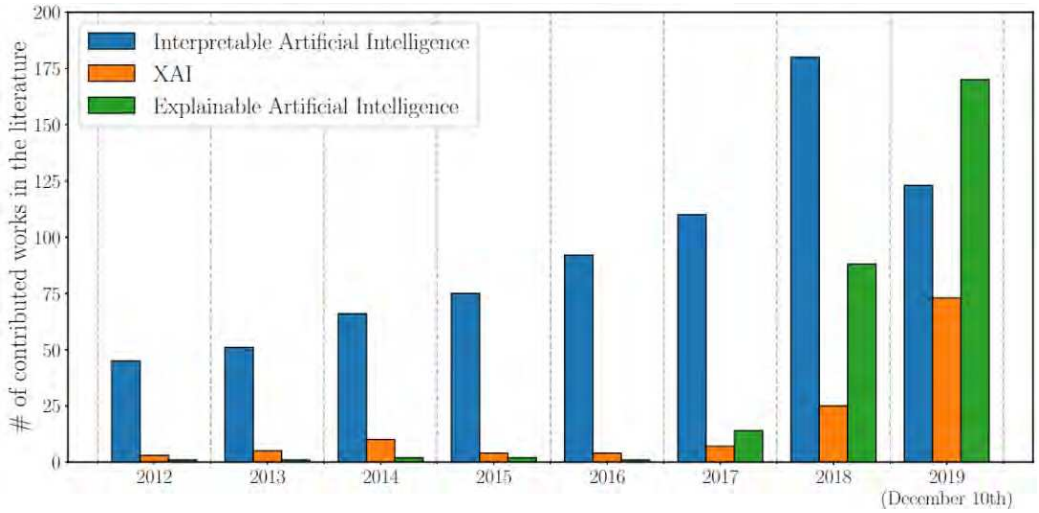


Figure 1.3: **Evolution and Adoption of XAI search over years**
Sources: Figure 1 [7]

derstanding reasoning behind a decision can be as important as the decision itself. A major focus of this thesis is on overcoming this challenge by using CGP to generate readable and accessible algorithms. The field of XAI has grown considerably in recent years and has become a real focus of research (as shown in **Figure 1.3**) and this work aims to make a contribution to image segmentation problems. This thesis is mainly based on the generation of *Interpretable Models*, and some ideas about *Model Induction* (as defined in **Figure 1.2**) are discussed in **Chapter 6**.

Chapter 2

Background

This thesis is at the intersection of various disciplines such as Immunology, Oncology, Microscopy (*DynAct Team* - CRCT, IUCT) on the one hand and EC (*REVA Team* - IRIT) and Computer Vision (CV) from my Master Degree on the other hand (all summarized in **Figure 1.1**). This Chapter is divided in 4 parts, to cover all the previously cited fields that are the founding pillars that make up my work. I first briefly introduce the world of Immunology to provide scientific context for studies included in that work. Afterwards I present different methods to treat images with basic filters or more complex operations. Then I summarize the DL models which is the area that has been most developed in recent years and from which we must obviously draw inspiration. Finally, this chapter concludes with the most important aspect of this thesis, despite its low visibility in the literature compared to DL: Cartesian Genetic Programming (CGP).

2.1 Immunological Context: The Battlefield

The immune response has been frequently compared to a battlefield in which a "good army", the immune system, deploys multiple highly specialized cells and extracellular molecules that act in concert to combat against "foreign invaders" of our body such as pathogens (bacteria, viruses, parasites) and "internal traitors" such as cancers. In this view, all the cells of the body that are not part of the immune system can be assimilated to a civil population that is victim of an attack and needs to be protected by the army. This view is obviously a simplification; however, it is a very convenient way for non-immunologist to look at immune system and to understand the dynamics of the immune response. As a Computer Scientist, I personally visualize and understand Immunology through the spectrum of a gigantic and very complex Multi-Agent System (MAS), which is in line with other work lead by my supervisor Sylvain Cussat-Blanc.

The immune system is composed of two main parts: the innate immune system (a complex cellular and molecular network that is alerted early after body invasion) and the adaptive immune system a molecular and cellular network that is more specialized against precise targets and enters into action after the innate immune system. An immune response is therefore composed by an initial innate phase that is followed by an adaptive phase. It is important to note that innate and adaptive immune systems are not independent entities, on the contrary, they strongly collaborate to efficiently eliminate pathogens and cancerous cells. It is interesting to note that the actions of innate and adaptive immune systems are somehow complementary. While

the innate immune system is rapidly and efficiently activated, thanks to preformed cells and receptors it does not keep memory of previous responses. It somehow starts every time from zero. On the contrary, the adaptive immune system keeps a memory of its previous responses, it is therefore able to potentiate its responses when repeatedly stimulated such as in the case of vaccinations.

2.1.1 The innate immune system

The protective body surfaces

The external (skin) and the internal (gut and respiratory mucosa) surfaces of the human body are part of the innate immune system since they participate in the defense against invading pathogens. They protect the body not only by acting as physical barriers, but also thanks to chemical substances (enzymes, mucus, acidic pH etc) and to antimicrobial molecules such as defensins.

The cells of the innate immune system

The first cells of immune system that enter into action after body invasion by pathogens are phagocytic cells named neutrophils that reach the site of infection very rapidly. Their role is to circulate, migrate to the site of infection and ingest bacteria. A second class of phagocytic cells are the monocytes. They arrive later into the site of infection and strongly contribute to pathogen phagocytosis and elimination and to tissue repair. Importantly, monocytes are also involved in the initiation of adaptive immune response since once they have ingested pathogens, they can digest their proteins and present

small portions of these proteins (the antigens) to T lymphocytes. By this way, they trigger the adaptive immune response. A third cell type belonging to the innate immune system is represented by the Natural Killer (NK) cells. These cells have several characteristics in common with T lymphocytes (e.g. they can kill virus-infected and tumor cells). However, they are part of the innate immune system since they participate in the early phase of immune response and do not express the T cell antigen receptor that characterize T lymphocytes. Like other cells in the innate immune system,NK cells express receptors that allow them to recognize stressed cells in a global fashion (either infected cells or cells undergoing neoplastic transformation).

The soluble molecules of the innate immune system

Several molecules play a key role in the innate immune system. Among these, it is important to mention the cytokines (molecules that are secreted by cells of the immune system and have effect on other cells). Some cytokines are released early after infection and therefore participate to innate immune response, others are more typical of adaptive immune response. Tumor Necrosis Factor-α (TNF-α) and IL-1 (Interleukine) are classical innate immunity cytokines that play a key role in the initiation of immune response. Among cytokines, the chemokines belong to a specialized subset that has the role of attracting cells of the immune system to the site of infection. The chemokines that participate to the innate immune response are released early and allow the recruitment of key cells of the innate and adaptive immune response.

The receptors of the innate immune system

A third very important class of molecules involved in the innate immune response are the receptors. They are expressed on the surface of the cells of the innate system and allow them to globally recognize structures that are expressed/produced by pathogens or by damaged or dying cells. These receptors, named pattern recognizing receptors play a key role in the initiation of the immune response by allowing the immune system to detect molecular structures that are exclusively produced by pathogens and danger signals that are emitted by stressed cells. In both cases, these very efficient receptors allow to detect in a "global fashion" and without specificity any dangerous aggression by pathogens or neoplastic cells.

2.1.2 The adaptive immune system

The cells of the adaptive immune system

The main cells of the adaptive immune system are the B and T lymphocytes. The B lymphocytes express on their surface the B Cell Receptor (BCR) that allows them to recognize the antigens associated to pathogens in their native 3D structure. Upon antigen-recognition, B lymphocytes undergo proliferation and, under the influence of other cells of the immune system (including the helper T cells), undergo to a process of maturation and selection to become antibody producing cells. Only the B cells that express high affinity antibodies survive selection while the other are eliminated. Via this process, at any immunization with a given antigen, a B cells of a person produce antibodies at higher affinity for a given antigen. By this way, antibodies of

immunized individuals can efficiently neutralize viruses and bacterial toxins.

The T lymphocytes express on their surface the T cell antigen receptor that allows them to recognize antigens derived from pathogens and cancer cells. Differently from B cell, the T Cell Receptor (TCR) cannot recognize the antigens in their native form, but only following their processing by specialized cells named the Antigen Presenting Cells (APCs). APCs uptake protein antigens and degrade them using intracellular proteolytic systems. Small pieces of antigens are exposed on the surface of the APCs in association with specific molecules named the histocompatibility molecules. Although various cells including monocytes can serve as APC, the most efficient APCs (defined as the sentinels of the immune system) are the dendritic cells. These cells are specialized in antigen capture and presentation for efficient activation of T cells as depicted in **Figure 2.1**.

Following initial antigen recognition, T cells proliferate and acquire their biological functions that are central in the development of adaptive immune responses. On the basis of their function, T cells are divided into two main sub-populations: the T helper (Th) and Cytotoxic T Lymphocyte (CTL). The Th (divided in several subtypes, each one with a specific functional role) all share the main function of helping other cells of the immune system to fulfill their mission. For instance, Th cells help B cells to produce high affinity antibodies, monocytes to efficiently eliminate phagocyted pathogens, etc. A second important population of T cells is the CTLs. These cells are at the center of the research of our team. Such as NK cells, CTLs have the role to eliminate infected cells and/or tumor cells (named target cells). However, the main property of CTLs enter into action late in the case of failure of

the innate immune system to control the aggression. Differently from NK cells, the CTLs express a TCR that can allow them to specifically recognize antigens of the surface on target cells. They have therefore a delayed but more precise action when compared to NK cells. To kill their targets NK cells and CTL secrete a protein named perforin that forms pores in the target cell membrane. Via these pores additional toxic molecules produced by the NK cells and CTLs penetrate in the target cell cytoplasm to induce cell death. The biologists team in which this PhD has been conducted is specialized in the study of the fight between human CTLs and tumor target cells. In particular the team studies the molecular events occurring within a specialized area formed at the contact site between CTL and target cells named the lytic synapse. At the lytic synapse, CTLs secrete their arsenal of toxic molecules, while tumor cells deploy mechanisms of defense and counter-attack against CTLs. We study the molecular steps of this fight with the aim of shifting the balance between CTL efficacy and tumor target cell resistance in the favor of CTL [8, 9].

The soluble molecules of adaptive immune system

As mentioned above, cytokines and chemokines are also released during the adaptive phase of the immune response, but they are released relatively late (hours/days after infection) and have a different function. Among the key cytokines of the adaptive immune response, it is important to mention IL-2 (also known as T cell growth factor) that induces the T cell proliferation that is at the basis of all adaptative immune responses and Interferon-γ (IFN-γ) a key amplifier of all steps of the adaptive immune response. Antibodies

are also key soluble molecules of the immune system. Released by the B cells, they play a key role in neutralizing viruses and bacterial toxins (to avoid their entry into cells) and in favoring the phagocytosis of bacteria and of other pathogens. A condensate report of inate and adpative system for computational biologists can be found in the introduction written by Khazen et. al [10].

The immune system/cancer cell interface

As mentioned above, the immune system fights against external enemies (such as bacteria and viruses) and against internal traitors (somatic cells undergoing neoplastic transformation). When compared to invading pathogens, cancers are much more subtle enemies since they exhibit an indolent progression. Cancer cells do not express the pathogen associated molecular patterns useful to alert innate immunity and release danger signals slowly during a period of months/years. For these reason, they can be easily missed by the immune system "radars" until it is too late. Moreover, during the prolonged competitive interaction between a developing cancer and the immune system the selective pressure exerted by the immune system "sculpt" the cancer. Cancer cells undergo random genetic mutations and those that are more fit to resist immune system attack survive and generate a progeny of resistant cells. The mechanisms of resistance acquired by cancer cells can be either intrinsic (resistance to CTL attack, mechanisms of membrane reparation, inhibition of cell death pathways, etc) or extrinsic (release of immune suppressive molecules, recruitment of immune suppressive cells, expression of immune suppressive molecules in the cancer microenvironment etc). This

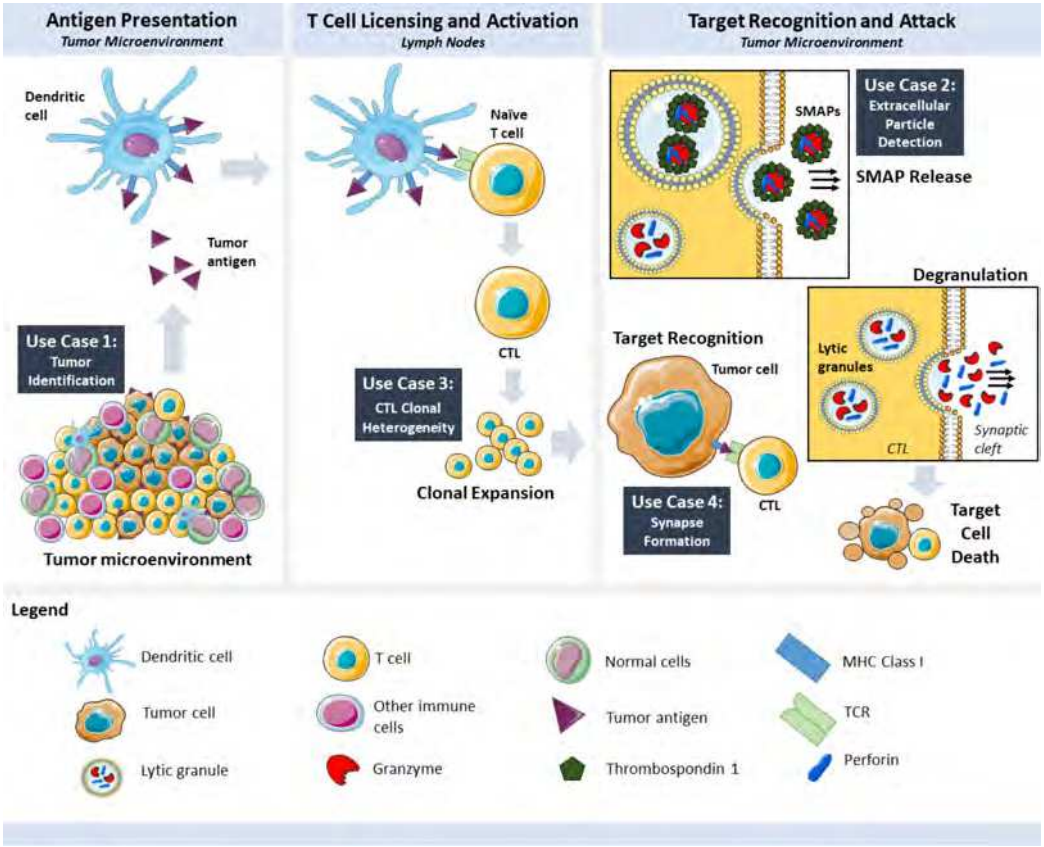


Figure 2.1: **Simplified T cell response against Tumor cells**
The 3 main steps of the immune response and the 4 Use Cases presented in the main article (see **Chapter 3**).
Adapted from: McKenzie et al [8] Done using: Servier Medical Art, licensed under a Creative Commons Attribution 3.0 Unported License (https://creativecommons.org/licenses/by/3.0/)

process named "Cancer Immunoediting" is consider to be a key mechanism of cancer cell escape to immune surveillance [11]. Scope of the immunotherapy is to re-invigorate immune system in cancer patients by re-directing many killer cells against cancer and, at the same time, inhibiting cancer-induced immunosuppression and cancer cell resistance to death. In this context of war, my contribution as CV expert can be viewed as that of cartographer who helps the army to identify the maximum of entities present on the battlefield.

2.2 Fundamentals of Image Processing

CV can be seen as the equivalent of the visual and sensory system in living species, artificially reproduced to enable machines to understand their environment. It includes a broad spectrum of areas, such as sensor calibration, data acquisition/encoding, data processing and feature extraction that may lead to a decision or classification. Among these, a fundamental called IP involves the manipulation of images through precise operations to extract high-level features, (or generate, restore, compress...) from a very specific type of data format: the image. The type of image data and its subsequent uses can vastly differ depending on the sensor employed. For example, stereo cameras generate depth maps, infrared sensors yield infrared images, while color cameras generally produce Red, Green, Blue (RGB) images.

Images primarily encode information in a spatial domain, typically spanning two dimensions x and y. This thesis primarily targets this spatial domain, but it's worth mentioning that other domains also exist and hold importance, such as the Fourier space representing frequencies and the Hough space encompassing equations.

An image can be conceived as a collection of channels; in the case of standard images, these are typically composed of RGB channels. Monochrome or grayscale images are a special instance that consists of just a single channel. Each channel can be considered as an array or matrix, hence inheriting the mathematical properties of these entities.

The specific value at a given coordinate within this array is known as the pixel's *intensity* I. Therefore, the values of the pixels forming the channel

represent the signal intensity captured by the camera sensor at the corresponding coordinates, considering the projection of the optic model. A higher pixel value typically corresponds to a more intense signal received, thus allowing for a broad array of image types.

The range of possible pixel intensity values depends on the *depth* or encoding of the image, which is determined by the number of bits allocated for storing each pixel value. For instance, a 32-bit float representation (*f32*) allows for managing values between 0.0 and 1.0, whereas an 8-bit unsigned integer encoding (*u8*) can handle values ranging from 0 to 255.

Figure 2.2 presents a standard IF image (of size 512×512 pixels), acquired and provided by Lucie Demeersseman, with a u8 encoding scheme. This image is composed of three grayscale channels (as illustrated in the second row): the transmission (similar as what human could watch on a microscope) and two additional channels (acquired by IF) each expressing the presence of a biological entity: DiO and WGA. While these channels are fundamentally grayscale, they are often assigned different colors for improved visualization and interpretation as illustrated in the first row.

2.2.1 Arithmetic Operators for Images

Treating an image as an array in the mathematical sense enables us to apply a variety of arithmetic operations.

For images of identical sizes, basic arithmetic operations are performed element-wise. For example, when adding two images A and B, each pixel value of image A is added to the corresponding pixel value in image B to create a

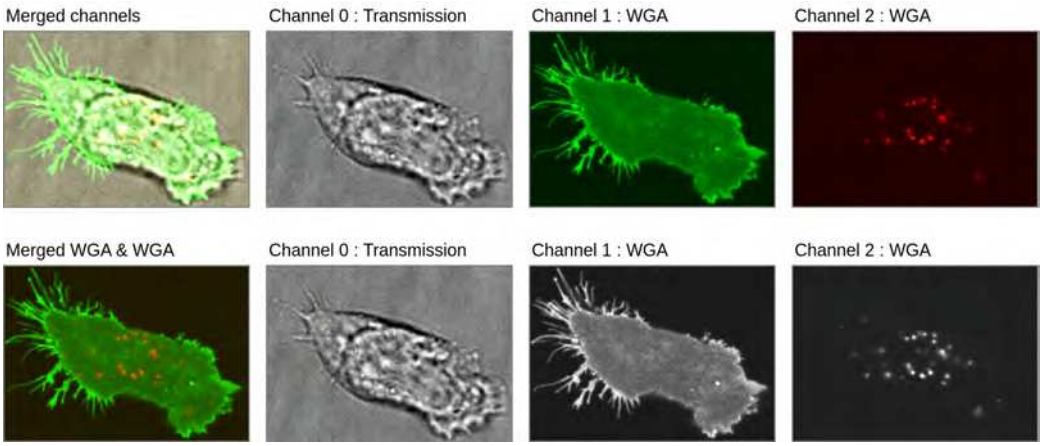


Figure 2.2: **An Immunofluorescence image, composed of 3 channels** Transmission (grayscale), DiO (green), WGA (red).

new image. This concept extends to other operations such as *Subtraction*, *Division*, and *Multiplication*, depicted in **Figure 2.3**.

Furthermore, the image can be manipulated in various ways, such as inverting it (i.e., subtracting each pixel value from the maximum value possible in the given encoding), multiplying it (to double its intensity, for example), or dividing it (to halve its intensity, for example). These operations provide a powerful toolkit for processing and enhancing images. Comparative operations can also be employed on a per-element basis, like calculating the *Mean* of two images or determining the *Min* and *Max* pixel values across the images. These operations are depicted in **Figure 2.4**

Arithmetic operations can also be applied to individual images. Mathematical functions such as the *Logarithm*, *Exponential*, *Square Root*, and *Square* can be used on a per-element basis to generate a new image.

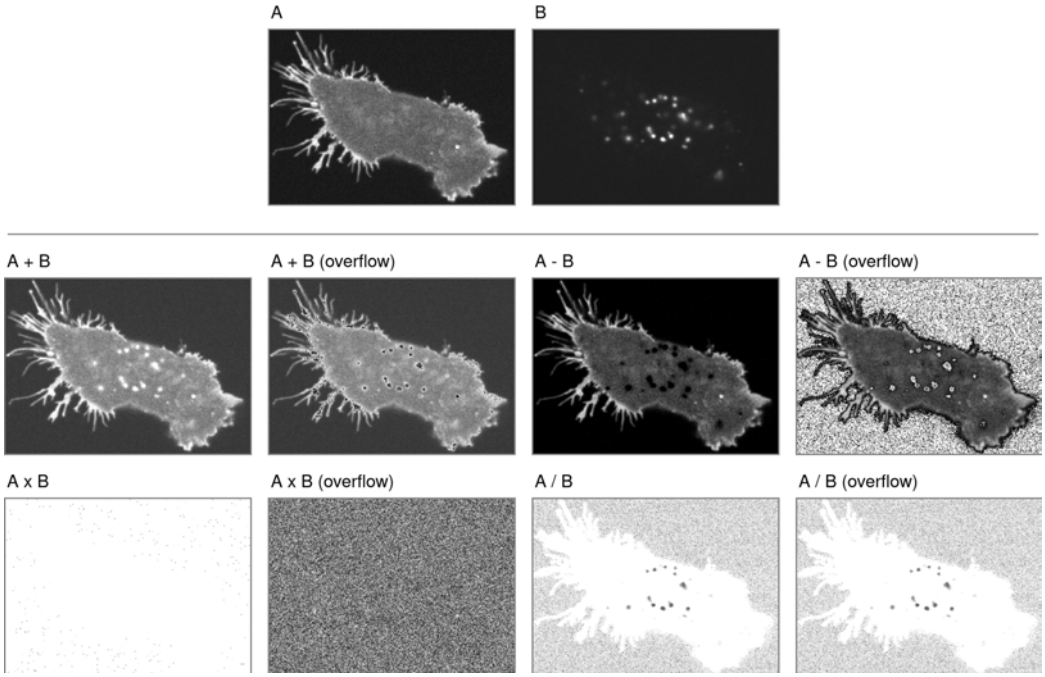


Figure 2.3: **Basic Arithmetic Operations and the Overflow effect**

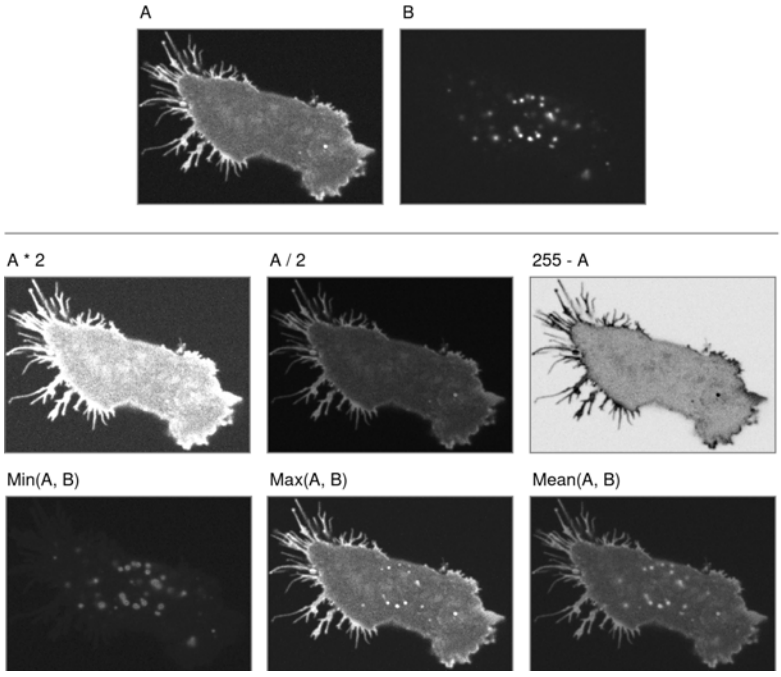


Figure 2.4: **More advanced Arithmetic Operations**

2.2.2 Logical Operators for Images

Boolean algebra, a mathematical structure introduced by George Boole in 1847 [12], provides a range of logical operators that can be used for image processing. The *And* operation (denoted $A \wedge B$) works on binary images, usually encoded with values of 0 or 1. This operation performs a logical intersection of two images, retaining only the pixels that are deemed positive in both images. Similarly, the *Or* operation (denoted $A \vee B$) performs a logical union of two binary images. It preserves pixels that are positive in at least one of the two images. The *Xor* operation (denoted $A \otimes B$) retains pixels that are positive in exactly one of the two binary images. In essence, it performs a logical union of two images, excluding the intersection. Lastly, the *Not* operation (denoted $\neg A$) operates on a single image and inverts its pixel values. A pixel value of 0 becomes 1 and vice versa.
These logical operations (depicted in **Figure 2.5**) enable to construct logical manipulations of images, significantly enhancing our ability to process and extract meaningful information from them.

2.2.3 Convolution filter

With the surge of DL [13], particularly in the realm of Convolutional Neural Network (CNN) models [14–17], the *Convolution* filter (denoted $*$) has risen to prominence among IP filters. This filter, rooted in signal processing, involves a sliding window (called a Kernel k) passing throught the whole image A. The operation calculates a new value for each position of this window, thus producing a new image B: $B = k * A$ a illustrated in **Figure 2.6**.

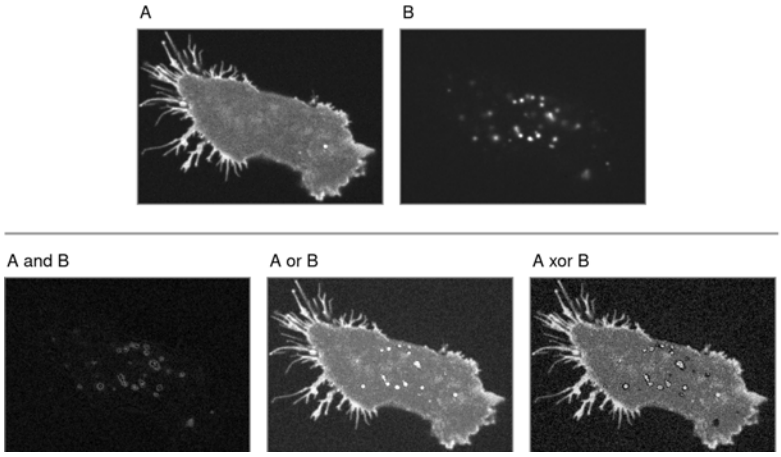


Figure 2.5: **Basic Logical filters**

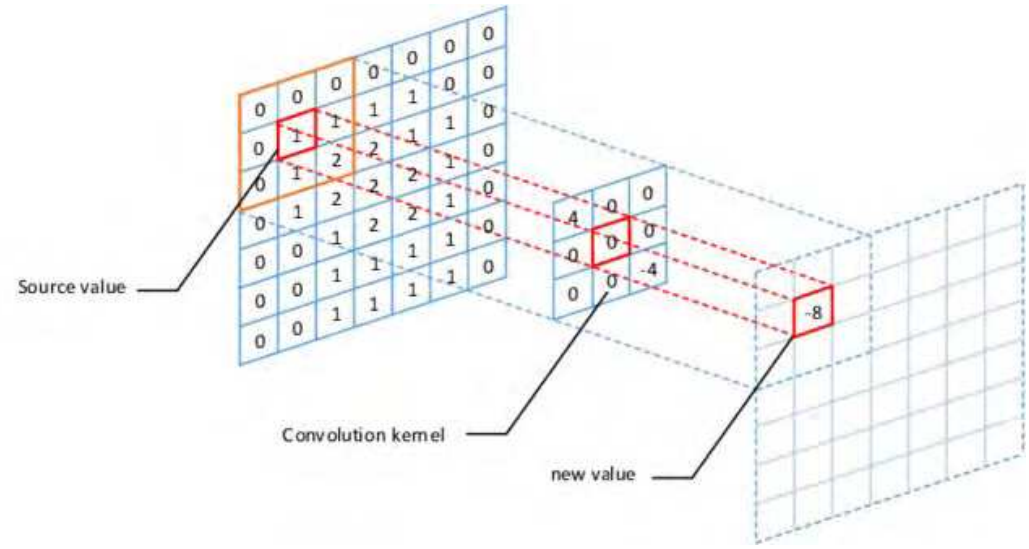


Figure 2.6: **Image Convolution Illustration**
Source: Figure 2 [18].

Convolution Kernel

The kernel k, which slides over the entire image, determines the effect of the convolution on the image. The size of the kernel is typically a 3×3 matrix, but it can be increased, bearing in mind that a larger kernel size will impact computation time. While the kernel need not be square, its dimensions should be odd to ensure the presence of a central pixel, the pixel of interest.

The power of convolution largely rests on the definition of effective kernels, the design of which has been a subject of intensive research over past decades. In recent times, convolution filters have been optimized using Stochastic Gradient Descent (SGD) [19] (a technique based on Gradient Descent [20]) and Backpropagation [14, 21] within the convolution layers of Deep Neural Network (DNN)s.

2.2.4 Blurring and Sharpening

Convolution finds a straightforward application in blurring an image. In this process, the sliding window computes the new pixel value based on the statistical attributes of the surrounding pixels, thereby reducing minor discrepancies. By tailoring the kernel, however, one can accentuate regions distinct from their surroundings by sharpening them.

The *Box Blur* is a common technique that employs a uniform kernel to compute an average, effectively causing the central pixel to take on the mean value of its neighboring pixels. The scope of this neighborhood can be expanded or contracted by adjusting the kernel's dimension.

The *Gaussian Blur* is another tool for smoothing an image, with its kernel adhering to a Gaussian distribution. This gives more weight to the nearer neighbors than those further away. The spread of the 2D Gaussian distribution is typically set to σ to populate the Kernel matrix.

In a similar vein, the *Median Blur* assigns the median value of the neighboring pixels to the central pixel. This technique is especially effective for noise reduction as it does not create a new intensity value for the image.

Finally, an image can be *Sharpened* or its acutance restored using a sharpening kernel. This technique helps to highlight the boundaries or edges in an image. The Box (k_B), Gaussian (k_G) and Sharpen (k_S) 3×3 kernels are represented as matrices, shown in **Equation 2.1**. These filters (including Median Blur) are depicted in **Figure 2.7**.

$$k_B = \frac{1}{9} \times \begin{bmatrix} +1 & +1 & +1 \\ +1 & +1 & +1 \\ +1 & +1 & +1 \end{bmatrix} \quad k_G = \frac{1}{16} \times \begin{bmatrix} +1 & +2 & +1 \\ +2 & +4 & +2 \\ +1 & +2 & +1 \end{bmatrix} \quad k_S = \begin{bmatrix} 0 & -1 & 0 \\ -1 & +5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{2.1}$$

Indeed, advanced methods such as L0 smoothing [22] or bilateral texture filtering [23] can also be utilized to smooth an image while preserving textures and edges.

The *L0 Smoothing* technique [22] minimizes the L0 norm of the image's gradient during the smoothing process. This results in an image with fewer non-zero gradients, thus rendering the image smoother while maintaining sharp edges.

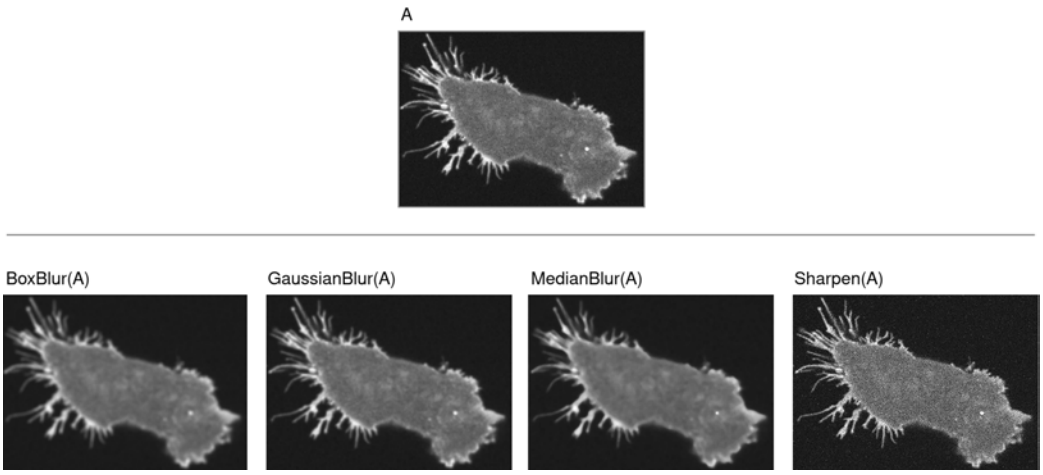


Figure 2.7: **Blurs and Sharpen Operations**
Box, Gaussian, Median Blurs and Sharpen filters.

On the other hand, *Bilateral Texture Filtering* [23] is a technique that decomposes the image into a base layer representing large-scale structures and a detail layer representing textures. The smoothing is then applied to the base layer, allowing for the preservation of textures and contours in the detail layer.

These methods contribute to the wealth of tools in IP that offer more refined control over the trade-off between smoothing and the preservation of image details.

2.2.5 Derivative and Edge Detection

Edge Detection is a fundamental task in IP, playing a key role in distinguishing objects of interest from the background and other parts of an image. This task can be accomplished through various methods, many of which involve the use of convolution (refer to **Section 2.2.3**) with specific kernels. Moreover, edge detection can also be achieved with DNN models such as the

Holistically-Nested Edge Detection model [24].

The foundation of edge detection lies in the mathematical concept of a derivative, which is instrumental in highlighting abrupt changes or variations. In the context of a 2D image, the derivative is usually calculated using a specific Kernel for each dimension, denoted as k_x (for the horizontal dimension) and k_y (for the vertical dimension).

Given these two dimensions, the *Gradient Magnitude* G can be computed (as shown in **Equation 2.2**). The computation relies on the Euclidean norm, also known as the magnitude vector.

$$\begin{aligned} G_x &= k_x * A \\ G_y &= k_y * A \\ G &= \sqrt{{G_x}^2 + {G_y}^2} \end{aligned} \tag{2.2}$$

The gradient magnitude provides a spatial mapping of the image contours in two dimensions. A high magnitude corresponds to a distinct contour, while a low magnitude suggests the absence of a contour.

Beyond the Gradient Magnitude, the *Gradient Direction* Θ is another piece of information that can be computed (as illustrated in **Equation 2.3**). This computation provides the angle direction for each pixel, resulting in a new direction map.

$$\Theta = atan2(G_x, G_y) \tag{2.3}$$

Edge detection filters serve to convert the simple pixel-level data into more complex features known as edges. These filters are primarily designed

around kernels that exploit subtle intensity variations at the pixel level. Some well-known filters are outlined in the subsequent paragraphs.

One of the first derivative operators was introduced by Lawrence Roberts, which used a pair of 2×2 kernels, shown in **Equation 2.4** for edge detection [25]. This approach was geared towards generating image gradients while minimizing computational requirements through the use of convolution.

$$k_x = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \quad k_y = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix} \tag{2.4}$$

In a similar fashion, the *Sobel* operator, developed by Irwin Sobel and Gary Feldman in 1968 [26], also utilizes two kernels (**Equation 2.5**) for edge detection.

$$k_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad k_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.5}$$

Parallel to the Sobel operator, the *Prewitt* method was introduced in 1970 [27], which also involves a pair of kernels (**Equation 2.6**). Sobel, Prewitt and Roberts operators are illustrated in **Figure 2.8**.

$$k_x = \begin{bmatrix} +1 & 0 & -1 \\ +1 & 0 & -1 \\ +1 & 0 & -1 \end{bmatrix} \quad k_y = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.6}$$

The *Emboss* filter is an effective tool for highlighting the edges or contours of an object in an image by giving it a three-dimensional or embossed

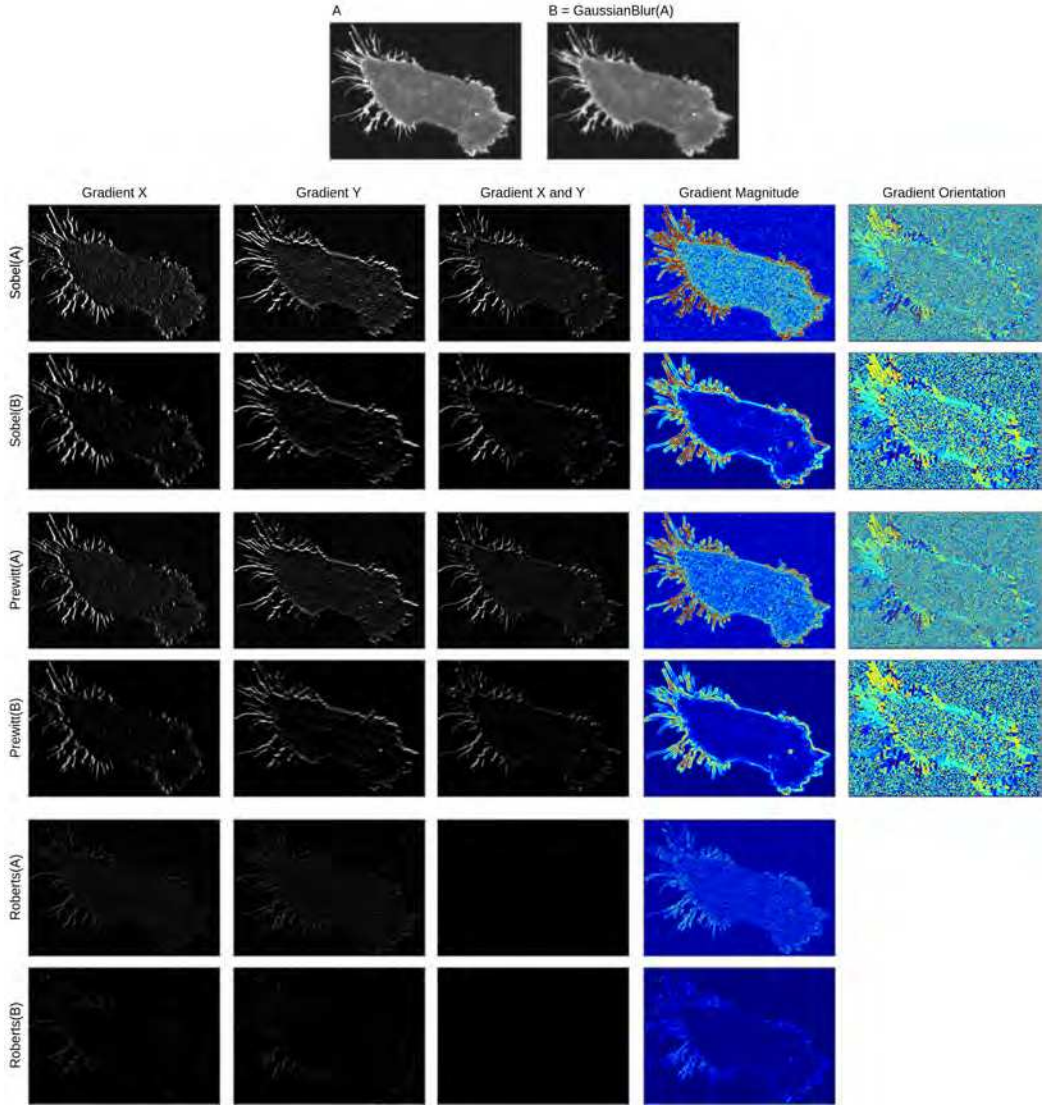


Figure 2.8: **Edge Detection (1) and Gradient**
Sobel, Prewitt and Roberts filters, Gradient Magnitude and Orientation.

appearance. It works by replacing each pixel in an image with a shadow, which is calculated based on the difference in color with surrounding pixels. This results in a final image where the edges are emphasized with a distinct 3D effect. Such as other filters, the Emboss filter employs a kernel (different orientations are possible) to achieve this, an example is given in **Equation 2.7**.

$$k_{Emboss} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \tag{2.7}$$

Expanding on the idea of filter pairs and orientation, *Kirsch* [28] introduced an edge detection model that operates in 8 directions, requiring a set of 8 different kernels (North: k_N, East: k_E, North-East: k_NE, and so on, shown in **Equation 2.8)**. The magnitude calculation for this method is derived from the maximum magnitude of each pixel across all directions.

$$k_N = \begin{bmatrix} +5 & +5 & +5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \quad k_E = \begin{bmatrix} -3 & -3 & +5 \\ -3 & 0 & +5 \\ -3 & -3 & +5 \end{bmatrix} \quad k_{NE} = \begin{bmatrix} -3 & +5 & +5 \\ -3 & 0 & +5 \\ -3 & -3 & -3 \end{bmatrix} \tag{2.8}$$

The *Canny* operator, introduced by John Canny in 1986 [29], is another enhancement to gradient detection. The process consists of several steps: **i)** Noise reduction in the image, usually achieved through a Gaussian filter; **ii)** Computation of the gradient amplitude; **iii)** A post-validation of pixels to

ensure they genuinely represent edges through a procedure known as Non-Maximum Suppression; and iv) Lastly, connected integer edges are filtered using a dual threshold (Hysteresis). An edge that exceeds the upper threshold is deemed a "strong edge", indicating a high degree of certainty that it is an actual edge. All edges connected to a strong edge are automatically validated. In contrast, edges falling below the lower threshold or those between the two thresholds are instantly discarded. This extends the earlier described edge filters, with the added advantage of minimizing the induction of false positives.

In 1987, Rachid Deriche proposed a variant [30], which modifies steps a) and b) of the Canny Filter by using an Infinite Impulse Response, thereby reducing the number of necessary hyperparameters to only one. This variant is also referred to as the *Canny-Deriche filter*. A further adaptation of the *Deriche filter* was proposed by Philippe Paillou in 1997 [31].

The *Discrete Laplacian Filter*, which represents the second derivative of an image, yields vertical and horizontal edge maps in one operation. Different kernels are used based on the neighborhood under study. In a $4n$ neighborhood, only the adjacent pixels are considered, while in an $8n$ neighborhood, the four diagonals are also included (either with less weighting or identical weighting as illustrated in **Equation 2.9**). Kirsch, Canny, Emboss and Laplacian filters are illustrated in **Figure 2.9**.

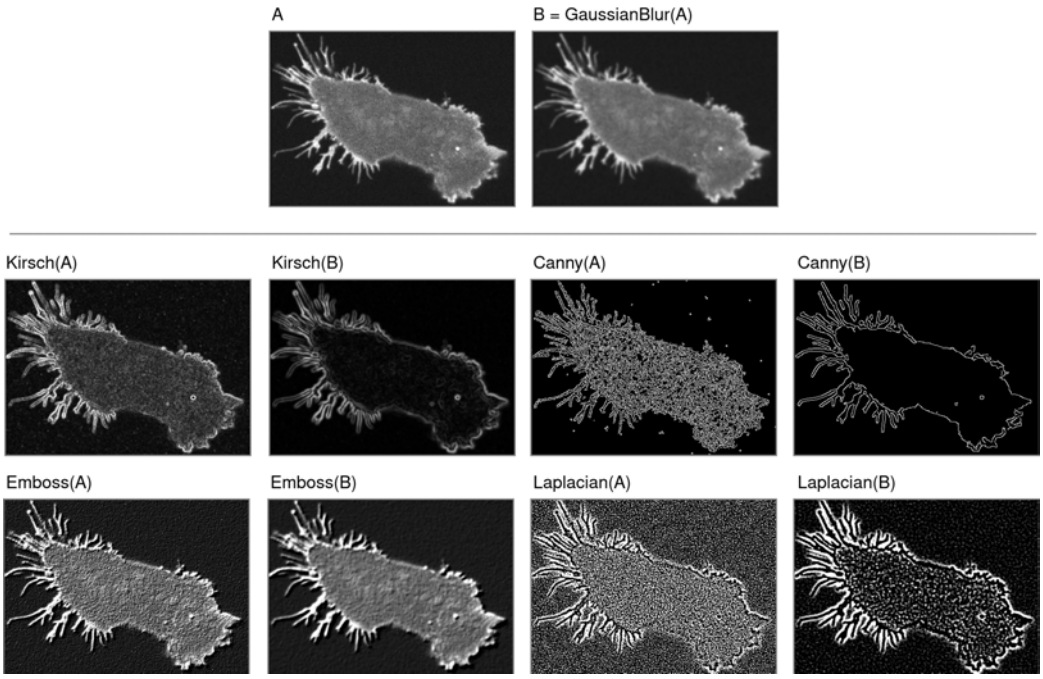


Figure 2.9: **Edge Detection (2)**
Kirsch, Canny, Emboss, Laplacian filters, with and without Gaussian Blur.

$$k_{4n} = \begin{bmatrix} +0 & +1 & +0 \\ +1 & -4 & +1 \\ +0 & +1 & +0 \end{bmatrix} \quad k_{8n} = \begin{bmatrix} +0.25 & +0.5 & +0.25 \\ +0.5 & -3 & +0.5 \\ +0.25 & +0.5 & +0.25 \end{bmatrix} \quad k_{8n} = \begin{bmatrix} +1 & +1 & +1 \\ +1 & -8 & +1 \\ +1 & +1 & +1 \end{bmatrix} \tag{2.9}$$

Note: As demonstrated in **Figure 2.8 and 2.9**, prefacing an edge detection filter with a blur filter (such as Gaussian Blur) can significantly enhance the quality of edge detection, making the search for the optimal combinations of functions even more compelling. This is a generalization that can be applied to any type of function and is the main focus of this thesis.

2.2.6 Mathematical Morphology

Mathematical Morphology [32] is an area of IP that draws inspiration from the physical interaction models between water and earth, leading to the development of morphological filters [33]. Fundamental concepts of these filters include erosion, dilation, and water flow. These morphological operations are typically applied to binary images where the activated pixels ($i \geq 1$) represent the material to be manipulated (the land).

For these convolution-based operators to function effectively, a binary kernel, referred to as a Structuring Element (SE), must be defined. This element can be viewed as the tool used to mold or deform the material. There are three predominant types of SE, illustrated in **Equation 2.10**: the Rectangle SE_R (or Square) which applies a chiseled effect; the Cross SE_C, which applies a sharp effect, especially on corners; and the Ellipse SE_E (or Disc), which imparts a smooth effect.

The size of the SE directly influences the extent of modifications: a larger SE results in more substantial alterations.

$$SE_R = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad SE_C = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad SE_E = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.10}$$

Erosion refers to the process of shaving off pixels that are situated along

the edges of a positive region in an image. To achieve this, the sliding window primarily traverses the negative regions (with the central pixel over a 0), and all positive pixels of the structuring element (SE) are reset to 0 in the resultant image. Consequently, minor positive regions may vanish entirely from the image if the SE is sufficiently large. This makes erosion a suitable filter for eliminating noise or unwanted specks from the background of an image. In the context of geographical terrains, erosion mirrors the effect of rising water levels engulfing coastlines.

Conversely, *Dilation* is the process of expanding positive regions in an image, particularly along the edges. This is accomplished by moving the sliding window exclusively across positive regions (with the central pixel over a 1), and subsequently setting all positive pixels of the SE to 1 in the new image. Among its applications, dilation effectively fills in small negative spaces located within larger positive areas, although it doesn't facilitate the emergence of new regions. Erosion and Dilation are illustrated in **Figure 2.10**.

Note 1: In the context of non-binary images, the process of Erosion and Dilation is equivalent to applying Min and Max filters, respectively.

Note 2: Dilation applied on an image is equivalent to the inverse of the erosion operation applied on its inverse.

The *Opening* operation involves the successive application of Dilation after Erosion on an image, utilizing the same structuring element (SE). This operation essentially removes minor positive regions while largely preserving the larger ones.

On the other hand, *Closing* is the inverse of the opening operation, and

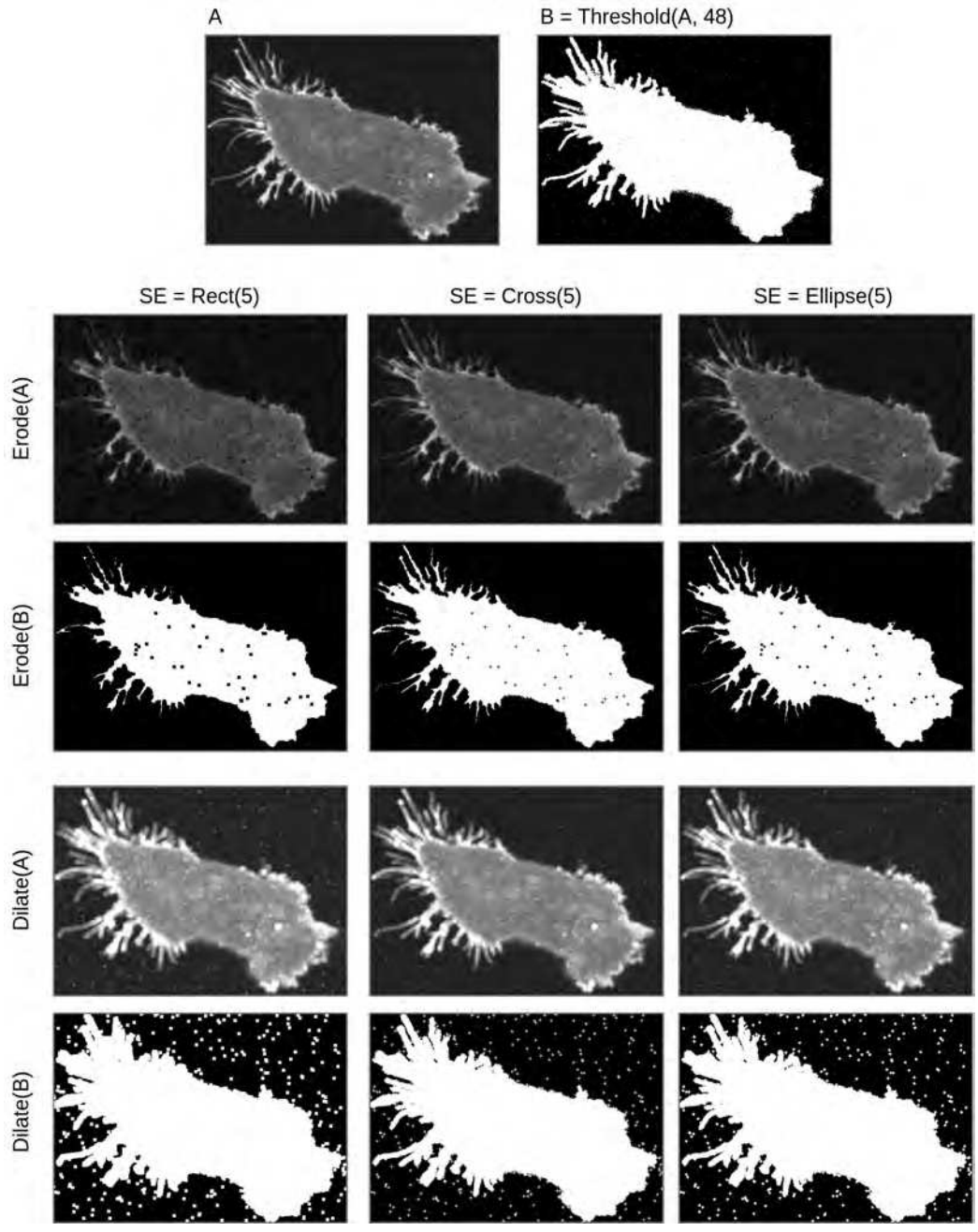


Figure 2.10: **Erode and Dilate filters**
Applied on both grayscale and binary images.

involves the sequential application of Erosion after Dilation on an image, using the same SE. This operation eliminates small holes present within larger positive regions while retrieving the remainder of the image. Open and Close operations are shown in **Figure 2.11**.

The *Morphological Gradient* is a method that performs the difference between dilation and erosion of an image. It is useful for highlighting the edges of objects within the image, as it emphasizes the contrast at the boundaries. *Top Hat* operation, also known as White Top Hat, is a process that subtracts the result of an opening operation from the original image. This operation is used to highlight bright spots of an image on a dark background. *Black Hat*, also known as Bottom Hat, is the opposite of the Top Hat operation. It is calculated by subtracting the original image from the result of a closing operation on that image. Black Hat operation is useful for highlighting dark spots of an image on a bright background. These operations, like others in mathematical morphology, can be combined and applied in various sequences to yield a wide range of image transformations, each useful in its own way depending on the specific needs of the image analysis task at hand.

The *Watershed Transform*, a groundbreaking contribution originating from Mathematical Morphology, was introduced three decades after by Serge Beucher [34]. This method, inspired by the behavior of water flow over terrains, has brought about an innovative perspective to the field of IP. The technique visualizes the digital environment as a topographical surface and involves filling labelled catchment basins, mimicking water basins in a landscape. Two adjacent basins fill with their respective labels until their streams meet. This process then halts along the interface, permanently establishing

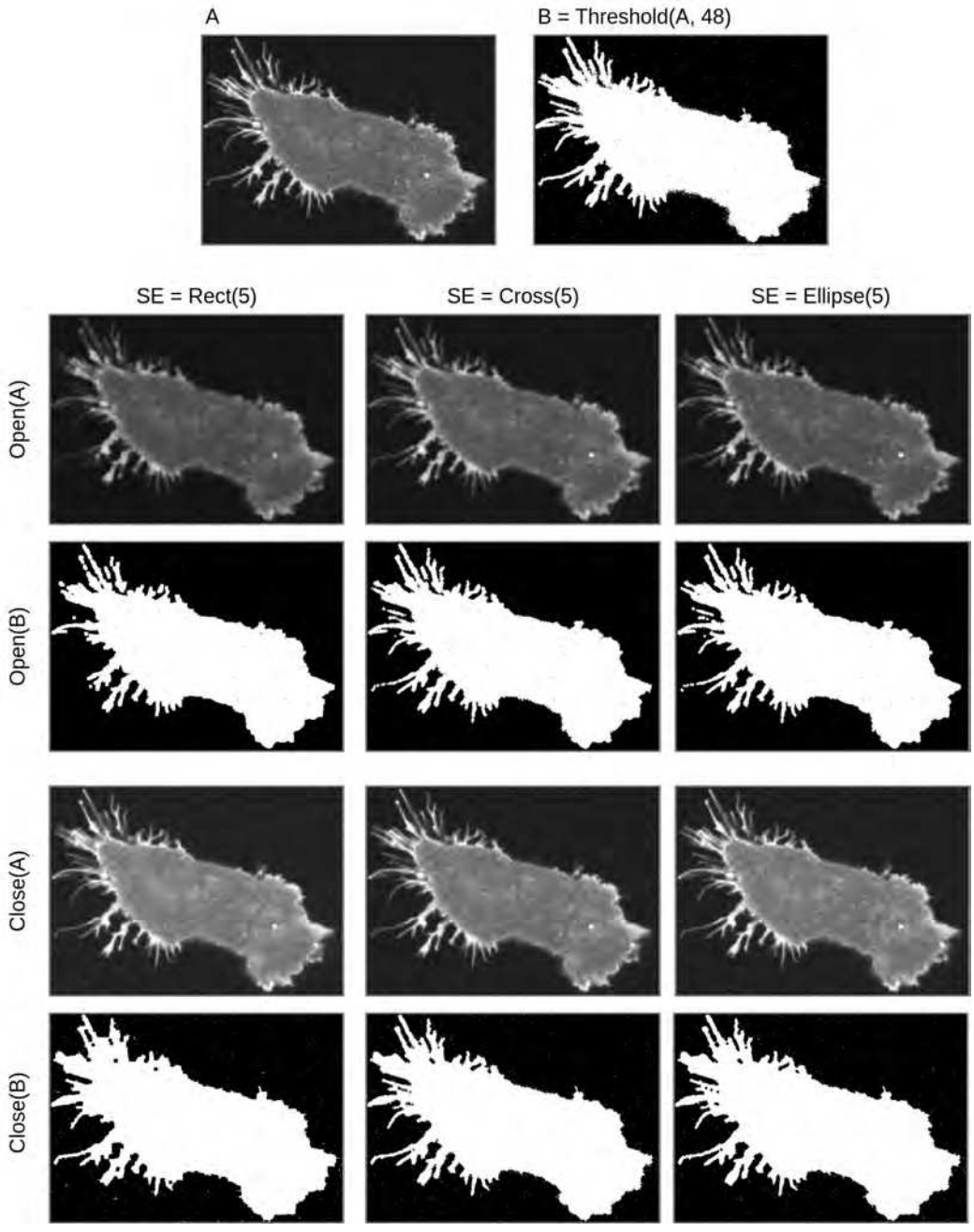


Figure 2.11: **Open and Close filters**
Applied on both grayscale and binary images.

the watershed line. The final outcome is a segmented area with distinct regions demarcated by the watersheds. The Watershed Transform is widely applied as a terminal function, specifically in instance segmentation. It comes in two main variants:

In the classic approach, the water propagation begins from the basin's bottom (local minima), requiring an inference from the topographic surface. This typically involves a sequence of operations: smoothing the image, applying a distance transform to shape the topographic map, and finally propagating from the local minima.

The second approach allows the independent definition of sources from the basins. It calls for a secondary map filled with small labeled areas known as markers (or seeds). Uniquely, this method accommodates multiple labels for a single catchment basin.

Regardless of the approach, the challenge of the Watershed Transform lies in obtaining an effective topographic map or heuristic. Numerous studies [35–37] have proposed algorithms culminating in this function.

Generally, a pre-processing phase is followed by a smoothing operation. There have been efforts to optimize the Watershed Transform topology heuristic using DL approaches [38]. However, achieving an optimal segmentation using Watershed is often challenging due to the required heuristic. This complexity is noted by the authors of *Cellpose* [39], who describe the Watershed Transform as excessively complicated in deriving proper segmentation.

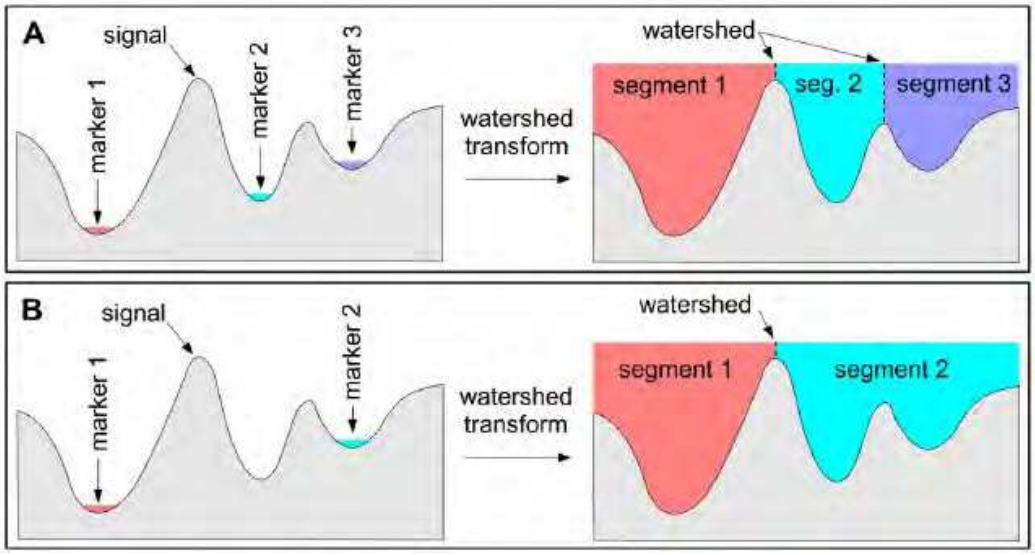


Figure 2.12: **Watershed Transform Illustration**
(A) shows the case where 3 markers are set in signal local-minima, inducing 3 segments. **(B)** illustrates the case where a marker is missing, inducing the loss of one segment at the end of the transform. In CV, the signal is a landscape feature map to guide the flow, also called *Mask* in this work.

2.2.7 Features Descriptors

Feature Descriptors allow us to distill complex images into more manageable high-level features, thereby transitioning from a substantial volume of pixel intensity data to a reduced number of instances. Objects can be linked to a variety of shapes, ranging from basic to complex. Geometric shapes that conform to mathematical equations have an associated parameter space, known as the *Hough Space*. This space is essentially an image, but its dimensions are not spatial (x and y) but relate to the parameters of a given equation. For instance, in the simplest case of the straight line equation $y = ax + b$, the space is described by a and b. An additional space, the accumulator, allows the gathering of pixels that belong to an edge, utilizing two other parameters

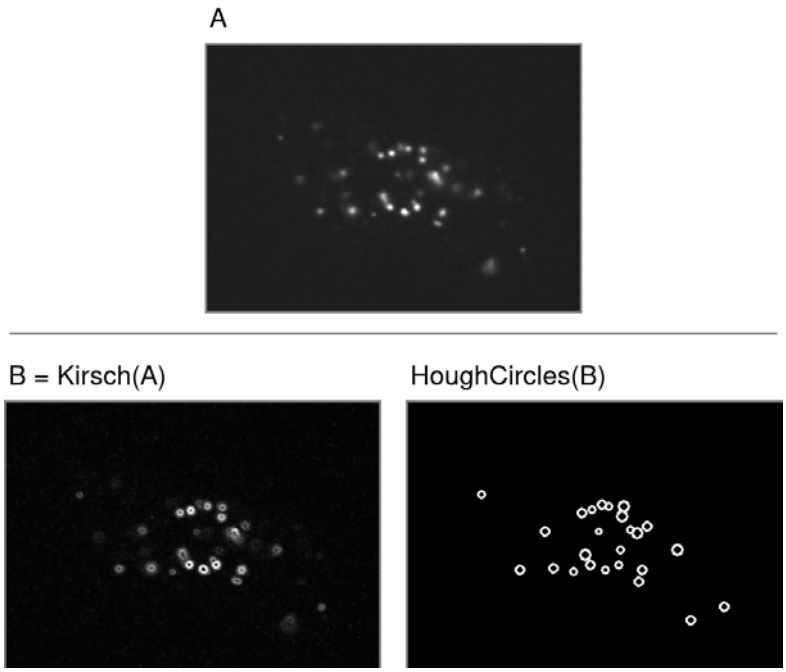


Figure 2.13: **The Hough Circle Transform**
25 circles are detected (white) after Kirsch edge detection filter.

from the Hesse normal form: ρ and θ. A pair (ρ, θ) with a high value at the conclusion of the process indicates a possible straight line passing through numerous points. A local maximum in the accumulator space could represent a line, with (ρ, θ) converted back to (a, b) parameters to derive the equation. This process is known as the Hough Transform [40] published by Duda in 1972 [41], and generalized by Ballard in 1981 [42] and optimized [43] to include more shapes such as circles and ellipses. An illustrative example (see **Figure 2.13**) displays a raw image filtered using the Kirsch filter to delineate edges. The Hough Circle Transform is then applied to identify circles, thereby simplifying 143,360 pixels into just 25 circles.

While the Hough Transform is incredibly effective when the shape to extract from an image is predefined, there are situations when shapes are

more abstract and not pre-determined. In such instances, shape descriptors come into play, examples of which include the *Fourier Descriptor* [44–46], *Shape Contexts* [47], or *Edge Orientation Histograms* [48, 49]. Objects can also be detected via cascades of trained and sampled filters, such as *Harr-like features* [50], as introduced by Viola and Jones [51] and later applied to pedestrian detection [52].

Feature extraction can also be accomplished by detecting keypoints in images, a strategy that proves particularly useful for image registration. There are primarily three keypoint detectors [53]: the *Scale-Invariant Feature Transform (SIFT)* introduced by Lowe in 1999 [54, 55], a faster version known as *Speeded Up Robust Features (SURF)* proposed in 2006 [56], and finally, the *Oriented FAST and Rotated BRIEF (ORB)* which enhanced the previous methods in 2011 [57].

The *Histogram of Oriented Gradients (HOG)* descriptor, introduced by Dalal in 2005 [58], is a feature descriptor used for object detection in CV. HOG involves the computation of local gradient directions or edge orientations in an image, and these are then placed into a histogram. This descriptor counts the occurrences of gradient orientation within localized regions of an image. The resulting histograms form a robust descriptor that can effectively describe the appearance and shape of objects in an image. HOG descriptors are notably resistant to geometric and photometric transformations, thus making them highly versatile.

HOG descriptors are often used in conjunction with *Support Vector Machine (SVM)*, a powerful supervised machine learning model introduced by Vapnik [59]. SVMs are classifiers that can identify complex decision bound-

aries, based on the concept of maximizing the margin between two classes. In the context of CV, the HOG features extracted from an image serve as the input for the SVM, which is then trained to classify the objects in the image. The combined use of HOG and SVM has been shown to be effective in various object detection tasks, such as pedestrian detection [58]. This combination of feature extraction and ML was one of the most popular and effective approaches for object detection and recognition before the surge of deep learning methods in IP.

2.3 Deep Learning

This thesis is primarily rooted in a distinct branch of ML known as GP. However, I consider it vital to introduce and delve into DL, especially its application to images, as it has dominated the state-of-the-art landscape for several years. This choice is also driven by my background in robotics, complemented by significant experiences in CV and DL. My previous work includes the publication of my first conference article as first author [4], along with an award for the best poster presentation during my Master's first-year internship at Imperial College London.

DL, a subfield of ML, has brought massive changes in the field of IP. This revolution traces its roots back to Alan Turing, a pioneering figure in AI, who conceived the idea of machine intelligence mimicking biological evolution and even proposed the first Neural Network (NN) concept known as B-type u-machines [60]. Turing also formulated the Turing Test, which aimed to assess the performance of AI by comparing it to human intelligence.

The evolution of AI has encompassed various stages, beginning with simple automata and progressing to more complex models, largely designed and tuned by humans. In 1959, Samuel [61] coined the term *"Machine Learning"* while training a model to play checkers. This marked a transition towards models that learned from observations, experiences, or examples. The development of AI technologies continued to accelerate, greatly facilitated by advances in hardware capabilities. Increased storage capacities allowed for larger data volumes, while improved computational power enabled more sophisticated algorithms. The panel of these algorithms includes SVM, Artificial Neural Network (ANN), Decision Tree (DT), K-means, k Nearest Neighbors (kNN), and many others.

Among these, DNNs have seen significant development, leading to the contemporary, highly sophisticated spectrum of DNNs. DNNs are a class of ML algorithms inspired by the human brain's functioning, relying on interconnected units or neurons that process and transmit information. DNNs, composed of multiple layers of these neurons, are capable of learning increasingly complex data representations. The learning phase primarily involves configuring the neurons, especially adjusting the weights that connect them. Once these weights are fine-tuned, the network can apply to new data during an inference phase.

The success of NNs stems from the convergence of several mathematical tools. Primarily, the combination of Backpropagation [62] and SGD significantly simplified the learning of the numerous weights within these networks. In 1989, Yann LeCun introduced the concept of assembling and tuning convolution filters by introducing CNN [14], a considerable contribution to image

classification tasks. However, the DL we know today really emerge until 2012, when AlexNet [16] enhanced previous CNN models by adding the Rectified Linear Unit (ReLU) activation function and dropout layers, with max pooling layers following the convolution as a fundamental component [17].

This breakthrough sparked a surge of research in the field, leading to the development of various techniques and architectures. From simple Feedforward Neural Networks used for tasks such as image classification and speech recognition to CNNs designed for image and video processing, and Recurrent Neural Network (RNN)s utilized for Natural Language Processing (NLP) tasks, each ANN type offers unique architecture and strengths. Among the notable advancements in NNs are the development of attention mechanisms, enabling the network to selectively focus on crucial parts of the input, and the advent of generative models like Transformers, enabling the network to generate new data.

2.3.1 Convolutional Neural Network Model

As previously noted, the field of IP has seen significant advancements through the deployment of ANN. The high dimensionality of images, being composed of vast amounts of input values, makes them one of the most complex objects for ML to analyze. The introduction of convolution within ANN [14] led to a major breakthrough, allowing for the learning of filters via backpropagation. A convolution layer, consisting of a set of filters, is represented as a weight matrix. Applying a filter to an image results in an activation map, hence a convolution layer with 'n' filters yields 'n' activation maps. These maps

are then aggregated into a new *Tensor*. **Figure 2.15** visualizes these feature maps [63]. Successive convolution layers enable the formation of increasingly high-level filters, leading to the construction of a CNN.

The convolution layer also serves as a means of dimension reduction, reducing the size of the image during the convolution process. Typically, a convolution layer is succeeded by an activation function such as ReLU, followed by a Max Pooling layer [16, 17] which further reduces the image's dimensionality. For classification tasks, convolution layers are usually followed by a flattening layer that transforms the last activation maps into a single extensive one-dimensional vector. This vector is subsequently processed by several Fully Connected (FC) layers to construct a classifier. The final layer, which usually corresponds to the number of classes, utilizes a *Softmax* (**Equation 2.11**) activation function to produce a statistical dispersion with a summation of 1.

$$\text{Softmax}(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}} \tag{2.11}$$

Thus, a CNN integrates both feature extraction learning and classification in its architecture. A standard CNN model is depicted in **Figure 2.14**. Noteworthy CNN models include LeNet [14], AlexNet [16], VGG-16 and VGG-19 [64], and GoogLeNet [65].

2.3.2 Fully Convolutional Network

Fully Convolutional Network (FCN) [66, 67] are a notable development in the realm of CNNs. They are designed to exclusively use convolutional layers for Semantic Segmentation (SS) tasks. The fundamental structure of an

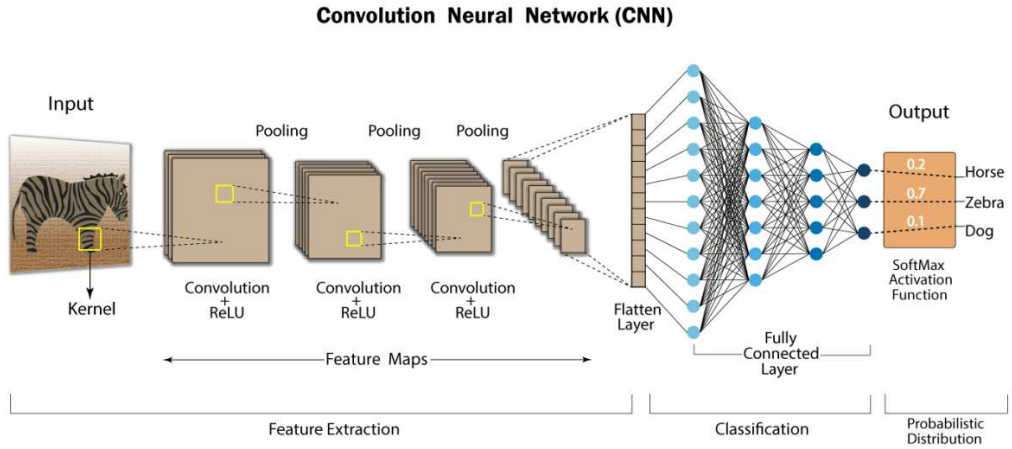


Figure 2.14: **CNN Standard Model: Convolution+ReLU, and Pooling**
Source: https://developersbreach.com/convolution-neural-network-deep-learning/

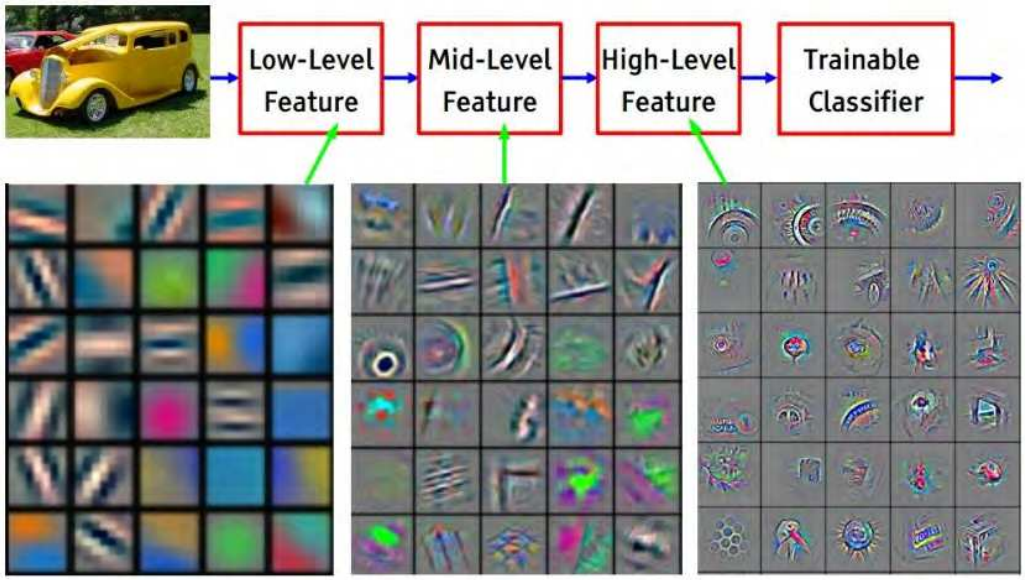


Figure 2.15: **CNN Features Visualization**
Source: Andrej Karpathy slides, CS231n Winter, 2016 [63]

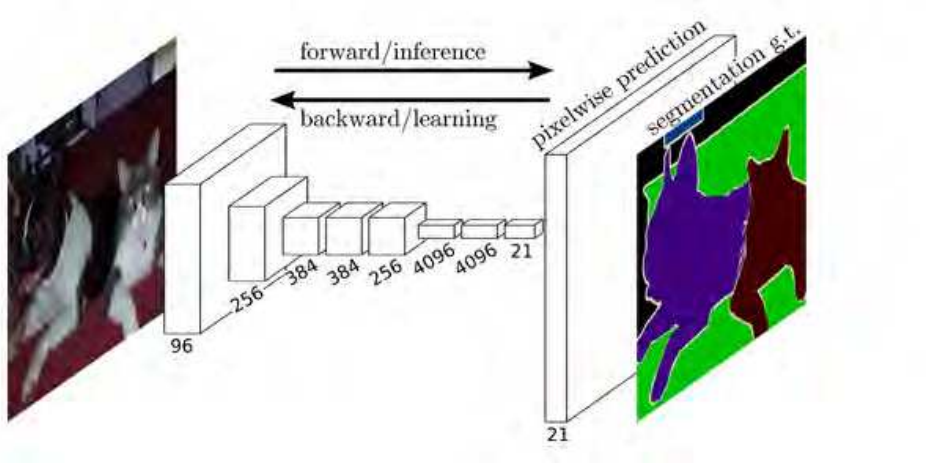


Figure 2.16: **Fully Convolutional Network Model**
Source: Figure 1 [66]

FCN retains the initial downsampling and feature extraction stages that are typical of a CNN. However, the fully connected layers are replaced with an upsampling phase, utilizing *Transposed Convolution* (also known as *UpConvolution*). The strength of this method lies in its ability to restore various features obtained from the network's preceding layers, not just the final one. This significantly enhances the quality of the generated segmentation masks. An illustrative model of an FCN is depicted in **Figure 2.16**.

The development of the *U-net* architecture [68] marked a significant advancement in CNNs for image segmentation tasks. Characterized by its symmetrical structure, U-net proposes a more harmonious architecture that aligns the encoding phase (convolutions) with the decoding phase (up-convolutions), as depicted in **Figure 2.17**. The name "U-net" originates from this U-shaped architecture.

In a departure from previous models, U-net proposed an image-to-image model. Instead of providing a single classification label for an entire image,

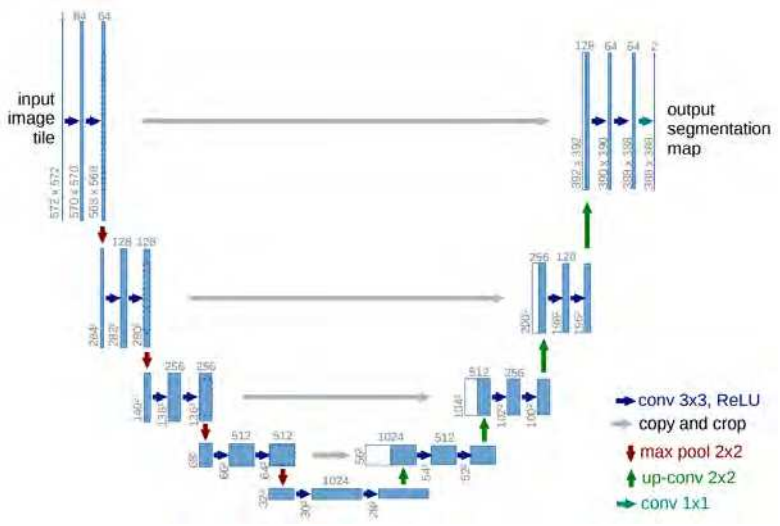


Figure 2.17: **U-net Model Architecture**
Source: Figure 1 [68]

it assigns a "label" (or at least a value) to each pixel within the image. This makes U-net particularly adept at tasks such as Biomedical Image Segmentation, where it continues to be a highly utilized architecture [39, 69].

2.3.3 Residual Neural Network Model (ResNet)

ResNet [70] was introduced in 2015 as an innovative advancement of CNNs. The key motivation behind this new architecture was to address the "degradation" problem [70] - the difficulty in training deep models like VGG [64] models (with 16 and 19 layers) and GoogleLeNet [65] (with 22 layers), which manifested as gradient deterioration with the increase of network depth.

The groundbreaking concept at the heart of ResNet is the introduction of *Skip* or *Residual* connections. Rather than relying solely on the output of one layer as the input to the next, ResNet architectures include direct links from the inputs to the outputs, in effect adding residuals to layer outputs. These

direct pathways, encapsulated within *Blocks*, allow gradients to backpropagate more effectively through the network, addressing the issue of vanishing gradients that tends to afflict deep networks. These ResNet blocks, as shown in **Figure 2.18**, can be built around two primary architectures: a 2-layer-based block known as a *"building block"*, and a 3-layer-based block referred to as a *"bottleneck block"*.

In the same manner as ResNet, *AdaNet* [71] provides another illustration of how residual connections can enhance network performance. Meanwhile, DenseNet [72] presents an alternative architecture that, while using fewer parameters, still outperforms ResNet on the ImageNet benchmark. Various models of *DenseNet*, with 50, 101, 151, and 201 layers, are frequently cited and employed in contemporary research.

Indeed, the advent of ResNet models and the introduction of *Depthwise Separable Convolutions* [73] significantly broadened the field of high-performance image processing, making it accessible to mobile devices with their resource-constrained environments. This has been achieved by devising new models designed for efficient inference on these devices, without sacrificing accuracy.

One of the pioneering models in this regard is *MobileNetV1* [74]. This network employs depthwise separable convolutions, a variant of the standard convolution operation that greatly reduces the computational cost. The model achieved competitive performance on image classification tasks, with a fraction of the computational resources required by previous architectures. This innovation was further extended by subsequent models such as *ShuffleNet* [75], which introduces a channel shuffle operation to improve informa-

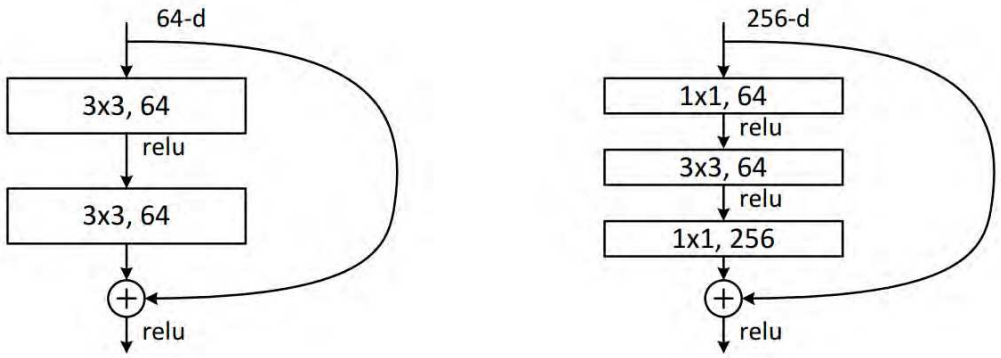


Figure 2.18: **ResNet Model blocks**
Source: Figure 5 [70]

tion flow across layers. Another significant contribution came from *NasNet* [76], which used a reinforcement learning based approach to automatically search for an optimal network architecture.

This was followed by *MobileNetV2* [77], which introduced an inverted residual structure that further improved the efficiency of the network. A similar approach was taken by *MnasNet* [78], which used reinforcement learning for Neural Architecture Search (NAS) but also took into account platform-related information, such as latency, during the search process. The most recent addition to this family of efficient architectures is *MobileNetV3* [79], which combined architecture search techniques with the use of novel operations to further improve performance.

All of these models have been instrumental in enabling the deployment of advanced image processing capabilities on mobile devices, contributing significantly to the democratization of AI technology.

2.3.4 Object Detection Algorithms

The use of CNN for classification has been extended to the localization and detection of objects in the image. *OverFeat* was indeed one of the pioneering models, combining CNNs and a sliding-window mechanism to classify, localize, and detect objects within an image. However, the task of testing a multitude of possible bounding boxes was computationally intensive.

The introduction of *R-CNN* (Region for CNN features) [80] mitigated this issue by using a process known as *Selective Search* [81] to pre-select 2000 candidate boxes. Although this significantly reduced the number of "sub-images" to process, it was still computationally demanding.

Fast R-CNN [82] further optimized this process by using an innovative layer called Region Of Interest (ROI) Pooling Layer for each candidate. This change made it possible for the model to handle bounding box candidates more efficiently. However, the Selective Search process was still a bottleneck in terms of speed.

This final problem was addressed by *Faster R-CNN* [83], which incorporated the candidate proposal directly into the NN through a sub-network called the *Region Proposal Network* (RPN). This network operates directly on the feature maps after the convolutional layers. A variant of Faster R-CNN, named Region-based FCN [84], proposed another FCN model.

Finally, building on all these models, Mask Region-CNN (Mask R-CNN) [85] introduced the possibility of Instance Segmentation (IS) in 2017. This was made possible by modifying the ROI Pooling Layer into the ROI Align layer, enabling mask prediction at a more precise level. This development

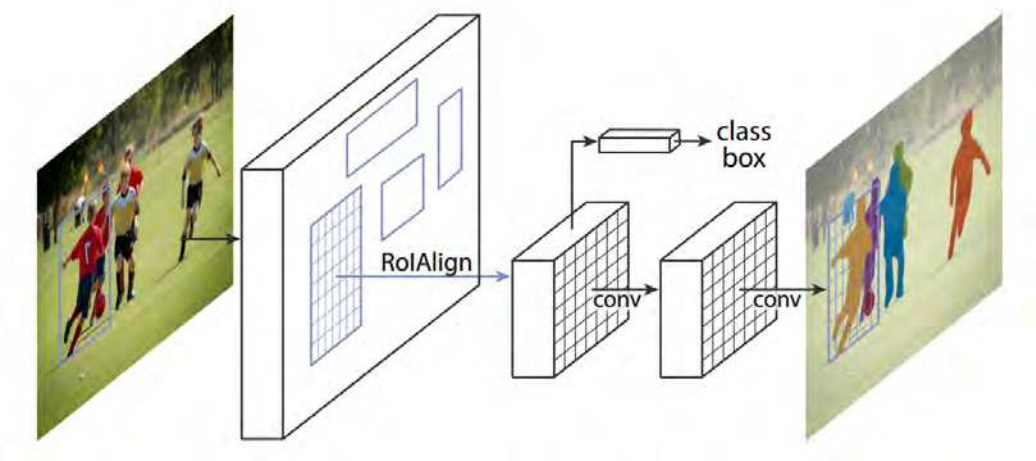


Figure 2.19: **Mask Region-CNN Model Architecture**
Mask R-CNN, Figure 1 [85]

essentially pushed the envelope in terms of object detection and segmentation capabilities, contributing to the state-of-the-art in the field.

In addition to these Region Proposal-based models, some models detect image entities in a single pass without going through each proposed region for classification and bounding box refinement. *SSD*, or Single Shot Multibox Detector [86], operates in a single pass, detecting objects at multiple scales using convolutional layers. It can be seen as a balance between the region proposal-based methods (like R-CNN series) and the *YOLO* series, offering competitive precision with high speed.

On the other hand, the YOLO (You Only Look Once) [87] series has revolutionized the object detection field by introducing a novel concept where the entire image is processed in one forward pass. Its successive versions, including *YOLOv2* (*YOLO9000*) [88], *YOLOv3* [89], and *YOLOv4* [90], have continuously improved upon this framework, boosting accuracy and speed. YOLO's main advantage is its extraordinary speed, making it suitable for

real-time applications.

These single-pass models have greatly advanced the field of object detection, offering efficient, speedy alternatives to the traditional region proposal-based approaches, expanding the applicability of object detection algorithms to real-time systems.

2.3.5 Embedding Models

Embedding models have shown profound impacts in many fields, including NLP, recommendation systems, CV [91, 92], and more. As you've described, these models help in compressing high-dimensional data into lower-dimensional spaces (called *embeddings*), which preserve the semantic relationships between the data points. The *AutoEncoder* (AE) [93] is a popular technique for learning embeddings. It uses an encoder to map the input data into a lower-dimensional latent space and then a decoder to reconstruct the original input from the latent representation. The main goal is to minimize the reconstruction error. The *Variational AutoEncoder* (VAE) [94] adds a probabilistic twist to the Auto Encoder (AE). Instead of learning a single point in the latent space, it learns a distribution. This is done through the introduction of a probabilistic layer that enforces the latent space to follow a specific distribution (commonly a standard normal distribution). The *reparameterization trick* is used to allow backpropagation through this stochastic layer. Variational Auto Encoders (VAEs), unlike standard AEs, are generative models: they allow generating new samples from the learned distribution in the latent space. The resultant embeddings from these models can be used

for various tasks, like classification, anomaly detection, visualization, and more. They are particularly beneficial when dealing with high-dimensional data, as they can capture the essential information in a more manageable form.

2.3.6 Sequential Models

Sequential neural network models comprise a significant portion of the overall landscape of DNN models. Initially conceived as a means to incorporate a memory mechanism and contextual information within the network, these models are specifically designed for handling sequential data. Although not directly relevant to my thesis, a brief introduction to sequential models is warranted, given their potential utility for "Model Induction" (as defined in **Figure 1.2**) using sophisticated NLP techniques. Moreover, these models gained considerable popularity towards the end of my thesis, particularly with the advent of Generative Pre-trained Transformer (GPT) models in the public domain (commonly referred to as Chat Generative Pre-trained Transformer (ChatGPT)).

The premises into sequential models in the realm of DL commenced with the development of RNN [95]. The primary goal of RNN was to process sequential data by incorporating information from the preceding step into the current step, thereby establishing a form of continuity. The RNN model was later extended to be bidirectional [96], signifying the capability to process data not just in a sequential manner but also in reverse order. The idea of "memory" was further enhanced with the introduction of a special

unit, which essentially provided the network its nomenclature: Long Short Term Memory (LSTM) [97]. This unit's objective was to enable the network to learn what information to retain, when to update it, or when to discard it, through a system of gates. An example of bidirectionality with three elements is illustrated in **Figure 2.21**. The utility of this model was demonstrated through its successful application in solving the *T-maze problem* [98]. A different type of gated unit, Gated Recurrent Unit (GRU) [99], was later introduced to minimize the number of parameters employed by the unit, and the two approaches were compared in [100] and are illustrated in **Figure 2.20**.

In the context of NLP, the model's input is typically a series of words. Given the absence of a unified mathematical representation for words, a method to convert words into vector space, referred to as *Word Embedding*, is needed first. This process is generally executed using AE models, with notable examples being *Continuous Bag-of-Words* (CBOW) [101] or *word2vec*, and *GloVe* [102].

2.3.7 Transformers

The introduction of the Attention Mechanism was a significant advancement in sequential models, leading to the development of Transformer models. Transformers were originally conceived to tackle various NLP tasks, including language translation and semantic understanding. A key feature of Transformer models is that they dispense with the recurrence mechanism and instead rely entirely on self-attention mechanisms for drawing global

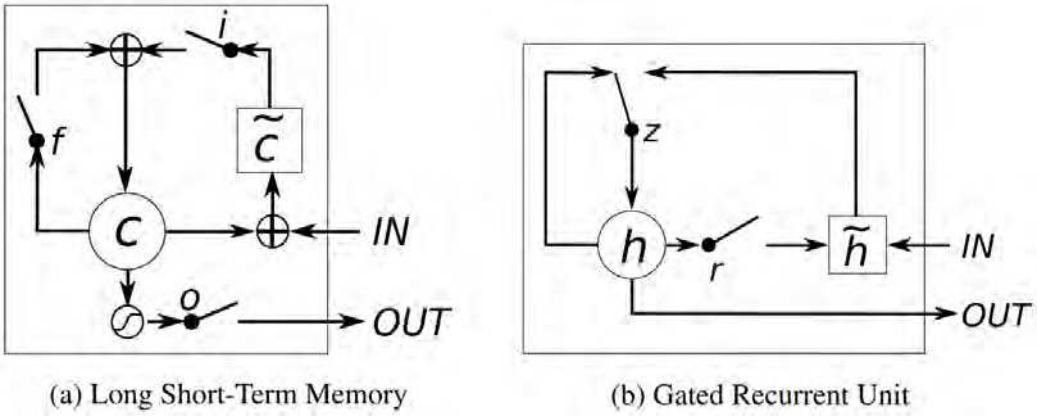


Figure 2.20: **Illustration of (a) LSTM and (b) GRU**
(a) i, f and o are the input, forget and output gates, respectively. c and $\tilde{c}$ denote the memory cell and the new memory cell content. **(b)** r and z are the reset and update gates, and h and $\tilde{h}$ are the activation and the candidate activation.
Source: Figure 1 [100]

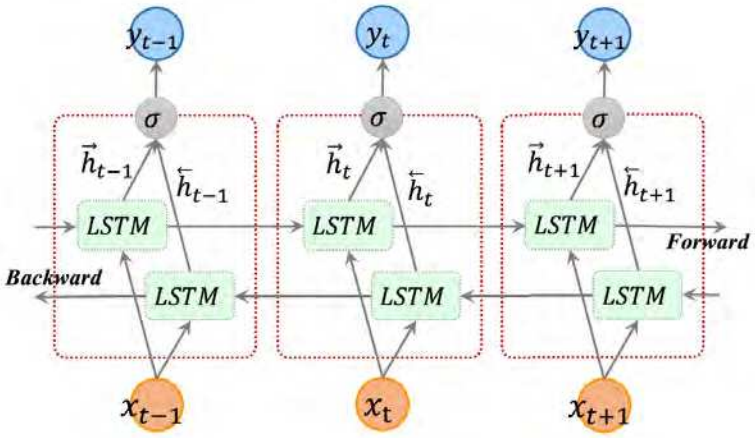


Figure 2.21: **Unfolded architecture of bidirectional LSTM with 3 consecutive steps**
Source: Figure 3 [103]

dependencies between input and output.

The *Transformer* model was first introduced by Vaswani et al. [104] and has since then spurred numerous variations and applications. The *Vision Transformer* (Vision Tranformer (ViT)) [105] was introduced to apply transformer models to CV tasks by treating image patches as a sequence of embeddings: an image becomes a sentence. This approach has further been utilized for SS [106, 107].

Two of the most prominent Transformer models in NLP are Bidirectional Encoder Representations from Transformers (BERT) [104] and GPT. BERT is bidirectional, providing a more comprehensive understanding of the language context. On the other hand, GPT model series are designed to generate human-like text and have found extensive use in various generation tasks like agent conversation.

In recent years, the *HuggingFace* library [108] has become a popular choice for implementing Transformer models, providing a wide variety of pre-trained models and tools. Additionally, the *Xplique* model [109] is a notable example of applying Transformers to provide explanations of ML predictions, highlighting the versatility of these models.

2.3.8 Biomedical Segmentation Models

DL models have revolutionized the field of biomedical and clinical [110–112] image analysis, aiding in accurate inference from complex biomedical data. These models provide robust and efficient methods to segment structures of interest, such as cells, tissues, and organs, from medical images.

Mask R-CNN [85] has been employed for instance segmentation in various biomedical contexts. By generating high-quality segmentation masks for each instance, it is well suited for applications that require distinguishing and separating individual objects within an image.
The U-Net architecture [68] is another foundational model in biomedical image segmentation. The model is characterized by its symmetric expanding path, which captures context and enables precise localization, making it exceptionally effective for tasks like tissue segmentation.

Cellpose [39] is a generalist algorithm for cellular segmentation in microscopy images. It employs U-Net backbone to perform cell instance segmentation. Cellpose is equipped with a unique feature: it sizes cells/objects in an image using an automated approach, making it scalable and applicable across a broad range of applications. At its core, Cellpose utilizes the concept of "flows". A flow is a vector field that points towards the center of an object from all locations on the object's boundary. In this sense, Cellpose can be seen as a two-step process: First, it computes the flows, and then it uses these flows to find the cell instances using Gradient Backtracking. The model is trained to predict these flows, and post-processing methods are applied to convert flows to labels. One significant advantage of Cellpose is its generalization capacity given by the image style embedded in the model. It's been designed and trained with diverse datasets, making it versatile in handling various cell types, magnifications, imaging modalities, and even 3D images.

StarDist [69, 113] is a U-Net-based model designed specifically for the

purpose of cell nuclei segmentation in microscopy images. The model's distinctive feature is its use of a "star-convex polygon" shape prior for cell nuclei, which are often star-convex in microscopy images. The original StarDist model (2D StarDist) [69] was designed for 2D images, and it outputs star-convex polygons as the segmented objects. It uses a two-output CNN: one output for object probabilities and another for the distances to the object boundary in multiple directions (star-convexity). To extend the approach to 3D images, the authors introduced 3D StarDist [113] which is similar in spirit to the 2D version but uses spherical coordinates instead.

SplineDist [114] is an advanced, robust image segmentation technique, specifically designed to tackle challenges faced in the precise delineation of intricate and convoluted biological shapes. It builds upon the foundational principles of StarDist, which uses polygonal shapes represented by star-convex polygons for object segmentation. Where SplineDist differs is in its ability to handle more complex and curved shapes, thanks to its utilization of B-spline representations instead of simple polygons. B-splines are versatile mathematical tools that can be used to represent complex, irregular curves and shapes with a high degree of precision.

CPP-Net[115] is another variant of the StarDist model designed to make predictions more robust.

HoVer-Net [116] is a revolutionary NN architecture explicitly designed for the simultaneous segmentation and classification of nuclei in histopathological images. The model is based on ResNet backbone and uses Horizontal and Vertical (hence the Ho-Ver) flows as Cellpose but the post-processing step results in using Marker-Controlled Watershed

nnU-Net [117]: nnU-Net is a self-adapting method for biomedical image segmentation includes automated preprocessing, training and postprocessing steps based on the data.

2.3.9 Limitations

While DL models have demonstrated remarkable capabilities in numerous applications, they come with their fair share of limitations. One of the most critical issues is the high computational cost and significant energy consumption associated with training and deploying these models. DL models often require powerful hardware resources, such as high-end Graphical Processing Unit (GPU)s or even clusters of servers, which can make them inaccessible or impractical for many users or small-scale projects.

Moreover, DL models typically require a vast amount of data to train effectively. Collecting and properly annotating such large datasets can be challenging and time-consuming. This reliance on massive datasets also raises questions about privacy and data security, as sensitive information may be needed to train models in fields like healthcare.

In the context of limited annotated data, two primary strategies have emerged to enhance DL model performance: generative models and data augmentation. Generative model, such as Generative Adversarial Network (GAN), VAE, and others, can create synthetic data from the original training examples. These models learn the underlying distribution of the input data and can generate new instances that respect this distribution. On the other hand, data augmentation offers new training samples by applying various

randomized transformations to the existing data. These transformations can include rotation, scaling, flipping, cropping, noise injection, and more. These approaches help to increase the diversity and quantity of the training data, making the model more robust to variations in the input data and reducing overfitting.

A further issue is the lack of interpretability and transparency of DL models, often referred to as the "black-box" problem [7]. While these models can make accurate predictions, understanding how they arrive at their decisions can be extremely challenging. This lack of transparency makes it difficult to identify and correct biases, errors, or other issues in the model. It also poses problems in scenarios where interpretability is crucial, such as in medical diagnostics or autonomous driving, where understanding the reasoning behind a decision may be as important as the decision itself.

To overcome these limitations, we chose to use GP, which has a better solution in terms of XAI [118], and more specifically CGP, recently already qualified as a "white-box" in the medical field [119].

2.4 Objectives and direction of the thesis

As mentionned in **Chapter 1**, the field of AI, despite these significant achievements, is still facing many challenges, particularly in terms of interpretability, frugality, and environmental impact. This motivates the development of a system that is not only efficient but also transparent, frugal, and eco-friendly.

Regarding XAI, most of the current AI systems operate as "black boxes".

Therefore, a key motivation behind my project named *Kartezio* is to create an AI model that can be easily interpreted, providing clear explanations for its outputs. Frugality is particularly important considering the increasing demand for AI systems and the consequent significant increase in energy consumption. and carbon emissions. The computational resources required by large-scale AI models contribute significantly to carbon emissions, making it crucial to develop more eco-friendly systems. The demand for hardware is also a major constraint when working in environments such as the CRCT or the IUCT.

In view of the background presented throughout this Chapter, DL, although at the forefront of the state-of-the-art, does not meet most of the criteria and is ultimately not very accessible to non-expert users. This work focuses on IP models, and more specifically on IS for biological entities: particles, cells, tumor regions... These are simple objects that can be processed with filters (see filters presented in **Section 2.2**) assembled by a human, and can therefore don't necessarily require large ANN models. This is absolutely the whole point of Kartezio: to enable the creation of these IP algorithms without using any intrinsic and cumbersome DL components. The model thus produced is of the same form and shares the same properties as a human-made model.

In this context, we have taken over the direction of GP and more specifically CGP for the generation of IP programs. By adapting its form using similar ideas from recent DL models, we have made it possible to segment instances of various entities, on different formats (even 3D) and microscopy scales. In the following Chapter, I introduce *Kartezio*, the proposed model

and I demonstrate that it is fully comparable to recent ANN models for cell segmentation, although in its most basic form. More importantly, it has the advantage of massively overcoming the weaknesses of DL presented above. Kartezio has been tested on four *Use Cases* directly related to Immunology studied within my team (see **Figure 2.1**) and presented in the article published in Nature Communications (see **Chapter 3**).

Chapter 3

Kartezio: A Darwinian Designer of Explainable Algorithms for Biomedical Image Segmentation

In this Chapter, I introduce Kartezio, the main project of my thesis, published in **Nature Communications** journal.

I first give a concise retrospective of my project and how we ended up using CGP and the choices we made to enable IS and 3D images process. I then introduce and develop the history of CGP and its variants, particularly around IP, CGP being the only component that brings the ML dimension to the optimization of the program that performs the processing of images. Then, I include the article submitted to **Nature Communications** (main text, material and methods, extended data) which presents the Kartezio model and 4

diverse application cases. Most of the experiments presented in the paper are already over a year old, but have been kept constant to maintain consistency. This has enabled me, in the meantime, to think about possible extensions to this central work. The most advanced works, intended for GECCO 2024 conference submission, are presented in **Chapter 5**, and some preliminary of reflection (tested or untested) are shortly mentioned in **Chapter 6**.

3.1 Genesis of the project

The genesis of Kartezio comes my first attempts at biological image analysis done by hand, especially with the OpenCV [120] library, on small datasets. The segmentation produced was of good quality, where using NN models would have been difficult with the absence of labeled annotations. The main motivation was to automatically generate these same algorithms which could sometimes be difficult to parameterize. For that, the first idea, and which remains the founding idea of all my work, was to produce the segmentation algorithm starting from the end. That is, if cell segmentation is the goal, I first impose a function producing mask instances, for example Watershed Transform [34]. Then, in case of Marker-Controlled Watershed, I try to feed the Watershed with 2 inputs: the topological mask and the markers mask, and so on until I get to the input channels that can be interesting. The construction of the algorithm is thus guided by my knowledge in CV and filters to assemble correctly the functions: use a blur filter before a threshold, do not use 2 binary thresholds in a row, etc. We first thought of growing the algorithms from the end, in the form of trees with rules, like the *L-Systems*

introduced by Lindenmayer, in 1968 [121]. This method, inspired by growth of plants, proposes an algorithm that iteratively builds trees using simple rules, from the root (output) to the leaves (inputs, terminals). Writing the rules and a grammar then becomes the basis for generating the algorithm. The major constraints are to manage the explosion of the number of nodes (aka. bloating effect), the repetition of patterns (fractal effect) and the condition to reconnect to an input. It is also constrained by the final algorithm produced that necessarily corresponds to a tree, while a graph could be more versatile. Another approach inspired by biology and natural processes is the EC that groups optimization algorithms inspired by the Theory of Evolution of Charles Darwin [122]. Kartezio is more precisely based on a sub-branch that optimizes programs: GP[123] with a special encoding/decoding allowing to evolve programs under the form of a Directed Acyclic Graph (DAG) with CGP [124].

Drawing direct inspiration from recent NN models, the similarities in the architectures become evident: preprocessing - backbone - postprocessing, where the backbone can represent any type of model. This is illustrated proficiently through models such as HoVer-Net[116], Cellpose[39], and Stardist[69], which utilize a backbone (ResNet or U-net) that generates heuristic maps as gradients. These maps are immediately evaluated and form part of the loss function, enabling the network to learn. However, during the learning stage, there is no assessment of the quality of the IS. Rather, the evaluation is solely based on the heuristics used to derive it. There is then a post-processing (Watershed Transform, Gradient Backtracking, Non-Maximum Suppression) phase to obtain the final labels in the form of segments. In this setup, the

backbone essentially serves as a heuristics optimizer. This was the initial inspiration for Kartezio: to replace the backbone with a series of functions that do not rely on NN.

The initial tests were conducted by training one model to perform SS to obtain the landscape map, while simultaneously training a second model to locate markers pointing to each instance. With these two independent models, it is then possible to implement a Marker-Controlled Watershed in post-processing and produce an IS map. This approach was preliminarily used in a study by Filali et. al. of which I am co-author [2]. With the publication of the Cellpose article [39] in January 2021, we had the opportunity to compare our model with a state-of-the-art model, particularly as the authors shared a dataset of 89 images on which they trained their specialist model, fitting the conditions of Kartezio's current implementation. Cellpose's specialist model achieved an AP50 score of 0.91 on the test set, which we used as a benchmark to evaluate Kartezio. Initially, models designed as a pair achieved an average score of just over 0.7. The main reason for this is that the landscape map was evaluated using an Intersection over Union (IoU) fitness metric, and there was no metric to evaluate the quality of the gradient, which is also important for good propagation with Watershed Transform [34].

Hence, we embarked on the combination of several models in the form of an ensemble by averaging predictions and employing a smoothing blur. Through this process, the topology of the landscape map gained relevance with distinct gradient, and the noise within the marker maps could be reduced. Consequently, the Watershed Transform became more pertinent and robust. With ensemble sets composed of 6 landscape models and 6 markers models, we

achieved a score of 0.85 (utilizing only 25% of the initial dataset), enabling us to surpass Mask R-CNN and Stardist. However, it's worth noting that having 12 algorithms combined through averaging may add complexity to interpretation and explanation efforts.

In the end, the most potent and meaningful models were obtained by directly evaluating IS. This was accomplished by incorporating the Watershed Transform directly into the final program and freezing it, leading to the emergence of the **Endpoint**. This final function, which doesn't belong to the CGP graph and does not evolve, can thus break certain restrictions (like returning multiple outputs or returning a probability). In this manner, the Watershed Transform is directly connected to the graph, which now has two outputs (landscape and markers). It is through evolution via CGP that the model optimizes the heuristics to produce the best IS direcly. This became possible by adapting the *a posteriori* evaluation metrics (AP50) from Stardist and Cellpose authors, optimizing them with Numba [125], and integrating them as a fitness function for evolution. Training with this method takes a longer time but remains within an acceptable range.

The Endpoint is a major addition to extend Cartesian Genetic Programming for Image Processing (CGP-IP) to IS. In the same way, we've added CGP's upstream preprocessing block to adapt to various image formats. The Kartezio model is illustrated in **Figure 3.2**. Interestingly, the idea of sets will be retained, notably for the creation of heatmaps on histology slides (see **Figure 3.4**). It is also the concept of merging predictions that will give rise to the Aggregation of predictions in 3D context (see **Figures 3.6 and 3.7**).

3.2 Cartesian Genetic Programming

Drawing inspiration from Charles Darwin's Theory of Evolution [122], a range of algorithms have been developed that emulate the evolutionary process within a population over successive iterations. These EA incorporate principles such as natural selection and survival of the fittest, and use them as the basis for optimization and problem-solving strategies.

Alan Turing was one of the first to conceive the idea of adapting Darwin's theory for application to machine intelligence [60]. He envisioned a process wherein machines could learn and adapt over time, guided by principles similar to the evolutionary forces acting on biological organisms. This primary idea has laid the groundwork for the evolution of numerous adaptive systems and learning algorithms in AI.

3.2.1 Evolutionary Computation

The field of EC includes a range of algorithms that are fundamentally rooted in the concept of species evolution. The primary objective of these algorithms is to optimize the individuals within a population in a manner that effectively solves a specific problem. The sub-domain of EA is distinguished by its implementation of an artificial selection process. By mimicking the mechanism of Natural Selection, it becomes possible to evolve individuals over multiple generations by **Selecting** those that best meet the set criteria (**Fitness**), reproducing them through **Crossovers**, and introducing random modifications via **Mutations**.

John Henry Holland further refined this concept by introducing the GA that

builds upon a genomic representation. In this context, a **Genotype** is devised (usually in the form of a string, integer, or floating-point vector), where each individual value is referred to as a **Gene** and encodes a piece of information. The nature of the problem at hand and the method of genotype decoding determine the specific sub-branch of GA that is involved.
In the context of this thesis work, the focus was on creating and implementing IP algorithms, which falls within the domain of GP. This approach is particularly suited for the design of highly adaptable solutions in the realm of CV and image analysis.

3.2.2 Genetic Programming

GP, first introduced and patented by Koza in the 1990s [123, 126], is designed to evolve computer programs. It can take 3 main different forms as detailed below. Other GP versions also exist, such as grammar-based GP [127] for rule evolution, or PushGP [128, 129] allowing multimodality using a stack-based architecture with its own language "Push".

Tree-Based Genetic Programming

The original model of GP, often referred to as Tree-based Genetic Programming (TGP), conceptualizes programs as tree structures where nodes represent operations or functions, and branches denote the flow of data or control. This representation aligns with the hierarchical nature of many computational tasks, facilitating the development of solutions that can be easily interpreted and modified.

Linear Genetic Programming

An alternative representation is offered by Linear Genetic Programming (LGP) [130, 131], which models programs as sequential lists of instructions. This approach is akin to traditional procedural programming, providing a straightforward path for the execution of operations.

Graph-Based Genetic Programming

Graph-Based Genetic Programming represents a significant evolution in the field of genetic programming, introducing a versatile and powerful approach to the automated design of algorithms. Unlike its predecessors, TGP and LGP, which respectively rely on hierarchical tree structures and linear sequences of instructions, Graph-Based GP utilizes a directed graph as the fundamental structure for representing programs, as depicted in **Figure 3.1**. This paradigm shift from linear and tree structures to graphs enables a more nuanced representation of computational processes, reflecting the inherent complexity of real-world problems more accurately. This thesis primarily focuses on the CGP variant, a branch introduced by Julian Francis Miller in 1999 [124], which represents programs using this flexible graph-based approach. The ability of CGP to generate complex structures from simple nodes makes it particularly suitable for designing innovative solutions across a variety of computational challenges.

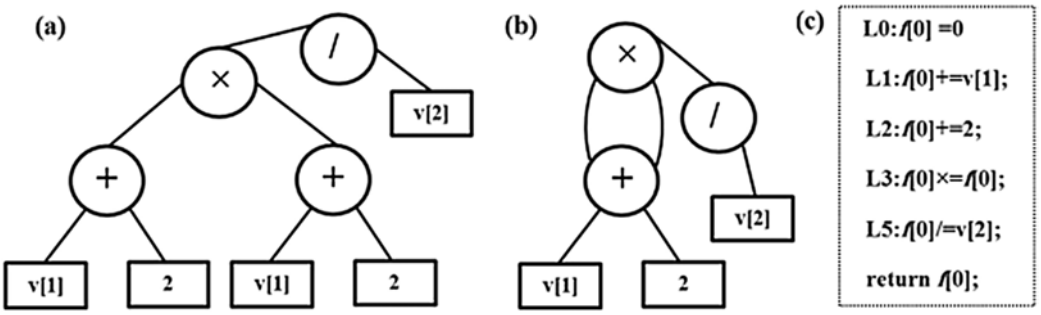


Figure 3.1: **Comparison of GP program structure**
"A comparison of a program structure evolved by **(a)** tree-based, **(b)** graph-based GP, and **(c)** linear GP for the expression of
$y = f(v) = \frac{(v_1+2)^2}{v_2}$"
Source: Figure 2 [132]

3.2.3 Cartesian Genetic Programming Fundamentals

Inspired by the principles of circuit design, the foundations of what would later be termed CGP were published by Miller and Thomson [133]. The term "Cartesian Genetic Programming" was officially introduced in 1999 [124]. CGP builds upon the concepts of GP [123, 126] (see **Section 3.2.2**), by adopting a DAG based representation of programs. In this representation, nodes symbolize operations, and these nodes are derived from a vector known as the genotype. Initially, the nodes were organized into a two-dimensional grid (rows and columns) – hence the term "Cartesian". Each node could only be connected to another node in a preceding column. However, this grid concept was later abandoned in favor of using a single row, thereby defining the genotype's "dimension" by the number of nodes alone. The directed graph is obtained by connecting the output nodes to the inputs through functional nodes, making the resulting graph not necessarily a tree. Another important characteristic is the graph's acyclic nature, due to each node being always

connected to a preceding one, thereby eliminating the possibility of intrinsic loops.

Within the graph, certain nodes may not belong to the resultant structure and can therefore be disregarded. The graph that remains can be termed the "active" graph, comprising only the "active" nodes. In the domain of GP, a known phenomenon called "bloating" [134, 135] can occur, which denotes an increase in the number of nodes, and hence operations in the evolved program. Interestingly, CGP does not suffer from this bloating phenomenon [136].

Typically, CGP employs a $1 + \lambda$ ES and does not utilize the Crossover operation, although a few methods have been proposed [137, 138].

Genotype Representation

The Genotype in CGP is a numerical vector typically of fixed size. This vector encodes both functional and output nodes. Both integer-based and floating-point-based genotypes can be found in literature, with the latter having values ranging between 0.0 and 1.0 [5, 137, 139].

The nodes within the genotype are represented as sub-vectors of equal size, which are broken down into: a) the ID of the function used, b) the nodes that serve as inputs for the given node, and c) any additional parameters required for the function. The genotype is completed by the output nodes, which point to either functional nodes or direct inputs. While the genotype can be easily modified by changing any of its values, the implications of such modifications only become clear when the genotype is decoded into a phenotype.

Phenotype

The phenotype is the practical expression of a genotype. In the case of CGP, the phenotype is a DAG that is assembled, connected, and parameterized based on the nodes specified by the genotype. As noted by Miller [140], decoding a genotype is a straightforward process that begins from the output nodes.

The Decoding Operation

The decoding operation in CGP starts with each output node pointing to a functional node. Each functional node, in turn, points to one or more preceding nodes depending on the arity of the function in question. This operation is recursively repeated until input nodes are reached, which eventually results in the final active graph. The combination of an adapted decoding method and a novel genotype encoding can give rise to new CGP models.

Library and Primitives

In standard CGP, a **Library** is a collection of functions, also referred to as **Primitives**. All primitives are expected to adhere to certain rules, such as producing outputs of the same type and dimensionality as their inputs. For instance, in a model operating with u8-encoded scalars, an *'addition'* primitive should return a u8-encoded scalar. In the context of vector data, the dimensionality of the processed vector must remain unchanged to prevent either complete decimation (reduction to a scalar or even an empty set) or an unexpected increase in the vector size. Some models can accommodate

vectors of different dimensions; for example, adding a vector of size 5 to a vector of size 7 may yield a new vector of size 5 by adding only the first 5 elements [5, 141]. Importantly, primitives must always be deterministic – given the same inputs and parameters, they must always produce the same output.

One way to adapt the model initially developed for electronic circuits is to change the data type of the primitives. This is particularly evident in models developed for image processing, where the library of functions comprises filters or operations on images (typically taken from the OpenCV Library [120]). This has resulted in variants of CGP such as CGP-IP [142–144] and Automatic Learning of Image Filters using Cartesian Genetic Programming (ALIF-CGP) [145], which restricts its operations to binary and morphological operators.

Mutations

Mutation plays a fundamental role in GP and CGP, as it serves as the primary mechanism for altering genotypes during evolution. In CGP, point mutation of a gene involves changing its value. This could involve modifying connections, the function, the function parameters, or the direct output connections. Mutation rules are implicitly directed and constrained by the genotype as well as the type of model. For instance, in the case of Recurrent Cartesian Genetic Programming (RCGP) [146–148], connection mutation rules can be altered to permit connections to subsequent nodes in the graph, necessitating an adaptation of the decoding phase.

Another approach to mutation in CGP involves performing operations di-

rectly based on the phenotype. Kalkreuth [149], for instance, proposed two operators that can activate an inactive node by reconnecting it (**insertion**) or render a node inactive by disconnecting it (**deletion**), with a certain probability. While this approach maintains a fixed genotype size, Fang [150] suggested implementing these operators directly on the genotype without considering the phenotype. This method, named Frameshift Mutation Cartesian Genetic Programming (FMCGP) [150], draws inspiration from biological DNA through the concepts of *substitution* [151] and *Frameshift mutation* [152]. A similar approach is proposed for Genetic Improvement [144], where a node can be removed from between two other nodes, reconnecting them: $A \to B \to C$ becomes $A \to C$. Conversely, a node can be inserted between two others: $A \to B$ becomes $A \to C \to B$. Nodes can also be added or deleted at the extremes of the graph. These mutations can be performed on a variable-size genotype, generating very small or extensive graphs.

Mutations Strategies

In addition to the specific effects of a mutation, various strategies can be employed. Miller summarizes these strategies [153], building on the work of Goldman and Punch [154]. The initial aim of these strategies is to minimize the evaluation of genotypes that have phenotypes identical to those of previous generations. In other words, if the effects of mutations do not influence the phenotype (active graphs and parameters), the predicted outputs and hence the fitness obtained will remain the same. In this manuscript, I propose to view mutation strategies as mutations in themselves. That is, as a *wrapper*, which permits the combination of any type of strategy with any

type of mutation, or even to nest strategies. This approach offers a flexible and potent tool for controlling and guiding the evolutionary process.

3.2.4 Cartesian Genetic Programming Models

Name	Acronym	Publications	ES	Decoder	Libraries	Level Back
Cartesian Genetic Programming	CGP	Miller, 1999	$1+\lambda$	Sequential	Scalar	$[0, n_i]$
Self-Modifying CGP	SM-CGP	Harding, 2007	$1+\lambda$	**Hard-iterative**	Scalar, **Alterations**	$[0, n_i]$
Recurrent CGP	RCGP	Turner, 2014	$1+\lambda$	**Soft-iterative**	Scalar	$[\mathbf{0}, \mathbf{N}]$
Differentiable CGP	dCGP	Izzo, 2016	$1+\lambda$	Sequential, **Weighted**	**Taylor Trunc. Poly.**	$[0, n_i]$
Filter optimisation with CGP	-	Harding, 2008	**Tournament**	**Soft-iterative**	Scalar	$[0, n_i]$
CGP for Image Processing	CGP-IP	Harding, 2012	$1+\lambda$ (Islands)	Sequential	**Array (High-Level)**	$[0, n_i]$
Mixed-Type CGP	MT-CGP	Harding, 2012	$1+\lambda$	Sequential	**Scalar, Vector**	$[0, n_i]$
CGP for Instance Segmentation	CGP-IS	Cortacero, 2023	$1+\lambda$	Sequential, **Endpointed**	**Array**	$[0, n_i]$
CGP for 3D images	CGP-3D	Cortacero, 2023	$1+\lambda$	**Soft-iterative, Endpointed**	**Array**	$[0, n_i]$

Table 3.1: Panel of CGP variants

Recurrent CGP

Turner introduces the concept of recurrent connections within CGP, allowing feedback within the graph with RCGP [146]. This gives the nodes the ability to refer to nodes that come after them in the genome sequence, unlike traditional CGP where a node can only refer to inputs or previous nodes. This creates potential cycles or loops within the graph, which can be very useful for problems that require feedback or have a temporal aspect, such as time-series prediction, dynamic system modeling, and certain control tasks.

Mixed Types CGP

Working with primitive types has resulted in the development of a new model type: Mixed-Types Cartesian Genetic Programming (MT-CGP), introduced by Harding [141]. This model is based on polymorphism, allowing for the

integration of multiple types within the same graph. Each primitive can be overloaded to handle possible type combinations and return different types. To illustrate, let's consider an MT-CGP model with scalar type (s) and vector type (v), and the primitive *add* that requires two inputs ($\alpha = 2$). First, all different function signatures need to be defined: $add(s, s) \rightarrow s$, $add(s, v) \rightarrow v$, $add(v, s) \rightarrow v$, and $add(v, v) \rightarrow v$. Indeed, this results in $2^{\alpha=2} = 4$ signatures per primitive for only two types. If we want to add a third matrix type (m), we would need $3^{\alpha=2} = 9$ signatures per primitive, which quickly becomes laborious to construct.

In **Chapter 5**, I propose a variant of this "Mixed-Type" model, which is initially based on function signatures and tackles the exponential complexity of adding more types. This new model not only simplifies the management of various types but also significantly extends the range of problems that can be addressed within a single graph, thereby broadening the scope of CGP applications.

Self-Modifying CGP

Self-Modifying Cartesian Genetic Programming (SMCGP), a variant of CGP first introduced by Harding [155], expands the model's capabilities by allowing the genotype to dynamically modify itself during the execution of its phenotype. In the traditional CGP, the genotype remains unchanged throughout the execution, while in SMCGP, specific functions or operations in the function library are capable of altering the genotype. These self-modifying functions can rewrite parts of the genotype, either adding/modifying/removing the functionality of existing nodes or changing the topology of the graph.

Balanced CGP

The Balanced Cartesian Genetic Programming (BCGP) is another variant that was introduced by Yazdani in 2015 [156]. The main motivation behind this variant was to improve the balance between the exploration and exploitation of the search space, using Biogeography-Based Optimization (BBO) [157]. This balance is often a critical factor in the performance of EA.

Differentiable CGP

Differentiable Cartesian Genetic Programming (dCGP) was introduced in 2017 [158] and extends CGP to allow for the differentiation of the program expression with respect to its functions and its internal parameters. The motivation behind this model was to combine the expressive power of Genetic Programming with the training capabilities of derivative-based methods. An additional specific gene is allocated for each node that is expressed as a weight factor to the result of the function. These weights are automatically adjusted. The Amplitude oriented Cartesian Genetic Programming (ACGP) [159] variant is very similar to the dCGP. While dCGP uses gradient to update the weight genes, ACGP is evolving these values through random mutations.

3.2.5 Cartesian Genetic Programming for Image Processing

The extension to CGP was first achieved by optimizing image filters, especially for image denoizing. The evolved program is the mathematical expres-

sion that must be performed for each pixel of the new image. Therefore, the model uses standard scalar operations and generates a filter that is then applied to the entire image to output the image that will be assessed. Sekanina and Martinek [160–162] proposed implementations on Field-Programmable Gate Array (FPGA) hardware. Harding [163] has drastically improved the computation time performance by optimizing the final image filter computation on GPU.

A major part of the use of CGP for image processing is synthesized in 2011 [164]. In this study, we see an extended application where we try to reproduce results given by various existing filters, such as Canny [29], or even morphological operations such as Dilation and Erosion. The authors also underline that this method is limited by the size of the filter. Considering a 3x3 filter, we have to find an algorithm from 9 input pixels. We go from 25 inputs for a 5x5 filter to 49 inputs for a 7x7, which makes the evolution very complicated.

Subsequently, CGP-IP was introduced in 2012 to extend the utility of CGP to build an integral image processing algorithm using high-level operations. For this, the main function library comprises pre-existing filters, applying directly to an entire image and returning a filtered image. The diverse applications include the segmentation and classification of rocks on Mars [142], classification of cell mitosis [143], and detection of simple objects in robotics [143]. Another application of CGP-IP in robotics allows robots to detect their own hands [165] and improve object detection [166].

3.3 Main Text (as submitted)

Kartezio: A Darwinian Designer of Explainable Algorithms for Biomedical Image Segmentation

Authors: Kévin Cortacero[1], Brienne McKenzie[1], Sabina Müller[1], Roxana Khazen[1], Fanny Lafouresse[1], Gaëlle Corsaut[1], Nathalie Van Acker[2], François-Xavier Frenois[2], Laurence Lamant[2], Nicolas Meyer[3], Béatrice Vergier[4, 5], Dennis G. Wilson[6], Hervé Luga[6], Oskar Staufer[7], Michael L. Dustin[7], Salvatore Valitutti[1, 2*#] and Sylvain Cussat-Blanc[6*#]

Affiliations: [1]INSERM UMR1037, CNRS UMR5071, Centre de Recherche en Cancérologie de Toulouse (CRCT), Université de Toulouse III-Paul Sabatier, 31037 Toulouse, France. [2]Department of Pathology, Institut Universitaire du Cancer-Oncopole de Toulouse, 31059 Toulouse, France. [3]Department of Dermatology, Institut Universitaire du Cancer-Oncopole de Toulouse, 31059 Toulouse, France. [4]Service de Pathologie CHU de Bordeaux. [5]Équipe INSERM UMR1053 -UMR BaRITOn (Eq 3), Université de Bordeaux. [6]University of Toulouse - Institut de Recherche en Informatique de Toulouse (IRIT) - Centre National de la Recherche Scientifique (CNRS) UMR5505, Artificial and Natural Intelligence Toulouse Institute. [7]Kennedy Institute of Rheumatology, Nuffield Department of Orthopedics, Rheumatology and Musculoskeletal Sciences, University of Oxford, Oxford, UK.

[#]Co-senior authors [*]Corresponding authors:

salvatore.valitutti@inserm.fr and sylvain.cussat-blanc@irit.fr

Keywords: Cartesian Genetic Programming; Computer Vision; Artificial Intelligence; Image Analysis; Instance Segmentation; Machine Learning

3.3.1 Abstract

An unresolved issue in contemporary biomedicine is the overwhelming number and diversity of complex images that require annotation, analysis and interpretation. Recent advances in Deep Learning have revolutionized the field of computer vision, creating algorithms that compete with human experts in image segmentation tasks. However, these frameworks require large human-annotated datasets for training and the resulting "black box" models are difficult to interpret. In this study, we introduce Kartezio, a modular Cartesian Genetic Programming-based computational strategy that generates fully transparent and easily interpretable image processing pipelines by iteratively assembling and parameterizing computer vision functions. The pipelines thus generated exhibit comparable precision to state-of-the-art Deep Learning approaches on instance segmentation tasks, while requiring drastically smaller training datasets. This Few-Shot Learning method confers tremendous flexibility, speed, and functionality to this approach. We then deploy Kartezio to solve a series of semantic and instance segmentation problems, and demonstrate its utility across diverse images ranging from multiplexed tissue histopathology images to high resolution microscopy images. While the flexibility, robustness and practical utility of Kartezio make this fully explicable evolutionary designer a potential game-changer in the field of biomedical image processing, Kartezio remains complementary and potentially auxiliary to mainstream Deep Learning approaches.

3.3.2 Introduction

Contemporary imaging techniques in biology and medicine produce a tremendous quantity of images, which must be analyzed to provide meaningful interpretations of biological and clinical processes. The scale, context, and resolution of such image is highly variable, ranging from super-resolution and electron microscopy in biology to whole-tissue imaging in clinical pathology. Generating "actionable intelligence" from such imaging data, using either human or automated analysis of images, require quantitative knowledge extraction [167]. The bottleneck of the knowledge extraction process has historically been the human power required for image evaluation, annotation, and interpretation, all of which can be time-consuming prone to inter-human variability, and subject to error. Moreover, human nature imposes a subjective framework of reasoning that can affect the overall objectivity of the knowledge extraction process and the subsequent validation or invalidation of a given hypothesis. The development of sophisticated automated image analysis pipelines that preserve the nuances of human insight, while circumventing the above-mentioned limitations, is a formidable challenge in contemporary biomedicine. To keep pace with the development of increasingly sophisticated imaging techniques, engineers and researchers have designed a variety of image processing filters and pipelines that assist experimentalists and clinicians in extracting quantitative information from their digital images. Nevertheless, filters and pipelines need a certain level of expertise to be used optimally. AI and more specifically DL approaches have since provided a great step forward in image processing and revolutionized the field of

CV. This is particularly evident in tasks such as SS, which clusters together parts of an image belonging to the same object class, and IS, a complex task that additionally involves the demarcation of overlapping or interacting objects even if they belong to the same class. AI was proven to be competitive with human expertise in medical image analysis within the fields of cardiology, dermatology, ophthalmology, radiology and pathology among others [168, 169]. To date, these approaches have primarily been based on artificial DNN, which are conceived to work at the pixel level to generate filters and pipelines from scratch without building on previous human knowledge and experience. This requires huge computational efforts to provide satisfactory solutions. DNN models are composed of thousands of simple computational units, the neurons, which are connected in a complex network of weighted interactions [13]. Designing the architecture of such a network is a key step before training. Because the subsequent training of an DNN to solve a given task requires the optimization of millions of parameters, DNN require huge computational efforts to provide satisfactory solutions. Despite important advances in XAI [170, 171], DNN have generally been difficult for humans to analyze and interpret, leading to the definition of these models as "black-boxes" [170]. As such, there is substantial interest in developing more explainable "white-box" methods to complement existing DNN approaches in medical and biological image analysis.

Herein, we propose an innovative approach to image analysis that tackles the important challenges described above while remaining fully explainable and interpretable to humans. Our approach, named "Kartezio", is based on the evolutionary design of pipelines assembled from pre-existing human-

engineered filters. This is accomplished by using a specific type of GP [172] algorithm, known as CGP [124, 173, 174], to generate image processing pipelines, known as CGP-IP [142, 143, 145]. As a case study for this powerful approach, we deployed Kartezio to solve different IS and SS challenges, thematically centered around the field of tumor immunotherapy.

Proposed by Miller in 1999 [124], CGP has been successfully used in multiple variants (Mixed Types, Recurrent, Self-Modifying, etc. as reviewed in [153]) for logical circuit generation [174], symbolic regression [137], agent control [5], neural architecture search [175, 176] and simple image processing tasks such as noise reduction [164] and SS [142, 143] but to date the application of this powerful approach to IS in biomedicine has been limited. By merging CGP-based IS with CV measures and classical unsupervised ML, we designed a highly effective AI solution to extract quantitative information from immunofluorescent IF and immunohistochemical Immunohistochemistry (IHC) images. Our CGP-based strategy is: **i)** trainable on small datasets, **ii)** transparent and interpretable by humans, and **iii)** capable of integrating human knowledge. The strategy presented in this manuscript (summarized in **Extended Data Fig. 1 (3.8)**) represents a substantially new conceptual framework for the design of image processing pipelines, and offers a novel approach to tackle the challenge of image analysis in contemporary medical and biological sciences.

3.3.3 Results

Evolving programs for Instance Segmentation with CGP

GP is based on the artificial evolution of a population of syntactic graphs composed of mathematical functions that, when executed, process given inputs to produce an expected output [172]. While GP directly works on the optimization of trees, CGP considers a specific encoding of this graph within an integer-based genotype, also called genome (**Fig. 1a (3.2)**, see **Material and Methods**). In this paper, we developed a modular framework (called Kartezio) to expand CGP into IS tasks and to confer enhanced versatility to its application, thus revitalizing the CGP approach to iteratively evolve image segmentation pipelines, as illustrated in **Extended Data Movie 1**. To this end, we first constituted a library of 42 functions specifically designed for IP (full library is provided in **Extended Data Table 1 (3.3)**). In addition, we introduced the notion of non-evolvable nodes which are functions not subjected to optimization of the syntactic graph (**Fig. 1 (3.2)**). The objective of these two initial steps was to introduce human knowledge about the input images, expected outputs and, if requested, additional known operators into CGP-generated pipelines. With these adaptations, we re-invigorated the CGP paradigm to make it more versatile and introduced **i)** non-evolvable preprocessing nodes to transform input images (e.g. transmission light images, fluorescent images, IHC tissue images, etc.) into the most appropriate format for the expected outcome, and **ii)** appropriate endpoints (such as Watershed Transform [34, 177] or Circle Hough Transform [178]) which can

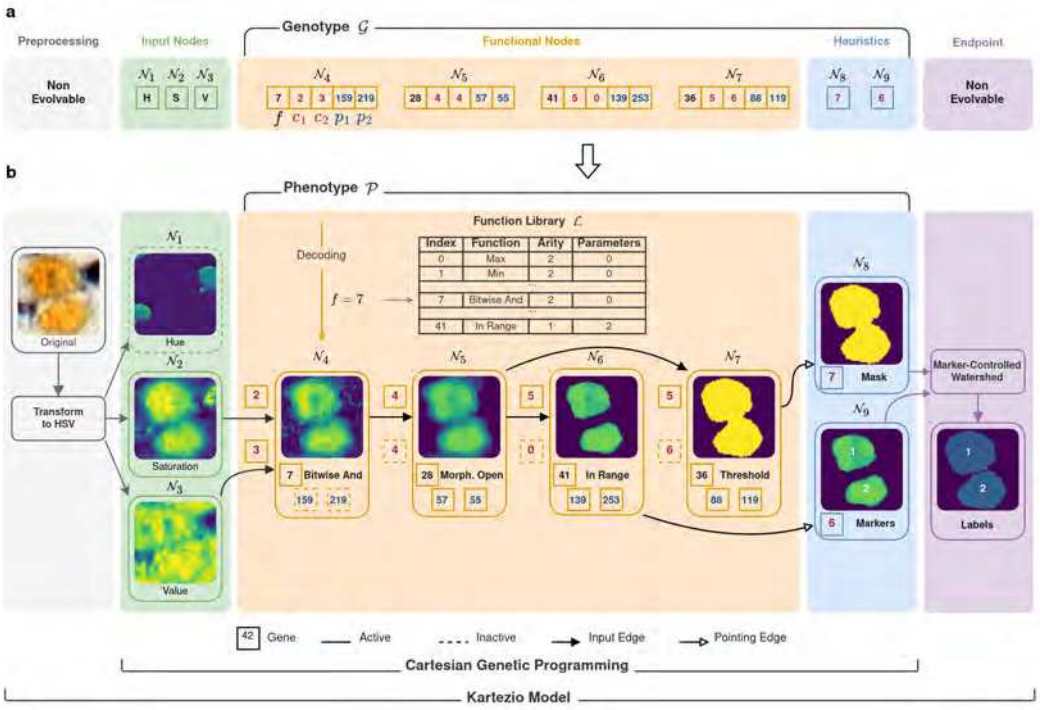
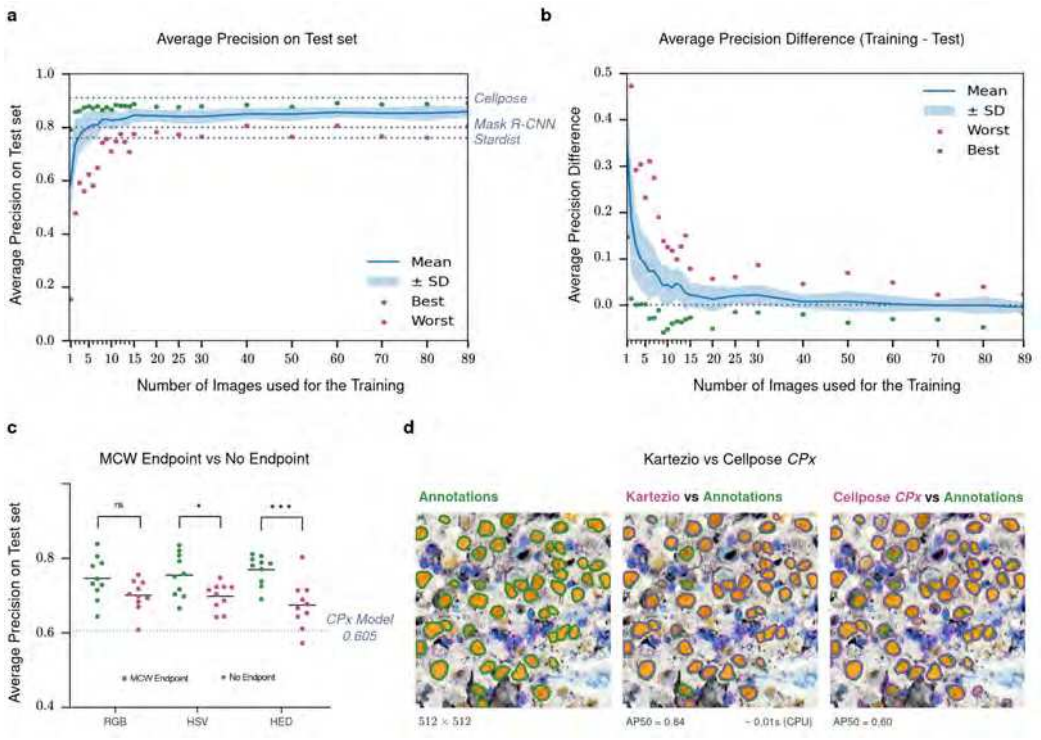


Figure 3.2: **Fig. 1 | Architecture of a Kartezio-generated model: a fully explainable and transparent algorithm for Instance Segmentation**

a, b) An illustrative example of a pruned model trained using Kartezio on a melanoma tissue nodule dataset derived from a previously published melanoma cohort (see **Extended Data Fig. 2b (3.9)**) [2]. From left to right, preprocesing is represented in gray; input nodes are shown in green; functional nodes are shown in orange; heuristics are shown in blue; and endpoint is shown in purple. **a)** The genotype $\mathcal{G}$ is a sequence of integers known as genes (each one represented as a box containing a single integer) that are organized into nodes, $\mathcal{N}_i$. A node is composed of a single function{) values shown in black) drawn from the function library $\mathcal{L}$, at least one connection (values shown in red, in this example, $c_1$and c_2), and optional parameters (values shown in blue, in this example, p_1 and p_2. Nodes are categorized as either functional nodes (in this example, $\mathcal{N}_{4\to7}$) or outputs (in this example, $\mathcal{N}_{8\to9}$). In this example, three entry channels ($\mathcal{N}_{1\to3}$), four functional nodes ($\mathcal{N}_{4\to7}$) and two outputs ($\mathcal{N}_{8\to9}$) are shown. **b)** The phenotype $\mathcal{P}$ decoded from the genotype described in **a)** takes the form of a DAG. To obtain the active graph (solid lines), only the genes that are directly or indirectly connected to outputs are considered. The last operation of the pipeline, called the Endpoint, adds a non-evolvable user-chosen function (in this case, the Marker-Controlled Watershed for IS) that creates an intentional bottleneck to drive the heuristics optimization. While the Endpoint is part of the evaluation of the genotype, it is not subjected to evolution and is chosen by the user depending on the task to solve.



strongly reduce the search space to perform IS, similarly to what is done in many current DNN-based methods [39, 69, 85, 116, 179]. Together the decoded genotype and the non-evolvable nodes will produce executable IP pipelines (**Fig. 1b (3.2)**).

In the next section, we compare our CGP approach to state-of-the-art DL techniques applied to cell IS within microscopy and IHC images.

Evaluation of Kartezio performance versus state-of-the art DL models

Figure 3.3: **Fig. 2 | Kartezio-generated models achieve comparable accuracy as state-of-the-art Deep Learning approaches with fewer training requirements**
(a,b) Kartezio was trained on 1 to 89 images, derived from Cell Image Library [39]. Its performance on a test set of 11 images was compared to Cellpose, Mask-RCNN and StarDist (each trained on 89 images; dotted lines). **a)** Average Precision (AP50) scores are shown for the indicated number of training images (mean +/- SD of $n = 35$ models per condition; each model represents an independent experiment). **b)** Average differences in score between training and test datasets (mean +/- SD of $n = 35$ models per condition; each model represents an independent experiment). Dotted line indicates optimal difference of 0. Dots indicate best (green) and worst (red) scores obtained by Kartezio for the indicated number of training images. **(c, d)** For each set of conditions, ten Kartezio models were trained on twelve images from the melanoma cohort [2]. Each model is indicated by one data point; $n = 10$ independent experiments per condition. In parallel with the best generalist Cellpose model (*CPx*), these models were evaluated on a test dataset of ten images. **c)** AP50 scores for models trained using no defined Endpoint versus models trained using Marker Controlled Watershed (MCW) endpoint combined with different preprocessing methods (RGB, an HSV transformation, and a HED deconvolution) for ten Kartezio generated models and compared to the AP50 of CPx. Each datapoint represents a single Kartezio generated model ($n = 10$ independent experiments per condition). Bars indicate mean of ten experiments. Dotted line indicates score of *CPx* (0.605). MCW versus the No Endpoint control was compared using unpaired 1-way ANOVA with multiple comparisons. **d)** Representative histology image from the test dataset with Instance Segmentation (IS) performed by Kartezio or Cellpose, compared to manual annotations. Left: manual annotations (green outline); middle: IS performed by the Kartezio-generated model represented in **Fig. 1 (3.2)** (magenta outline) and compared to ground truth (green outline); right: IS performed by CPx (magenta outline) and compared to ground truth. ($*p < 0.05$, $***p < 0.001$, ns = non significant). Source data for all experiments are provided with this manuscript and summary statistics are listed in **Extended Data Table 2. (3.4)**

To evaluate the performance of this approach against state-of-the-art DL models, we compared Kartezio to Cellpose [39], a recently released DL approach for IS in biology with both specialist and generalist models, as well as to two previously released approaches, Mask R-CNN [85] and StarDist [69]. As described below, we first evaluated Kartezio on the Cellpose dataset and compared it to the reported performance of the Cellpose specialist

model (**Fig. 2a,b (3.3)**) [39]; we then assessed both Kartezio and the Cellpose generalist model side-by-side on our own recently published dataset (**Fig. 2c,d (3.3)**) [2].

In their initial study, Stringer et al introduced the Cellpose model to perform IS on diverse types of cells or nuclei; this generalist model was trained on over $70,000$ segmented objects [39]. The authors also proposed a distinct specialist model, trained on in vitro cultured neurons (individual channels from a representative image are displayed in **Extended Data Fig. 2a (3.9)**); this dataset was composed of 89 training images and 11 test images [39]]. Since Kartezio is a modular framework for generating specialist models, we chose to initially compare our approach to the Cellpose specialist model, and accordingly utilized the 89-image training dataset and 11-image test dataset previously used in the generation of the Cellpose specialist model. We utilized randomly selected subsets of these 89 images (ranging from 1 to 89 images) to train Kartezio from scratch and subsequently evaluated its performance on the previously unseen Cellpose test dataset. **Fig. 2a (3.3) and Extended Data Table 2 (3.4)** show the scores obtained on the 11-image test dataset after first training Kartezio on datasets of different sizes. Dotted lines indicate the results obtained by the other three methods as reported in Stringer et al [39], when trained on all 89 images. It is interesting to note that Kartezio frequently outperformed [85] and [69] even when trained on much smaller datasets (e.g. on average, Kartezio matched Mask R-CNN when trained on as few as 6 images, and matched StarDist when trained on only 3 images). This illustrates the exceptional capacity of Kartezio to

extrapolate from a very small set of training images (a property known as "few-shot learning") and yet achieve comparable or superior accuracy compared to state-of-the-art DL approaches. Our results also show that the best score obtained by Kartezio was 0.89, a value comparable to the reported best performance of Cellpose (0.91) (**Extended Data Table 2 (3.4)**). Of note, increasing the size of the training set in 10-image increments from 30 to 80 images did not perceptibly change Kartezio's performance.

The capacity of Kartezio to extrapolate from only a few training events was also evident when we compared the scores obtained on training subsets with those obtained on the 11-image test dataset. In this analysis, a score of 0 is optimal as it shows no decrease in accuracy when transitioning from the training to the test set. As shown in **Fig. 2b (3.3)**, the difference of mean average precision (AP) between the training and test sets was close to 0 when only 9-10 images were used for training. These data indicate that Kartezio generates pipelines of comparable precision to state-of-the-art DL approaches but with a drastic reduction in the size of the training dataset. Interestingly, Kartezio generalized well on this dataset as the training versus testing scores were comparable (**Extended Data Table 2 (3.4)**). In a second exercise, we compared the performance of Kartezio to the Cellpose generalist model in the context of identifying individual melanoma nuclei within complex real world histopathology images (**Fig. 2c,d (3.3)**). The original dataset, consisting of tissue samples from melanoma patients and the associated clinical data, has been previously described [2]. In the present study, we employed 12 histopathology images for Kartezio training, and a further 10 images for Kartezio and Cellpose side-by-side comparison (cropped as shown in **Fig.**

2d (3.3)). As depicted in **Fig. 2c (3.3)**, we also used this opportunity to evaluate different non-evolvable nodes in the Kartezio pipeline: RGB versus HSV versus HED in the pre-processing node, and Marker-Controlled Watershed (MCW) versus No Endpoint in the endpoint node. Individual RGB, HSV, and HED channels are shown in **Extended Data Fig. 2b (3.9)**. All six CGP-based approaches significantly outperformed the Cellpose generalist model ($AP50 = 0.605$) on this dataset ($p < 0.01 - 0.0001$, one-sample t-test). Furthermore, the addition of the MCW endpoint significantly improved Kartezio's performance compared to the No Endpoint control, particularly when used in combination with either HSV ($p < 0.05$) or HED ($p < 0.001$) preprocessing (1-way ANOVA with multiple comparisons). To visualize the differences between Kartezio and Cellpose, annotations of the ground truth (manual annotation, left) and the segmentation predicted by each model (middle, right) for a representative image from the test set are provided in **Fig. 2d (3.3)**. In the middle and right panels, nuclei that have been correctly identified by the models are circumscribed with similar outlines in both green (manual annotation) and magenta (model annotations); nuclei that failed to be identified by the model (false negative) appear in green only, while areas that have been incorrectly identified as nuclei (false positive) are circumscribed in magenta only. In this image, Kartezio achieved an AP50 of 0.84 while Cellpose achieved an AP50 of 0.60; however, a modest number of errors such as those mentioned above are visually detectable. Of note, the model of Kartezio used to illustrate this example is represented in **Fig. 1 (3.2)** and its corresponding generated Python class is shown in **Extended Data Fig. 3 3.10**. Thus, Kartezio is capable of generating a highly

effective IS pipeline, containing just four common IP functions preceding the MCW, which is fully explainable to humans and implemented using only a few basic lines of code (**Extended Data Fig. 3 (3.10)**). The execution time of the whole pipeline for a 512×512 pixel image such as that shown in **Fig. 2d (3.3)** is a mere $0.01s$, using one Central Processing Unit (CPU) of one laptop. In the following sections of this article, we highlight the versatility of Kartezio. We first demonstrate the performance of Kartezio in the SS of patient tissue sample images based on a limited number of annotations (*Use Case 1*), underlining the complete explicability and accessibility of the generated models. Both explicability and accessibility are key features in clinical settings. We then illustrate three additional Use Cases in which Kartezio successfully performs IS of biological entities (from nanoscale extracellular particles to microscale immune cells), again based on very few annotations. All the images that we included in our study are conceptually focused on the theme of CTL and tumor cell interactions in the context of tumor immunology.

Extended Data Table 3 (3.5) outlines the key features of dataset preparation and analysis for the four different Use Cases in the portfolio. **Extended Data Table 4 (3.6)** summarizes the model parameters for each Use Case and **Extended Data Table 5 (3.7)** includes the fitness scores (min, max, and mean) for testing and training datasets. All four Use Cases utilize the same default library of 42 functions (**Extended Data Table 1 (3.3)**) and the same hyperparameters (e.g. number of functional nodes, mutation rates, etc.). Of note, the library and hyperparameters can be still adapted by the user if requested. We have also included examples

of possible post-segmentation analyses for each Use Case (performed after image segmentation by Kartezio and listed in **Extended Data Table 3 (3.5)**); these are for illustrative purposes and can be modified to answer different experimental questions without changing the function of Kartezio.

Use Case 1: Explainable and accessible models for Semantic Segmentation of tumor nodules in patient tissue samples

We propose an end-to-end translational application of Kartezio for tumor nodule SS in patient tissue samples. The goal is to provide clinicians with an easy-to-apply and fully interpretable tool that is automatically generated starting from a small annotated dataset. For an initial application, we profited from an off-the-shelf tissue sample cohort of metastatic melanoma patients we have previously described [2]. In this application, as shown in **Fig. 3a 3.4**, Kartezio was employed to create a SS mask of tumor nodules within IHC-stained tissue samples. Melanoma cells and CTLs were identified by Sox10 (a nuclear marker of melanocytic lineage, orange) and CD8 (a CTL marker, purple) expression respectively and cell lysosomal content was identified by CD107a staining (black). All cells were counterstained with hematoxylin (blue). In the present study, images were preprocessed by transforming RGB channels into HSV color space (**Extended Data Fig. 2c 3.9**). We limited the analysis to a dataset containing 24 images. **Extended Data Fig. 4 3.11** depicts the 12 images used for training (**Extended Data Fig. 4a 3.11**) and the 12 used for testing (**Extended Data Fig. 4b 3.11**). Both image series have been annotated by an expert pathologist (green lines); illustrative examples are shown in **Fig. 3a 3.4**. Based on 100 runs, Kartezio

obtained a mean IoU score of 0.87 ± 0.018 on the training dataset, with a maximum score of 0.91, as shown in **Extended Data Table 5 3.7**. Kartezio obtained a mean score of 0.85 ± 0.047 on the test dataset, with a maximum score of 0.89. For each image in the training and test dataset, we merged the predictions of 100 trained models to generate composite heatmaps providing integrated representations of the obtained masks, as shown in **Fig. 3a 3.4** and Extended Data Fig. 4. These composite masks can also be evaluated against the ground truth to provide IoU scores. As shown in **Extended Data Fig. 4c,d 3.11**, we evaluated the fitness of the ensemble of 100 pipelines at multiple different threshold values (between $t = 0.0$ and $t = 1.0$ in increments of 0.01; illustrative images shown at $t = 0, 0.35, 0.5, 0.6$ and 0.75). This analysis demonstrated that fitness is highly dependent upon threshold value; using the training dataset, we calculated an optimal threshold of $t = 0.35$. Across the 12 tumor nodules of the test dataset, the pipeline ensemble yielded a mean fitness score of 0.90. At this optimal threshold, the ensemble outperformed individual pipelines on average and even outperformed the best individual pipeline, which had a score of 0.89 on the test set, illustrating the robustness of the ensemble approach for semantic segmentation. It is worth noting that fitness scores ranged from 0.81 to 0.96 across the 12 images for the test dataset, indicating that the pipeline ensemble performed consistently (with an SD between images of 0.054) and with high degree of accuracy in every tumor nodule tested, an important feature for clinical applications.

Notably, each of the 100 Kartezio-generated algorithms can be interpreted and explained. As an example, we selected the pipeline that obtained the

best IoU score and dissected its behavior (**Fig. 3c**). The first step (node 3) of this pipeline consisted of subtracting the Hue channel (node 0) from the Saturation channel (node 1), allowing for the exclusion of pixels corresponding to CD8+ T cells and to other cells belonging to the tumor microenvironment. Then Kartezio used a Gaussian Blur filter (node 4) in order to smooth the previously obtained area. Next, Kartezio chose a Laplacian filter (node 10) to detect the variations in the image obtained in node 4, thus delimiting the tumor borders. Interestingly, the sequence Gaussian Blur/Laplacian filters is frequently used by image analysts for image segmentation tasks. To highlight the segmented areas, Kartezio used Morphological Black Hat (node 17) which reveals small objects that are darker than their surroundings. Altogether, these initial calculations provided an accurate estimation of tumor area.

In parallel, node 12 created a binary mask of the Value channel (brightness) allowing for the establishment of a global map of the area of interest. The next step selected by Kartezio consisted of merging the two maps with the min operator (node 19). This allowed for the elimination of all pixels detected by the first calculation (results of node 17) that fell outside the area of interest. Kartezio next selected a Fill Hole filter in order to transform the selected area limits into a solid binary mask. A final step consisted of removing the noise from the edges of the mask using a Median Blur filter (node 31). In its entirety, the described pipeline allowed for the sharp delimitation of tumor nodules with a high degree of accuracy. Crucially, a step-by-step dissection of Kartezio's pipeline, similar to the one we outlined above as an example, can be applied to each pipeline.

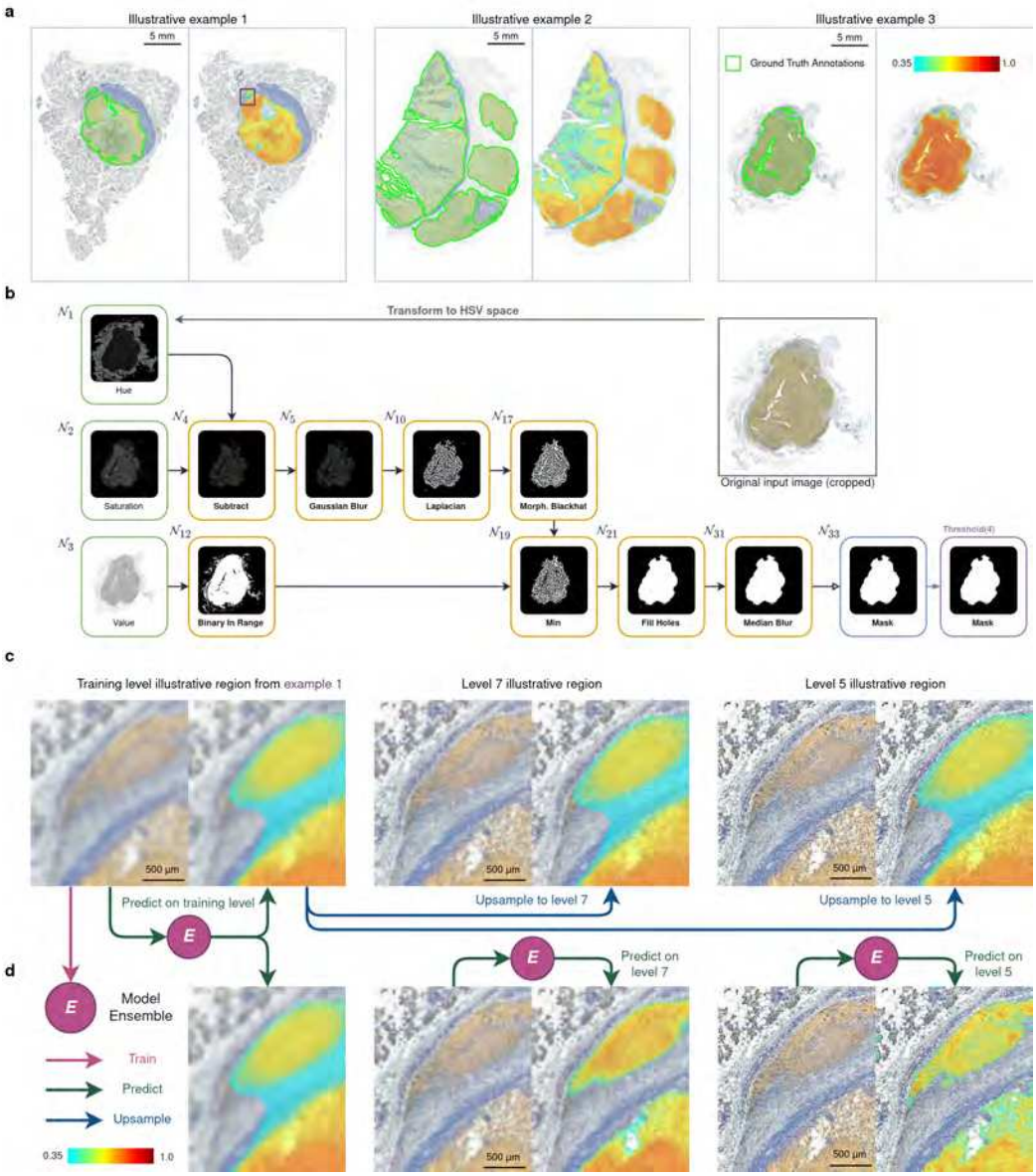

a
Illustrative example 1
5 mm
Illustrative example 2
5 mm
Illustrative example 3
5 mm
Ground Truth Annotations
0.35
1.0
b
Transform to HSV space
Hue
Saturation
Subtract
Gaussian Blur
Laplacian
Morph. Blackhat
Value
Binary In Range
Min
Fill Holes
Median Blur
Mask
Mask
Original input image (cropped)
c
Training level illustrative region from example 1
Level 7 illustrative region
Level 5 illustrative region
500 µm
Predict on training level
Upsample to level 7
Upsample to level 5
d
E
Model Ensemble
Train
Predict
Upsample
0.35
1.0
Predict on level 7
Predict on level 5

Figure 3.4: **Fig. 3 | Segmentation of tumor nodules within IHC-stained tissue specimens**
Use Case 1: **a)** Tissue slices stained for melanoma marker Sox10 (orange), CTL marker CD8 (purple), lysosomal marker CD107a (black) and hematoxylin (blue) were used for training and testing of Kartezio-generated semantic segmentation pipelines (three illustrative tumors shown; full dataset in **Extended Data Fig. 4 (3.11)**). Left panels depict tumor contours delineated by an expert pathologist (green lines). Right panels depict predictions of an ensemble of $n = 100$ Kartezio-generated pipelines, with outputs averaged to generate composite heatmaps of predicted tumor contours using an optimized threshold value of $t = 0.35$ (i.e. minimum probability depicted is 0.35; maximum is 1.0). **b)** The pipeline with the best IoU score (out of $n = 100$ independent experiments) is shown. The mask generated by this pipeline was combined with masks predicted by 99 other pipelines to create the $n = 100$ model ensemble (indicated by **E**), which formed the basis of heatmaps shown in **(a)**. **c-d)** Two strategies were tested for upscaling existing pipelines to high resolution images using an illustrative region (purple box inset) of the tumor nodule in **(a)**. **c)** The heatmap generated from the model ensemble trained on low resolution ("training level") images was upsampled and overlaid onto the high resolution images (Level 7 and Level 5). **d)** The model ensemble trained on training level images ($n = 100$) was deployed on high resolution Level 7 and Level 5 images to generate 100 *de novo* masks on each level, which were merged to generate a new heatmap corresponding to each level ($t = 0.35$; minimum probability depicted is 0.35; maximum is 1.0). Source data for all experiments are provided with this manuscript and summary statistics are provided in **Extended Data Table 5. (3.7)**

We next wanted to determine whether the model ensemble that generated the initial 100 masks on the low-resolution IHC images was capable of demarcating tumors in a higher-resolution context, using the optimal threshold ($t = 0.35$) calculated in **Extended Data Fig. 4c 3.11**. As shown in **Fig. 3c,d 3.4**, a technically challenging segmentation area of the tumor nodule pictured in Example 1, **Fig. 3a 3.4** (see purple box inset) was used to test this concept using two different strategies. First, as shown in **Fig. 3c 3.4**, the heatmap previously generated from 100 masks on low-resolution training level images was upsampled and applied to high resolution images (level 7 and 5). Second, as shown in **Fig. 3d 3.4**, the ensemble of 100 pipelines de-

rived from the low-resolution training set was deployed on higher resolution images, generating 100 new masks at each level, which were subsequently merged into a single heatmap as shown. Remarkably, in this illustrative example, the heatmap generated by this second strategy demarcated the tumor boundaries in close accordance with the high resolution IHC images, despite the region being technically difficult to segment. Furthermore, neither of these two strategies involved re-training Kartezio on high resolution images, but rather repurposing the masks (first strategy) or the pipeline ensemble (second strategy) from the low-resolution dataset. The ability to train Kartezio on low-resolution images to generate pipelines that perform well on high resolution images provides a substantial computational advantage, since the extremely cumbersome amount of data associated with high resolution IHC images makes them challenging to use in a training context.

This easily interpretable tool to rapidly segment tumor nodules has the potential to allow automatic assessment of various parameters that are critical for cancer patient management, such as tumor nodule size, morphological features, tumor localization, its borders with surrounding tissue, and distance from the resection margins, as described in more detail in the discussion. In the next sections, we illustrate three additional Use Cases in which Kartezio successfully performs IS in biological images of different scale and resolution.

Use Case 2: Multi-molecular scale: detecting and measuring CTL-derived extracellular particles

Recent studies revealed that the fight between CTLs and tumor target cells

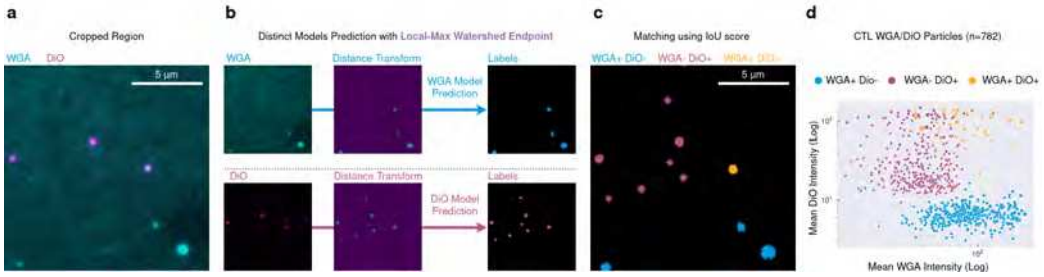


Figure 3.5: **Fig. 4 | Analysis of extracellular particles using Kartezio-mediated instance segmentation**

Use Case 2: CTL-derived extracellular particles stained with WGA and DiO were imaged using TIRFM (see **Extended Data Fig. 2d (3.9)**). Total Internal Reflection Fluorescence Microscopy (TIRFM) images of particles released from human polyclonal CD8+ CTLs (derived from two donors) and stained with Alexa 647-conjugated wheat germ agglutinin (WGA; a red fluorescent probe that stains sialic acid and N-acetyl-glucosamine moieties of glycoproteins), and DiO (a green fluorescent lipophilic dye that stains hydrophobic lipid membranes). **a)** A typical cropped image depicting extracellular particles that were positive for either WGA (cyan), DiO (magenta) or both. **b)** The two images on the left depict the individual channels (WGA and DiO). For each channel, Kartezio generated a model for IS (arrows) leading to labels via the Local-Max Watershed Endpoint. Each label corresponded to one instance (particle) in a given channel as shown. **c)** To begin the post-segmentation analysis, each label was classified as WGA+ (cyan), DiO+ (magenta) or WGA+DiO+ (gold), with matching evaluated on the basis of label overlap using the IoU metric. **d)** Feature extraction from each instance involved quantification of particle size and mean fluorescence intensity (MFI). Individual dots represent WGA+ (cyan), DiO+ (magenta) and WGA+DiO+ double positive particles (gold) plotted on the basis of MFI. Dot size is proportional to the size of the particle mask corresponding to each instance. Representative data shown are derived from one experiment (i.e. one model per channel) from a total of $n = 35$ generated models per channel. Source data from $n = 35$ experiments are provided with this manuscript and summary statistics are provided in **Extended Data Table 5 (3.7)**.

is not limited to lytic synapses (specialized cell-cell contact areas where cytotoxic lytic granules are secreted from killer CTLs [180, 181]) but also occurs via the release of extracellular aggregates of lytic components and additional bioactive molecules called Supra Molecular Attack Particles (SMAPs) [182, 183]. SMAPs are currently under investigation as potential anti-cancer therapeutics and may represent a potent new tool in the arsenal of oncologists. It is therefore crucial to develop new methods to characterize these particles in order to better define their biological function and therapeutic potential. As individual CTL-derived SMAPs are small ($\simeq 120nm$ in diameter, as detected by D-STORM imaging) [183], high resolution imaging techniques are required for their visualization.

In a first application of Kartezio, we acquired high resolution Total Internal Reflection Fluorescence Microscopy (TIRFM) images of particles released from human polyclonal CD8+ CTLs and stained with Alexa 647-conjugated wheat germ agglutinin (WGA; a red fluorescent probe that stains sialic acid and N-acetyl-glucosamine moieties of glycoproteins), and DiO (a green fluorescent lipophilic dye that stains hydrophobic lipid membranes); individual channels from a representative image are displayed in Extended Data Fig. 2d. The population of particles included WGA+ (cyan), DiO+ (magenta) and double positive particles (lavender) (**Fig. 4a 3.5**). The aim of this test paradigm was to determine if Kartezio could distinguish putative SMAP particles (WGA+ DiO-) from a mixed population of extracellular particles and vesicles, and then to extract quantitative information about the different particle populations. Although conceptually simple, this Use Case enabled us to demonstrate the performance of Kartezio in the context of single-particle

high resolution imaging. Using Kartezio, we created an automated procedure to perform IS, which permitted the subsequent identification of single- and double-positive particles and characterization of their features. For each channel, we first trained Kartezio on a dataset consisting of one manually-labelled image, split into quarters (the number of ROIs is indicated in **Extended Data Table 3 3.5**). Based on 35 training runs, Kartezio achieved an average AP50 score of 0.81 ± 0.037 and 0.79 ± 0.030 for the WGA and DiO channels respectively, with a maximum score of 0.87 and 0.84 for WGA and DiO respectively. To assess generalizability, all IS pipelines were then tested on a dataset of one manually-labelled previously unseen image, split into quarters. Under these test conditions, Kartezio achieved an average AP50 score of 0.73 ± 0.083 and 0.76 ± 0.090 for the WGA and DiO channels respectively, with a maximum score of 0.84 for WGA and 0.85 for DiO. These results are summarized in **Extended Data Table 5 3.7**. For each channel, the best overall pipeline (based on test and training fitness scores) was then applied to an image cohort containing an additional 9 images of CTL-derived extracellular particles. For the WGA channel, the best overall pipeline had a training fitness score of 0.87 and a test fitness score of 0.84. For the DiO channel, the best overall pipeline had a training fitness score of 0.81 and a test fitness score of 0.85. **Fig. 4b 3.5** shows a representative example of cyan and magenta particle segmentation obtained with the best pipeline for each channel out of the 35 runs performed. In order to identify double-stained particles post-segmentation, the WGA and DiO instances previously detected using Kartezio were tested for matching using a pairing mechanism based on the Intersection over Union (IoU) metric

(detailed in Materials and Methods). Instances detected in the WGA and DiO channels with the IoU score above a threshold of 0.05 were identified as double-positive instances. The entire process enabled the identification of three sets of particles: WGA+DiO- (cyan), WGA-DiO+ (magenta), and WGA+DiO+ (orange) (Fig. 4c 3.5). From these datasets we could readily calculate the Mean Fluorescence Intensity (MFI) and area of each instance as shown in **Fig. 4d 3.5**. The panel displays a scatterplot of WGA+DiO- (cyan), WGA-DiO+(magenta), and WGA+DiO+ particles (orange) released from CTLs. This analysis proved capable of distinguishing extracellular vesicles (WGA-DiO+ and WGA+DiO+ particles) from putative SMAPs, which were previously shown to be WGA+ but negative for lipophilic plasma membrane dyes [183].

Taken together, the above results show that Kartezio can successfully perform rapid and accurate IS of extracellular particles at a multi-molecular scale, providing access to further analysis of a statistically relevant number of events. It is important to note that the pipelines generated by a Kartezio can be exported to standard Python scripts, which can be readily re-used on additional images acquired in comparable experimental conditions.

Use Case 3: Sub-cellular scale: assessing the lytic arsenal of individual CTLs

To test Kartezio's performance on a larger scale, we assessed the capacity of Kartezio to automatically measure expression of various molecules within cells, in order to classify individual cells based on their molecular content. Here we focused on the CTL lytic arsenal by measuring the expression of various lytic granule components (e.g. perforin, granzyme B, and CD107a) in polyclonal human CTLs. Cells were stained with antibodies directed against

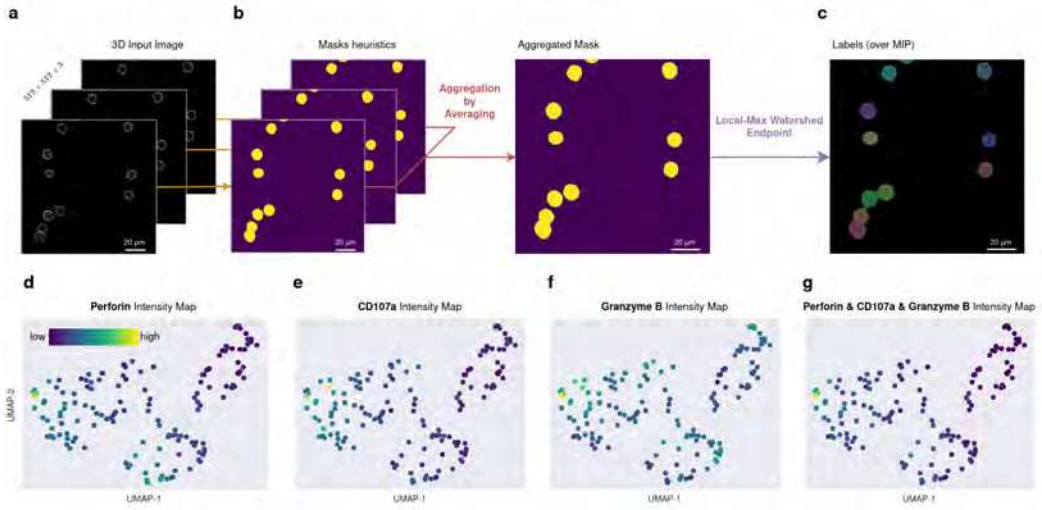


Figure 3.6: **Fig. 5 | 3D analysis of lytic molecule content in individual CTLs within a polyclonal population**
Use Case 3: Human polyclonal CD8+ T cells were stained with antibodies directed against CD45, perforin, granzyme B and CD107a (see **Extended Data Fig. 2e (3.9)**). **a)** z-stacks of images in the CD45 channel were used as a 3D input to delimitate individual cells. **b)** Kartezio generated an instance segmentation pipeline that was sequentially applied to each layer of the z-stack to generate the corresponding mask. The masks were aggregated using an averaging operator. **c)** The application of the Local Max Watershed Endpoint allowed the generation of 2D labels. **d-f)** For the post-segmentation analysis, the MFI of perforin, granzyme B, and CD107a was calculated for each instance to derive a single feature vector for each cell. This vector was a composite of different quantitative readouts and integrated both individual and combined counts of positive pixels, sums of pixel fluorescence intensities, and average intensities measured in the different channels. Each cell was visualized on a 2D UMAP representation [184] based upon this composite feature vector. **g)** The same UMAP representation of merged lytic components. Representative data shown are derived from one experiment, from a total of $n = 35$ independent experiments. Source data for $n = 35$ experiments are provided with this manuscript and summary statistics are provided in **Extended Data Table 5 (3.7)**.

CD45 (grayscale), perforin (magenta), granzyme B (yellow) and CD107a (cyan) (see **Extended Data Fig. 2e (3.9)**). Staining for CD45, a molecule expressed on the surface of all hematopoietic lineage cells including CTLs, allowed Kartezio to delineate individual cells, while the other markers were subsequently used to score the CTL lytic arsenal during the post-segmentation analysis.

For the initial analysis, we applied Kartezio to series of z-stacks containing CD45 staining only (**Fig. 5a (3.6)**). To achieve this aim, we upgraded Kartezio to segment cells in 3D stacks to provide accurate 2D masks of each identified cell. Kartezio-generated pipelines were sequentially applied to each layer of the z-stack to produce initial Watershed masks of each z-stack. All the masks were merged with an averaging operator which allowed us to obtain the maximal area of the cells in the z-dimension (**Fig. 5b (3.6)**). Local Max Watershed was applied to produce the IS masks of the cells (individual segments, **Fig. 5c (3.6)**).

For this analysis, the training fitness was more stringent than in the previous analysis (AP with a threshold of 0.7 instead of 0.5) to avoid overfitting. The training dataset was limited to 5 images containing 45 cells, highlighting the aforementioned capacity of Kartezio to be trained for cell segmentation on a very small set of images. On 35 independent runs, Kartezio obtained an average score of 0.98 ± 0.007 on the training dataset ($n = 5$ images with 45 cells) and 0.83 ± 0.028 on the test dataset ($n = 4$ images containing 34 cells), with a maximum score of 1.00 during training and 0.89 during testing (results shown in Extended Data Table 5). The best overall pipeline had a score of 1.00 during training and 0.85 during testing.

After the individual cells were segmented using Kartezio's best pipeline, we measured staining of the three lytic granule components (perforin, granzyme B, and CD107a) in each z-section to derive a single feature vector for each cell. This vector was a composite of different quantitative readouts and integrated both individual and combined counts of positive pixels, sums of pixel fluorescence intensities, and average intensities measured in the different channels (see Material and Methods). Each cell was then positioned on the Uniform Manifold Approximation and Projection (UMAP) [184] based upon this composite feature vector. The gradient of expression of each lytic granule component within cells of the whole population is shown in **Fig. 5d-f (3.6)**. **Fig. 5g (3.6)** shows the gradient of expression in the whole population for the three markers together.

Together these results underline the capacity of Kartezio to serve as a versatile tool for the analysis of molecular content in cells. Importantly, this function can be used more broadly to automatically assess staining intensities in a wide range of microscopy images, one of the most common tasks in image analysis in cell biology.

Use Case 4: Cell-cell interface scale: discerning CTL/target cell lytic synapses within a mixed population of cells

In the next step of Kartezio's validation, we focused our analysis on lytic synapses, which are highly ordered structures that form when a CTL productively engages a target cell that expresses its cognate antigen [180, 181]. A CTL/target cell interaction that forms around a lytic synapse is known as a cell-cell "conjugate". Identifying a CTL/target cell conjugate is a complex task, which involves correctly identifying and demarcating different cell

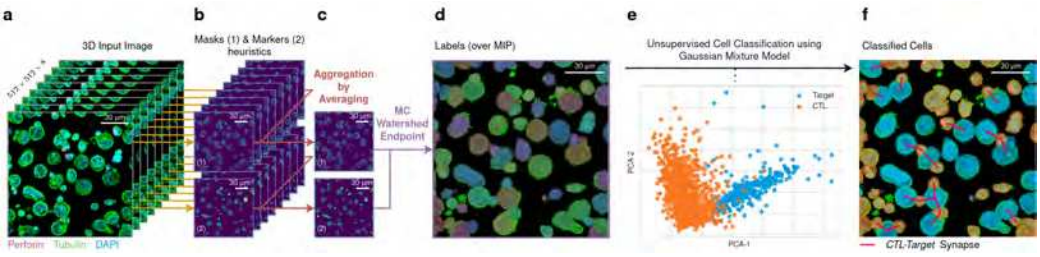


Figure 3.7: Fig. 6 | A hybrid Kartezio/AI approach to detect CTL/target cell conjugates

Use Case 4: **a)** A mixed population of interacting cells containing both human clonal CTLs and target cells was stained with DAPI (cyan) and antibodies directed against α-tubulin (green) and perforin (magenta), as shown in **Extended Data Fig. 2f (3.9)**. Each series of z-sections was used as a 3D input. **b)** The same Kartezio-generated instance segmentation pipeline was applied to each z-section to generate pairs of masks and markers. **c)** All masks and markers were merged to obtain one 2D mask heatmap and one 2D marker heatmap. **d)** The Watershed Endpoint was applied to generate 2D labels. **e)** During post segmentation analysis, unsupervised Machine Learning (Principal Components Analysis followed by Gaussian Mixture Model) was used to classify the two-cell population. **f)** Visualization of cell classification. Colored masks indicate the two classes of cells, CTLs (orange) and target cells (blue). Red lines indicate putative synapses between CTLs and target cells. Representative data shown are derived from one experiment, from a total of $n = 35$ independent experiments. Source data are provided with this manuscript and summary statistics are provided in **Extended Data Table 5 (3.7)**

.

types and assessing whether two adjacent cells are interacting. Herein, our objective was to design an automatic procedure which would take as input 3D stacks of images of CTLs interacting with target cells to identify the CTL/target cell conjugates.

To achieve this, cells were stained with DAPI (cyan, to detect CTL and target cell nuclei), with antibodies directed against a-tubulin (green, to visualize the cytoskeleton in both cells), and perforin (magenta, a lytic molecule expressed only by CTLs) to allow Kartezio to distinguish the CTLs from the targets (**Fig. 6a (3.7)**); individual channels from a representative image are displayed in **Extended Data Figure 2f (3.9)**.

As previously illustrated in **Fig. 5b (3.6)**, Kartezio-generated pipelines were sequentially applied to each layer of the z-stack. This process generated paired images of masks and markers (**Fig. 6b (3.7)**). All the obtained pairs were then merged using an averaging function (see Materials and Methods) to produce one mask heatmap and one marker heatmap (**Fig. 6c (3.7)**). Watershed was then applied to the heatmaps to produce 2D masks containing distinct segments of cells (**Fig. 6d (3.7)**). At this step, Kartezio achieved an average AP50 score of 0.75 ± 0.022 over 35 runs during training with a maximum score of 0.81, and an average score of 0.67 ± 0.034 during testing on a validation set containing 150 previously unseen cells. These results are summarized in **Extended Data Table 5 (3.7)**. The best overall pipeline had a training score of 0.79 and a testing score of 0.72, and was utilized for the subsequent post-segmentation analysis. At this stage of the image analysis, the segmented instances generated by Kartezio (each corresponding to exactly one cell, either CTL or target cell) were further analyzed using conventional

image analysis approaches. Knowing the average dimensions of CTLs and target cells, we excluded all entities with a surface $\leq 350px(\simeq 24\mu m^2)$ and $\geq 6000px(\simeq 417\mu m^2)$, which might correspond to cell fragments or big cellular clusters. Additionally, we also applied a morphological closing to smooth cell contours. The next step, depicted in **Fig. 6e (3.7)**, was to classify each instance as either CTL or target cell. To this end, we calculated the area and the intensity of DAPI (cyan), a-tubulin (green) and perforin (magenta) fluorescence for each instance. We considered these four features sufficient to separate the two cell populations since CTLs are statistically smaller than target cells and express perforin. Unsupervised machine learning using Principal Component Analysis (PCA) [185, 186] followed by Gaussian Mixture Model (GMM) (see Material and Methods) was then used to separate the two population of cells. This method allowed us to reach an accuracy of 92.2% on a validation set containing 153 cells, as evaluated against the ground truth established by independent analyzers.

Once the CTL and target cells were identified, we selected CTL/target cell conjugates based on their contiguity using the following two complementary procedures: i) we calculated the distance between the centroid of the CTL and the target cell masks and kept only the cells with a distance less than 70 pixels ($\simeq 18.5\mu m$). This procedure allowed us to keep in the analysis CTL and target cells with high probability of contact; and ii) in the selected conjugates, each target cell mask was slightly dilated to potentially intersect with other masks. This procedure was required since IS produces distinct (i.e. non-overlapping) masks. If an intersection with a CTL mask was detected, the target cell was considered in conjugation with the T cell (**Fig. 6f (3.7)**).

Cumulatively, our procedure allowed us to keep in the analysis bona fide conjugates while excluding big cellular clusters and cell fragments. Each conjugate included in the analysis was verified by independent analyzers, showing an accuracy of the combined procedure of $75,4\%$ on 52 lytic synapses.

In conclusion, the above results highlight a two-step approach, in which Kartezio can identify cells that are then classified by ML into different cell populations on the basis of their size and staining features. This offers the opportunity to explore automatic high-throughput scoring of defined parameters of immunological synapse formation and of other cell activation/death parameters in cell-cell conjugates. We show that Kartezio can synergize with state-of-the-art ML methods to create a hybrid AI approach capable of assessing and interpreting complex functional structures in biological images.

3.3.4 Discussion

In the present study, we release an innovative new computational tool called Kartezio for image segmentation of biological and medical images that draws inspiration from Darwinian evolution to automatically generate image processing pipelines. When compared head-to-head with DL approaches, Kartezio consistently achieved scores that were competitive with state-of-the-art approaches (**Fig.2 (3.3), Extended Data Table 2 (3.4)**), despite being trained on only a very limited number of input images. We applied our approach to four distinct Use Cases (**Fig. 3-6 (3.43.53.63.7)**, summarized in **Extended Data Table 3 (3.5)**) related to tumor immunotherapy, a highly dynamic research area in contemporary medicine wherein automated image

analysis has garnered significant interest [187, 188]. We validated Kartezio on tasks ranging from the characterization of nanometer-scale extracellular vesicles released by CTLs (average mask diameter of $\simeq 0.6\mu m$, based on manual user annotations) to micrometer-scale analysis of CTL phenotype (average mask diameter of $\simeq 12\mu m$, based on manual user annotations), and ultimately to centimeter-scale detection of cancer nodules in patient tissue samples (average estimated mask diameter of $\simeq 1.1cm$, based on pathologist annotations). This stringent testing strategy provided a crucial validation of Kartezio's capabilities across a wide variety of image types and across scales differing by five orders of magnitude.

A conceptual step-by-step summary of the Kartezio workflow is illustrated in **Extended Data Fig. 1 (3.8)**. The experimentalist first prepares the training and testing datasets by manually annotating a limited number of images. Then, the training images are provided to Kartezio, which generates initial pipelines composed of functions randomly drawn from the default function library, randomly parameterized and randomly arranged into an image processing pipeline. Each pipeline is evaluated against user defined annotations and the best pipeline is selected to proceed through the evolutionary selection process (illustrated in **Extended Data Movie 1**). The optimized pipeline proceeds to the testing phase and, if satisfactory, it can be then deployed on a previously unseen dataset. As demonstrated in the four Use Cases, once image segmentation (IS or SS) has been achieved using Kartezio, post-segmentation analysis of the segmented images can be performed, ranging from simple feature extraction from instances (e.g. MFI, size of particle, etc.) to ML-based analyses, depending upon the experimental design and

needs of the experimentalist (illustrated for the four Use Cases in **Extended Data Table 3 (3.5)**).

Our work tackles two major scientific challenges in the fields of CV and biomedicine. Firstly, it is becoming increasingly important to generate fully explicable image analysis strategies and to track the logic of decisions made by algorithms to ensure responsible patient management. Unlike Kartezio, the so-called "black box" DNN networks are difficult for humans to analyze and interpret, in spite of recent progress in the field of XAI [170, 171]. Limited interpretability is a potentially important drawback, particularly in the clinical context, where the process of extracting information from medical images should be fully transparent and interpretable in order to foster trust, ensure compliance with professional ethical standards, and facilitate regulatory oversight. This is also in line with the main principles of the European Union AI Act [189], currently under discussion by the European Council, that regulates the use of AI in sensitive fields such transportation and health care. As shown in **Fig. 1 (3.2) and Fig. 3 (3.4)**, as well as **Extended Data Fig. 3 (3.10)**, Kartezio is natively interpretable, meaning that the generated pipelines can be humanly read, evaluated and tested for provability. Given the potential applications for Kartezio in clinical pathology and related fields, its fully interpretable "white-box" nature presents a major conceptual advantage [153]. Recent advances in Large Language Model (LLM) are also in line with the quest for more interpretable data analysis pipelines. Indeed, LLM such as ChatGPT [190, 191], SantaCoder [192] and GitHub Copilot [193] can directly generate the code necessary to analyze a given image or dataset, provided that appropriate prompts are specified by the expert

(e.g. type of input image, expected output, functions to use in the analysis pipeline, etc.). Further research is ongoing to identify the optimal strategy to generate such prompts, in order to guide LLM towards the generation of adequate models. Finally, LLM and GP are currently being used to generate ML pipelines capable solving complex tasks [194, 195]. Each ML model could be trained to solve a specific and comprehensive task (e.g. extraction of tumor nodules, segmentation of cell nuclei, etc.) and the assembly of these individual ML models would build an interpretable pipeline at a higher scale. Interestingly, because of the low requirement of data and the simplicity to train the model, Kartezio could be used to produce these specific models before the pipeline is assembled at a higher level.

A second important challenge concerns image annotation, which is a time-consuming task that requires high biomedical expertise. It is therefore crucial to minimize experts' effort by designing ML algorithms that can handle the smallest datasets possible, while keeping adequate generalization capacity. To address this point, our study utilizes CGP to generate solutions that leverage the cumulative expertise of human CV experts. In other words, CGP produces pipelines by assembling algorithms that integrate decades of existing research and development in CV. While the number of images used in our study is already remarkably low, it is conceivable to upgrade Kartezio to an active "human-in-the-loop" training mode to further reduce the size requirement of the training dataset and improve performance. In such a configuration, intermediate results from Kartezio-generated pipelines could be displayed while the expert annotates. At a given point, the expert would not annotate anymore, but simply amend the automatically generated annota-

tion. In turn, expert-operated adjustments would be informative to pursue the training of Kartezio. In addition to these major conceptual advantages, Kartezio offers several concrete technical improvements over existing IS approaches. For example, Kartezio can handle images of diverse dimensions within the same dataset. In addition, it can be trained on small patches and later deployed on much larger images. As shown in **Fig. 3 (3.4)**, Kartezio can also be trained on low resolution images and the resultant pipelines applied to high resolution images, which removes the need to train the models on cumbersome high resolution images. Finally, Kartezio is not restricted to gray scale input images and can easily accommodate RGB, HSV, and HED formats. Nonetheless, there are certain limitations to bear in mind when assessing the applications of Kartezio. For example, while the pipelines shown in **Fig. 1 (3.2) and Fig. 3 (3.4)** are clear, streamlined and explicable, there is no guarantee that all Kartezio-generated pipelines will be so intuitive to understand. This is particularly salient if the predetermined number of nodes is high and/or the number of available functions in the function library is high. In the field of evolutionary computation, research is ongoing to take into account the size of the graph in the evaluation process using multi-objective optimization [196–198]. Another complementary approach to control graph size can come from automatic modularization strategies of produced graphs [199]. Further research is required to explore and apply these solutions to Kartezio.

Our approach is particularly notable for its integration of human knowledge into the image analysis process. Expert knowledge is integrated into three specific areas: **i)** the construction of the function library **ii)** the choice

of hyperparameters, and **iii)** the selection of non-evolvable components (in this case, preprocessing nodes and endpoints), discussed individually below. Firstly, for the design of the function library, specific expertise is necessary to determine the minimal set of functions that can adequately solve the task while keeping the dimensions of the search space as small as possible. For the image processing applications in this manuscript, the function library represents a comprehensive expert-curated list of the most common image processing functions in the literature, and as such certain redundancies between similar common functions are not unexpected. The existing list was designed to maximize the diversity of functions available to Kartezio while ensuring that all key types of function were represented, including: morphological (e.g. Erosion, Dilation), arithmetic (e.g. Add, Sub, Square), logical (e.g. And, Or, Not), and edge detection (e.g. Laplacian, Sobel), amongst others. Ongoing syntactic graph studies are currently underway to determine the frequency with which individual functions are represented in final pipelines and the propensity of given function pairs (bi-grams) or triplets (tri-grams) to be used together (e.g. Gaussian Blur followed by Sobel Filter). Clusters of functions typically used together can be entered into the function library as new functions, and rarely used or redundant functions can be removed. These decisions can be made based upon real data derived from the analysis of large numbers of generated pipelines. Such data-driven insights will enable the function library to be fine-tuned, and expert users have the option to further refine the function library, providing an additional opportunity for human knowledge to be integrated with the ML task. The current version of Kartezio contains 42 functions that have been manually

selected to address the specific tasks of this study. A possible improvement of Kartezio would be to automatically select functions included in existing programming libraries to generate *ad hoc* CGP libraries suitable for a given task. In this case study, Kartezio was applied to the task of IS, but this approach could clearly be applied to other tasks in the CV field as long as pre-existing functions are available and already used by human-made algorithms. Secondly, the selection of hyperparameters is an important consideration in the use of Kartezio. The hyperparameters currently used in this manuscript ($\eta = 30$ nodes, $\lambda = 5$ offspring over $\mathcal{K} = 20,000$ iterations) represent common hyperparameters in the CGP literature, and these could serve as default parameters or recommended settings for non-expert users of Kartezio. However, we envision that expert users may wish to modify certain hyperparameters to personalize the model to their needs and optimize performance, and further research is currently underway to assess the impact of alternative hyperparameters on generated pipelines. Thirdly, the addition of non-evolvable nodes (such as preprocessing nodes or endpoints) is an important mechanism for integrating human knowledge into the image analysis pipeline, a feature that is also seen in contemporary DL approaches. In this study, we provide evidence (shown in **Fig. 2c (3.3)**) that pipelines given a non-evolvable endpoint appropriate to the objective (in this case, Marker Controlled Watershed for IS) significantly outperform those with no endpoint indicated. We envision that non-evolvable endpoints should initially be selected by expert users based upon the specific objective of the image processing task (e.g. Hough Circle Transform for segmentation of round objects, MCW for IS), and these would become the suggested default settings for non-

expert users, who could choose from recommended non-evolvable nodes based on indicated task objective. Likewise, as demonstrated in **Fig. 2c (3.3)**, appropriate preprocessing nodes can be selected based on the properties of the input image, and their impact on model performance quantitatively assessed. Ultimately, automation of non-evolvable node selection may also be a powerful approach. Although the design of the function library as well as the selection of hyperparameters and non-evolvable components are by nature subjective, the impact of these components on the performance of the model can be objectively assessed (as shown in **Fig. 2c (3.3)**), which enables validation of the expert knowledge integrated into this approach. It is important to note that in the present configuration, Kartezio only handles images. However, in biomedicine a plethora of data is collected in different formats and must be analyzed in an integrated fashion. In biology, these data may be derived from diverse molecular and functional experiments, as well as from microscopy. In medical practice, these clinical data may include laboratory test results, demographic, lifestyle and prognostic factors, and imaging data from diverse sources (e.g. histopathology, PET scans, etc). MT-CGP [141] is a promising approach to handle and integrate heterogenous data, such as scalars, vectors and matrices. Upgrading Kartezio in this direction would allow us to create data analysis pipelines that, while remaining fully explainable, would integrate data of different kind and origin. In clinical routine, such an approach would endow practitioners with AI-based clinical assistants that would be compatible with the upcoming regulatory guidelines. In biological research, only a fraction of generated data is included in publications. Experimental data are indeed often analyzed only to

answer specific questions, but are rarely used for broader integrated analyses. Such deep analyses are time and energy consuming and might not provide results immediately beneficial to the advancement of a given research. They may therefore be neglected. Kartezio could be trained at low human cost on small datasets to automatically extract predefined features (geometrical information from images, intensity of staining from flow cytometry data, activation of death pathways in target cells, single cell mRNAseq data, etc). These extracted data would feed large pre-analyzed data bases that could be used by data scientists to address new questions using mathematical and ML approaches. An additional important benefit of Kartezio's application to microscopy concerns the necessity of limiting the subjectivity of the experimentalists who analyze and scores the images. A way to bypass the problem of subjective image scoring is the use of high-throughput techniques such as high-content imaging and imaging flow cytometry (e.g. ImageStreamTM) that permit the rapid acquisition of thousands of biological images and subsequent analysis of these images using dedicated bulk software [200]. While these high-throughput imaging techniques are very powerful and are useful under many circumstances, they lack resolution. As a consequence, their utility is limited to the analysis of bulk cellular/subcellular structures. The application of Kartezio to high resolution, super-resolution, and even electron microscopy images can in principle ensure automated non-subjective analysis with no compromise in terms of resolution. The ability to reconcile automated image analysis with high resolution imaging is particularly important in the field of nanomedicine, wherein nanotechnologies are applied to diagnostics, therapeutics, vaccines etc. Indeed, characterization of

drug vectors and bioactive particles requires the implementation of methods that allow researchers to effectively evaluate and optimize parameters such as particle size and composition. In this sense, Kartezio offers a first stepping stone to address this issue by rapidly creating pipelines for analysis of subcellular structures, such as CTL-derived extracellular particles (**Fig. 4 (3.5)**). Further optimization of Kartezio is required to make it a reliable "pharmacologist assistant" for validation of nanometer/micrometer size therapeutical preparations. In summary, Kartezio offers a robust, flexible, and ground-breaking approach to automated image processing. It facilitates the generation of specialist models starting from small or very small datasets. Combined with its low hardware requirements, this confers upon Kartezio an exceptional level of versatility and frugality, broadening its potential applications to include contexts in which large datasets or powerful hardware are unavailable. Although in our study we applied Kartezio to biomedical images, its application can be extended to different fields. It is interesting to note that one of the first application of CGP to image analysis comes from a study in which CGP was applied to images acquired by a Mars rover [142]. Taken together, our study and these pioneering observations strongly suggest that CGP will become increasingly important in the development of CV and its application to different aspects of human life.

3.3.5 Material and Methods

Mathematical Modeling

Kartezio was developed using Python3 [201] programming language and built mainly over NumPy [202], OpenCV [120], Scikit-image [203], and Scikit-learn [186]. For the visualization, images and plots were generated using Matplotlib [204], Seaborn [205], OpenCV [120], and Fiji [206].

Cartesian Genetic Programming

Genotype. CGP considers a specific encoding of the syntactic graph within integer-based genotype (Paper Fig. 1a). Any integer value of the Genotype is called a gene and each node $\mathcal{N}_i$ of the graph is identified with the unique integer value i, to which the other graph nodes connect. In CGP, the genotype $\mathcal{G}$ (corresponding to the phenotype $\mathcal{P}$) is an $m-by-n$ matrix where m is the number of nodes and is equal to the total number of functional nodes (η) and output nodes (o). n corresponds to the size of one encoded node (i.e. the number of genes in one node) which relies on 3 values based on the function library $\mathcal{L}$. The first gene corresponds to the unique function index. Then, the next α (called arity) genes correspond to the connection of the graph and so the inputs of the function. α is determined by the maximum number of inputs of the functions from the function library $\mathcal{L}$. Lastly, the node is ended with ρ integers corresponding to additional parameters used by the function. Formally, the genotype can be written as 2 sub-matrices:

Nodes. A functional node $\mathcal{N}_i$ is a 1-D vector composed of 3 significant

Kartezio Hyperparameters	
Symbol	**Description**
$\eta \in \mathbb{N}_{\geq 1}$	Number of functional nodes for the CGP model
$\lambda \in \mathbb{N}_{\geq 1}$	Number of children of the population involved in Evolution Strategy $1 + \lambda$
$\mu \in \mathbb{R}[0, 1]$	Mutation probability for functional nodes
$\nu \in \mathbb{R}[0, 1]$	Mutation probability for output addresses
$\mathcal{K} \in \mathbb{N}_{\geq 1}$	Number of iterations for the evolution loop
Kartezio Induced Parameters	
$\iota \in \mathbb{N}_{\geq 1}$	Number of input channels for CGP model; induced by the size of X
$o \in \mathbb{N}_{\geq 1}$	Number of intermediate outputs for CGP model; induced by the arity of the Endpoint
$\alpha \in \mathbb{N}_{\geq 1}$	Number of edges feeding a node (arity); induced from the Function Library $\mathcal{L}$
$\rho \in \mathbb{N}_{\geq 0}$	Number of parameters used by the functions; induced from the Function Library $\mathcal{L}$
$\varphi \in \mathbb{N}_{\geq 1}$	Size of the Function Library $\mathcal{L}$
Kartezio Data	
$x_i \in X$	One single input (2D channel in case of images)
$\hat{z}_i \in \hat{Z}$	One single intermediate heuristic (2D channel in case of images)
$\hat{y}_i \in \hat{Y}$	One single finale output (any type, usually 2D masks or labels)
$y_i \in Y$	One single annotation (any type, usually 2D masks, annotations)
$X: \{x_1 \cdots x_\iota\} \in \mathcal{X}$	Input vector of ι channels corresponding to one entry of a Dataset $\mathcal{D}$
$\hat{Z}: \{\hat{z}_1 \cdots \hat{z}_o\}$	Intermediate output vector of o heuristics feeding the Endpoint $\mathcal{E}$
$\hat{Y}: \{\hat{y}_1 \cdots \hat{y}_n\}$	Finale n outputs produced by the Endpoint $\mathcal{E}$ ($n \geq 1$; usually $n = 1$)
$Y: \{y_1 \cdots y_n\} \in \mathcal{Y}$	Ground truth annotations ($n \geq 1$; usually $n = 1$)
$\mathcal{X}: \{X_1 \cdots X_n\}, \mathcal{X} \in \mathcal{D}$	Training or Test set; list of n ($n \geq 1$) entries X
$\mathcal{Y}: \{Y_1 \cdots Y_n\}, \mathcal{Y} \in \mathcal{D}$	Training or Test set annotations; list of n ($n \geq 1$) entries Y
$\mathcal{D}: \{\mathcal{X}, \mathcal{Y}\}$	Training or Test Dataset composed of inputs $\mathcal{X}$ and correspond annotations $\mathcal{Y}$
Kartezio Model Components	
$\mathcal{G}$	Genotype (vector of integers)
$\mathcal{P}$	Phenotype (directed acyclic graph)
$\mathcal{N}$	Node
$\mathcal{L}: \{f_1 \cdots f_\varphi\}$	Function library composed of φ ($\varphi \geq 1$) functions f
Kartezio Operators	
$\delta: \mathcal{G} \rightarrow \mathcal{P}$	Decodes to parse one Genotype to the active Phenotype
ψ	Executes the Phenotype function sequence
$\mathcal{A}$	Aggregation (3D specific), function that stack predictions from different image layers
$\mathcal{E}$	Endpoint, acting like a Node to produce the final output $\hat{Y}$
$\mathcal{F}$	Fitness minimization function, generating a score for the Genotype

Table 3.2: Table 1 | Parameters

segments described as follows:

where the integer b_i indexes one of mathematical functions from $\mathcal{L}$ (Paper Extended Data Table 1), C_i are addresses of upstream nodes within the graph that will be used as inputs of the node's function and P_i are constant (but optimized) parameters of the function. All functions from the library must be deterministic and must return one 2D image which is identical to the input image in dimensions and type (generally unsigned integer encoded on 8bits, noted *u8*). Most of the time, one node operation consists of one function call from the OpenCV Python package. All functions must be image-size independent, which implies that the input and output for a given image is equivalent. However, there is no image size constraint for the model in both training and test datasets.

The first ι nodes, noted as $X\{x_1 \dots x_\iota\}$ are the receiving inputs (e.g. image channels) while the last o nodes represents the intermediate outputs $\hat{Z}\{\hat{z}_1 \dots \hat{z}_o\}$. An example of genotype $\mathcal{G}[\iota = 3, \eta = 4, o = 2, \alpha = 2, \rho = 2]$ with its nodes is illustrated in Paper Fig.1.

Non-evolvable Preprocessing Node. The first stage of a standard image processing pipeline is the preprocessing or/and formatting of the raw data. Depending on the input images (e.g. transmission light images, fluorescent images, IHC tissues images, etc.) and the expected outcomes (e.g. cell identification, event counting, cell distance and interaction, etc.) needed to address a given biological question, the user can impose a preprocessing of the dataset. When loading the dataset, the preprocessing stage is done on-the-fly and produces a formatted input X for each entry of the dataset. For example, the default behavior for a standard bitmap image is

$X_{rgb}[x_1 = r, x_2 = g, x_3 = b]$. For specific needs of certain datasets, any function can be used for preprocessing (e.g. HSV transformation, pyramidal mean shift filtering, etc.) including those requiring all three channels to operate. This can help evolution by providing important insight given by the domain expert.

Non-evolvable Endpoint. We also introduce a new component extending CGP to IS using a determined final function called Endpoint. It is the last processing step before evaluation. It is a substantial bottleneck, not subject to optimization, of the syntactic graph. Moreover, the endpoint is the only node allowing the outcome size and type to change. We use Watershed Transform [34] as the main algorithm to produce IS. In cell segmentation, the cell nuclei are usually stained or visible, so it is possible to deduce good markers for a MCW ($o = 2$) [34]. These two endpoints produce a label map that is then assessed. In SS, the default endpoint is a Threshold (binary or to zero).

Genotype Decoding and Graph Execution. To generate an output from given inputs, the CGP genotype $\mathcal{G}$ is first decoded to produce its phenotype $\mathcal{P}$ (Equation 1). $\mathcal{P}$ is then executed to produce the intermediary results $\hat{Z}_{\mathcal{G},X}$ (Equation 2) which is provided as the input of the Endpoint (Equation 3) to calculate the final output $\hat{Y}_{\mathcal{G},X}$.

$$\mathcal{P} = \delta(\mathcal{G}) \tag{3.1}$$

$$\hat{Z}_{\mathcal{G},X} = \psi(\mathcal{P}, X) \tag{3.2}$$

$$\hat{Y}_{\mathcal{G},X} = \varepsilon(\hat{Z}_{\mathcal{G},X}) \tag{3.3}$$

When used on 3D images, Equation 1 is used only once to obtain a unique syntactic graph which will be used for all z-sections. Equation 2 is executed for each z-section X_z of the image. All $\hat{Z}_{\mathcal{G},X_z}$ are then aggregated through the A operator (averaged in the example Equation 4) to obtain the intermediary output $\hat{Z}_{\mathcal{G},X}$, which is in turn feeding the endpoint (Equation 3) to produce the final output $\hat{Y}_{\mathcal{G},X}$.

$$\begin{aligned} \hat{Y}_{\mathcal{G},X} &= \varepsilon(\mathcal{A}(\hat{Z}_{\mathcal{G},X})) \\ &= \varepsilon(\frac{1}{N} \times \sum_{z=1}^{N} \hat{Z}_{\mathcal{G},X_z}) \end{aligned} \tag{3.4}$$

Evolution. Kartezio uses the $1 + \lambda$ ES. A first population ($k = 0$) of genotypes $\mathcal{G}_{k=0}^{n}$ with $n \in [0, \lambda]$ is initialized randomly and independently. These genotypes then follow an evolutionary loop in which they will first be evaluated by a defined fitness function F and the best genotype $\mathcal{G}_{k=0}^{*}$ selected based on evaluation score to produce the next parent genotype. $\mathcal{G}_{k=1}^{*}$ is then mutated to produce λ offspring $\mathcal{G}_{k=1}^{n}$ with $n \in [1, \lambda]$. The evolutionary loop (evaluation, selection, mutation) is repeated a finite number of $\mathcal{K}$ iterations to produce the final genotype $\mathcal{G}_{\mathcal{K}}^{n}$. The following section details each evolu-

tionary operator, the $1 + \lambda$ ES.

Evaluation. To evaluate a genotype $\mathcal{G}$, this genotype is first decoded as describe previously. The resulting graph is executed on each image of the training dataset $\mathcal{X}_{train}$ to obtain a set of predictions $\hat{Y}_{\mathcal{X}}$. Predictions are compared to the ground truth of the dataset $\mathcal{Y}_{train}$ using a specific fitness function $\mathcal{F}$ which evaluates the prediction errors. This function is domain-specific (e.g. AP50 for IS or IoU for SS) and must be defined for each task Kartezio is used for. The genotype evaluation $\text{fitness}_{\mathcal{G}}$ is finally given by the average of prediction error of each dataset entry:

$$\text{fitness}_{\mathcal{G}} = \frac{1}{N} \times \sum_{i=1}^{N} \mathcal{F}(\mathcal{Y}_{train_i}, \hat{Y}_{\mathcal{X}_i})$$

Selection. The objective is to minimize the prediction error over many iterations. The best genotype is therefore selected among the offspring and the parent by selecting the smallest prediction error. If an offspring obtains a fitness strictly equal to the current parent, they are distinguished by minimizing the execution time of the graph to promote frugality in term of computational effort.

Mutation. The parent generates λ new offspring by mutation. Mutations consist of randomly altering random values of the genotype. Kartezio utilizes "Accumulate" mutation behavior proposed by Goldman [207], which implies that mutations accumulate until the active graph changes to avoid useless evaluations. Random indices (i, j) are sampled in the whole functional sub-matrix: $\mathcal{G}_{i,j}$ gets a new independent value r, sampled as follows:

$$\mathcal{G}_{i,j} = \begin{cases} r \in (1, \varphi) & \text{if } j = 1 \\ r \in (1, i) & \text{if } 1 < j < 1 + \alpha \\ r \in (0, 255) & \text{if } 1 + \alpha < j < 1 + \alpha + \rho \end{cases} \tag{3.5}$$

The number of altered $\mathcal{G}_{i,j}$ genes is calculated to respect the mutation rate parameter: for instance, if $\mathcal{G}$ has 100 genes and $\mu = 0.1$, then 10 genes will be mutated. Lastly, each output address has a probability to be replaced by a new random value $r \in [1, \iota + \eta]$.

Benchmarking. We trained the models with a $\eta = 30$ nodes, $\lambda = 5$ offspring over $\mathcal{K} = 20,000$ iterations. If the parent reached a score of 0, the evolution process stopped as no prediction error was made on the training dataset. The mutation probability was $\mu = 0.15$ for the functional nodes and $\nu = 0.2$ for the outputs. 35 independent runs were made for each experiment for statistical purposes. They were run on a bi Intel Xeon @6140 (2×18 cores, 2.30GHz - 3.70GHz) with 192GB of memory.

Applications of Kartezio

Standardized Testing Using Published Cell Image Library (Paper Fig. 2, Paper Extended Data Fig. 2a). We compared Kartezio to three state-of-the-art IS algorithms: Cellpose, StarDist and Mask R-CNN. To this end, the dataset and the results for these three algorithms were taken from published data [39]. The dataset was composed of 100 images of in vitro cultured neurons stained for phalloidin and DAPI (as shown in Paper Ex-

tended Data Fig. 2a) and imaged using an epifluorescence microscope [39]. This 100-image dataset was split into 89 images for training and 11 images for testing [39]; the training/test split was retained in this manuscript in order to permit direct comparison of Kartezio with Cellpose under standardized conditions. No data augmentation was necessary for our approach. In this study case, Kartezio was composed of $\iota = 2$ inputs, corresponding to the a-tubulin and DAPI channels, and $o = 2$ outputs, corresponding to the mask and markers necessary for the Watershed Transform non-evolvable endpoint. Generated syntactic graphs were evaluated using the Average Precision ($AP = \frac{TP}{TP+FP+FN}$) metric calculated with the standard IoU between the predicted mask and the ground truth. A threshold of $t = 0.5$ was applied to decide the correctness (true-positive) of the predicted mask, as described [39]. Kartezio was benchmarked on different training dataset sizes, ranging from 1 image to 89 images. For each size of dataset, images were selected at random from the original 89 training images. Each trained solution was then evaluated using the previously described dataset of 11 test images [39]. 35 independent training runs were performed, each time randomly selecting different training images from the original set of 89 to control for the impact of training split composition. The AP scores reported in Paper Fig. 2 correspond to the mean and standard deviation of these 35 runs.

Standardized Testing Using Melanoma Cell Nodules (Paper Fig. 2, Paper Extended Data Fig. 2b). To extend these results, we tested whether using a specific non-evolvable endpoint (namely the MCW Endpoint) improved Kartezio performance beyond that which was achieved in the absence of a non-evolvable endpoint, using Connected Component to ob-

tain evaluable labels. In the first case (Paper Fig. 2c, green dots), the model must optimize 2 heuristics: the mask and the markers aimed at producing the best IS (see Paper Fig. 1). In the second case (Paper Fig. 2c, magenta dots), the model must learn to distinguish the masks (future labels) directly, which is similar to the CGP-IP version for segmentation. In Paper Fig. 2c, we demonstrate statistically (one-way ANOVA with multiple comparisons) the importance of MCW, on a melanoma nucleus dataset (nuclei stained as orange, Paper Fig. 2, Paper Extended Data Fig. 2b). We also compared the use of different pre-processing steps (RGB, HSV, or HED) to generalize the comparison. For each of the six conditions, we performed 10 independent runs. Finally, we evaluated the predictions of the best Cellpose pre-trained model (model *CPx*, on blue channel) in order to estimate the generalist model performance on the same test set. Among the 30 MCW models, we selected a model with less than 5 active nodes to illustrate Paper Fig 1. This same model was converted into a Python class (Paper Extended Data Fig. 3) and finally its predictions are illustrated on a test image (512x512) shown in Paper Fig. 2d. In this figure, the first panel shows the manual annotations (green), the second panel shows the Kartezio model predictions (magenta) on top of the annotations and similarly for the last panel with the Cellpose model predictions. An important point to note, the Kartezio model can process more than 100 images like this one (512x512) per second on a single CPU, illustrating the frugality of this approach.

Following this, we assembled four Use Cases with different model inputs and outputs (summarized in Paper Extended Table 4) to illustrate Kartezio's versatility.

Use Case 1 - Tumor Nodules (Paper Fig. 6, Paper Extended Data Fig. 2e).

24 WSIs from the clinical cohort was assembled for training and testing, including 12 from each clinical site (Institut Universitaire du Cancer de Toulouse and Centre Hospitalier Universitaire de Bordeaux). WSIs corresponded to a 9-level pyramidal image from which we extracted a low resolution thumbnail of dimension 461x1024 pixels (hereafter known as the "training level") using OpenSlide library [208]. These training level images served as the input dataset for Kartezio and were subject to expert pathologist annotation to establish the ground truth for this Use Case. Images were allocated in a randomized fashion into the training (n = 12) and testing (n = 12) sets (summarized in Paper Extended Data Table 3). Images were preprocessed by transforming R-G-B channels into H-S-V color space (shown in Paper Extended Data Fig. 2e). As summarized in Extended Data Table 4, Kartezio's models were trained with $\iota = 3$ inputs and terminated with a Threshold to zeros endpoint ($o = 1$). Evaluation was made using the simple IoU Fitness function. An ensemble of 100 models was trained to segment tumor nodules. Each prediction from individual models was normalized (minmax scaler). All 100 predictions were merged by averaging the values pixelwise. All in all, the heat map $\mathcal{H}$ of one slide X is obtained by:

$$\mathcal{H} = \frac{1}{N} \times \sum_{i=1}^{N} \frac{\hat{Y}_{\mathcal{G}_i,X} - min(\hat{Y}_{\mathcal{G}_i,X})}{max(\hat{Y}_{\mathcal{G}_i,X}) - min(\hat{Y}_{\mathcal{G}_i,X})}$$

To determine the optimal threshold for the heatmap, we performed incremental evaluation by increasing the final threshold t from 0.0 to 1.0 at a step

of 0.01. This provided an evaluation curve (shown in Paper Extended Data Fig. 4) where the optimal threshold was found to be $t = 0.35$. The heatmaps were then thresholded ($t = 0.35$) to produce the heatmaps presented in Paper Fig. 3. To determine whether the models generated on the low resolution *"training level"* thumbnails could be upscaled to high resolution images, two strategies were employed using an illustrative example region from the IHC dataset (Paper Fig. 3a, example region indicated with purple box). In a first approach, predictions were performed on the original *"training level"* image (461x1024 pixels) and the resulting heatmap upsampled using a bilinear resizing function to fit Level 7 (1405x3123 pixels) and Level 5 (5621x12494 pixels) of the image pyramid (Paper Fig. 3c). In a second approach, the ensemble of 100 models trained on the original training level (461x1024) image was applied to Level 7 and Level 5 images to generate new masks without any retrain phase (Paper Fig. 3d).

Use Case 2 - Extracellular Particles (Paper Fig. 3, Paper Extended Data Fig. 2b). Because of the intensity difference between channels (shown in Paper Extended Data Fig. 2b), the dataset was split into two datasets, one to generate a WGA specialist model and one to generate a DiO specialist model. The same procedure was applied to both WGA and DiO datasets. Each dataset included one training image and one testing image of CTL-derived extracellular particles. Training and testing sets were augmented by splitting each image into four quarters. The final post-segmentation analysis was made using the test and training images combined with nine previously unseen images. Use Case 1 dataset preparation is summarized in Paper Extended Data Table 3. As shown in Paper Extended Data Table 4, Kartezio

used $\iota = 1$ channel either for WGA or for DiO and $o = 1$ outputs that produces mask. This mask was passed to the Local-Max Watershed non-evolvable endpoint, which consisted of creating a DT map from the mask to find local-max (peaks). Together the peaks and the smoothed gradients of the DT were provided as input to Watershed Transform. Generated syntactic graphs were evaluated as described in the first study case. Once trained, both models generated masks containing prediction of WGA particle and DiO particle instances. The two masks were merged using the same procedure to evaluate mask prediction to expected ground truth (IoU with a threshold of 0.05).

Use Case 3 – CTL Lytic Arsenal (Paper Fig. 4, Paper Extended Data Fig. 2c).

From the original set of 19 images, 5 images were selected for the training dataset and 4 for the test dataset. Each 3D image was composed of 4 channels. Use Case 2 dataset preparation is summarized in Paper Extended Data Table 3. To reduce the computational cost, we resized the images (halved the size). We chose a threshold $t = 0.7$ for the fitness function (AP70) to be more stringent with the predicted mask. To process a 3D image, we provided the z-sections of $\iota = 1$ channel (CD45) to the model, which generates $o = 1$ mask (as summarized in Paper Extended Data Table 4). All the masks were averaged to produce one final mask of the whole 3D image. We then applied the Local-Max Watershed Endpoint to produce the final 2D IS. Cells were segmented from the 19 original images using the best model out of 35 runs. 161 cells were filtered from predicted cells using their pixel area ($2000px < area < 14000px$). For each cell, we calculated intensity features

(see below) based on the perforin, granzyme B and CD107a channels (shown in Paper Extended Data Fig. 2c). All the cell's feature vectors were then used to generate the UMAPs [184] of Paper Fig. 4. To this end, we used the following parameters to the UMAP function: dimensionality reduction down to 2 components, using following parameters: *n_neighbors=15, min_dist=0.01.*

Use Case 4 - Lytic Synapses (Paper Fig. 5, Paper Extended Data Fig. 2d). Four 3D images (using the z-stack function) were acquired. These large images were cut into a 3x3 grid, with a depth interval set for each tile (adapted to the focus). Images with high background noise and culture well edges were removed, thus giving a total of 32 selected images (representative image shown in Paper Extended Data Fig. 2d). We randomly extracted 8 images for the training set and 4 for the test set (summarized in Extended Data Table 3). As shown in Paper Extended Data Table 4, the models handled $\iota = 2$ channels (a-tubulin, DAPI) and used the MCW endpoint ($o = 2$). Similar to the previous application, each z-section produced one markers map and one mask map, which were averaged individually before being passed to the Endpoint. In that case, the models segmented cells with no distinction of types. Over the 32 images, predicted cells were filtered by pixel area ($350 < area < 6000$). Segmented cells were then classified as either CTLs or target cells by using their intensity feature vector (see below) combined with their area. Unsupervised learning was used for classification using first PCA (*n_components="mle"*) followed by a GMM (*n_components=2, covariance_type="tied"*). Finally, CTL/target cell synapses were detected when the Euclidian distance between the centroids of a CTL and a target cell was less than 70px ($\simeq 18.5nm$). The target cell masks of potential synapses were

then dilated (morphological dilation) to assess potential overlapping with CTL mask. A valid CTL/target cell synapse was detected if at least one pixel was overlapping.

Intensity Feature Vector. This vector contained individual and combined counts of positive pixels, the sum, and the average pixel fluorescence intensities. Combined counts were calculated by first applying a binary threshold to determine the positive pixels within each channel. Each pixel of a given cell mask incremented a counter corresponding to its combination of positive channels. All the counters, augmented with the total count, the sum and the average fluorescence intensity for each channel were gathered into the intensity feature vector. The length ℓ of the vector was determined by the number of channels $\mathcal{C}$: $\ell = 2^{\mathcal{C}} + 3 \times \mathcal{C}$

Image acquisition and processing

Preparation of extracellular particles and single particle imaging For preparation of extracellular particles (EPs), CD8+ human primary T cells were isolated from healthy donor blood with negative selection kits (RosetteSep Human CD8+ T cell Enrichment Cocktail, STEMCELL Technologies, UK) following the manufacturers instruction. Peripheral blood samples were provided from the National Health Service under ethics agreement number 11/H711/7. CD8+ cells were expanded after isolation by addition of anti-CD3/anti-CD28 T-cell activation beads (Dynabeads ThermoFisher Scientific, UK) at 25 μl/ 106 cells and with 5000 Units/ml recombinant IL-2 for 3 days following the manufacturer's instructions. Subsequently, the magnetic beads were removed and cells were rested for 4 days at 106 cells/ml. For all incuba-

tion steps, cells were cultured in RPMI (Gibco) medium supplemented with 10% fetal bovine serum, 1% penicillin/streptomycin, 1% L-glutamine, 1% non-essential amino acids and 50 mM HEPES in cell culture treated flasks at 37°C, % CO2 and 100% humidity.

After resting, cells were transferred to fully-supplemented culture medium containing exosome-depleted serum (Thermo Fischer Scientific, UK) and cultured for 48 hours. For EP isolation, the culture medium was subsequently collected (total of 50 ml) and cells were removed by centrifugation for 10 min at 300 g. Larger particles and cell debris were removed by centrifugation for 20 min at 900 g. Subsequently, EPs were collected by centrifugation for 60 min at 100,000 g and at 4°C. The EP pellet was resuspended in 1 ml phosphate buffered saline (PBS) and stored until further use at -20°C. Repeated thawing and freezing cycles were avoided.

The EP solution (total protein concentration of 137 μg/ml) was diluted 1:10 with PBS and stained with a final concentration of 40 μg/ml AlexaFluor647-coupled WGA (ThermoFisher Scientific, UK) and 20 μM DiO (ThermoFisher Scientific, UK) for 1 hour at 4°C in the dark. 150 μl of the EP sample were then deposited onto clean glass coverslip (SCHOTT UK Ldt, Stafford, UK) fixed to flow chambers (sticky- Slide VI 0.4, Ibidi, Thistle Scientific LTD, Glasgow, UK) that were coated with 0.1 mg/ml poly-L-lysine (Sigma Aldrich, UK) for 15 min and washed twice with PBS before addition of EPs. The EPs were incubated for 30 min at 4°C in the dark to allow for EP binding on the surface. Excess dye and non-bound EPs were washed from the flow channels by washing twice with PBS. Samples were subsequently imaged by TIRFM using an Olympus IX83 inverted microscope (Keymed, Southend-

on-Sea, UK) equipped with 405-nm, 488-nm, 561-nm and 640-nm laser lines and a Photometrics Evolve delta EMCCD camera. Images were acquired with a 150x 1.45 NA oil-immersion objective.

Confocal microscopy Staining for CTL lytic components. Total CD8+ T cells were purified from healthy donor blood samples using the RosetteSep Human CD8+ T Cell Enrichment Cocktail (StemCell Technologies) following the manufacturers instruction. Blood samples were collected and processed following standard ethical procedures after obtaining written informed consent from each donor and approval by the French Ministry of the Research (transfer agreement AC-2020-3971). Approbation by the ethical department of the French Ministry of the Research for the preparation and conservation of cell lines and clones starting from healthy donor human blood samples has been obtained (authorization no. DC-2021-4673). CD8+ T cells were cultured in RPMI 1640 medium supplemented with 5% human AB serum (Institut de Biotechnologies Jacques Boy), 50 μM 2-mercaptoethanol, 10 mM Hepes, 1% MEM-Non-Essential Amino Acids (MEM-NEAA) Solution (Gibco), 1% sodium pyruvate (Sigma-Aldrich), ciprofloxacin (10 μg/ml) (AppliChem), human recombinant interleukin-2 (rIL-2; 100 IU/ml), and human rIL-15 (50 ng/ml) (Miltenyi Biotec). CD8+ T cells were collected 7 days after activation using anti-CD3/anti-CD28 T-cell activation beads (Dynabeads ThermoFisher Scientific) at 2.5 μl/ 106 cells. Cells were washed in serum-free RPMI 1640 and allowed to adhere to poly-L-lysine (Sigma Aldrich) coated slides (Erie Scientific, ER-208B-CE24) at 37°C, 5% CO2 for 10 min. Cells were then fixed with 3% paraformaldehyde (MP Biomedicals) for 10 min at room temperature, washed twice with PBS and subsequently per-

meabilized with 0.1% saponin (in PBS/3%BSA/HEPES). Cells were stained in a two-step process with primary antibodies followed by isotype matching secondary antibodies all at 10 μg/ml. Both steps were performed for 1h at room temperature. We used the following antibodies: Anti-human perforin mAb (clone δG9, IgG2b, BD 556434) followed by goat anti-mouse IgG2b AlexaFluor647 (Thermofisher A-21242), anti-human CD107a rabbit Ab (polyclonal, Abcam ab24170) followed by goat anti-rabbit AlexaFluor555 (Thermofisher A32732), anti-human granzyme B mAb (clone GB11, IgG1, Thermofisher MA1-80734) followed by goat anti-mouse IgG1 AlexaFluor488 (Thermofisher A-21121), and anti-human CD45 rat Ab (clone YAML501.4, Thermofisher MA5-17687) followed by goat anti-rat AlexaFluor405 (Thermofisher A48261). The samples were mounted in 90% glycerol-PBS containing 2.5% DABCO (Sigma). Randomly selected cells were imaged with a LSM 780 confocal microscope equipped with a 63x-NA 1.4 oil immersion Plan-Apochromat objective (Zeiss), zoom 1.0. Images were acquired as z-stacks with an interval of 0.75 μm.

Staining for immunological synapse detection. Target cells were either unpulsed or pulsed with 10 μM antigenic peptide for 2 h at 37°C in RPMI 5% FCS/HEPES and washed three times. Conjugates were formed by 1 min centrifugation at 455g, incubated for 15 min, gently disrupted and seeded on Poly-L-Lysin coated slides and fixed with 3% paraformaldehyde at 37°C. Cells were then permeabilized with 0.1% saponin (in PBS/3%BSA/HEPES), and stained with anti-human perforin mAb (10 μg/ml, clone δG9; BD Pharmingen) followed by AlexaFluor555 goat anti-mouse IgG2b (10 μg/ml; Invitrogen) and anti-α-tubulin mAb (1 μg/ml, clone DMA1; Sigma) followed by

AlexaFluor488 goat anti-mouse IgG1 (10 μg/ml; Invitrogen). The samples were mounted in 90% glycerol-PBS containing 2.5% DABCO (Sigma) supplemented with DAPI (1μg/ml; Invitrogen) and examined using a LSM 780 (Zeiss) confocal microscope over a 63x Plan-Apochromat objective (1.4 oil). 3D images (using the tile scan and z-stack functions) were acquired.

Patient cohort The established melanoma clinical cohort (previously described [2]) was comprised of patients treated for advanced melanoma both at the Oncodermatology Department of the Institut Universitaire du Cancer de Toulouse (IUCT) and at the Centre Hospitalier Universitaire (CHU) de Bordeaux. Samples in Toulouse were stored at the CRB Cancer des Hôpitaux de Toulouse collection. In accordance with French law, this cancer collection was declared to the Ministry of Higher Education and Research (DC-2020-4074) and a transfer agreement was obtained (AC-2020-4031) after approbation by ethical committees. Samples in Bordeaux were stored at the Cancer Biobank of CHU Bordeaux collection. In accordance with French law, this cancer collection was declared to the Ministry of Higher Education and Research (DC 2014-2164) and a transfer agreement was obtained (AC-2019-3595) after approbation by ethical committees. Patients with available formalin-fixed, paraffin-embedded (FFPE) tumor blocks at the Department of Pathology were included in the analysis.

Immunohistochemistry As previously described [2], CD8, CD107a and Sox10 were visualized by multiplex IHC for the entire tumor nodule and surrounding tissue. The Discovery ULTRA (Ventana Medical Systems, Innovation Park Drive Tucson, Arizona 85755 USA, ROCHE) was used for automated staining. After dewaxing, tissue slides were heat pre-treated us-

ing CC1 buffer (05424569001, ROCHE). Slides were stained using the RUO Discovery Universal procedure (v0.00.0370) in a 3 step protocol with sequential denaturation (CC2 buffer (pH6), at 100°C, 05279798001, ROCHE). Tissue slides were incubated using primary antibodies CD107a (clone D2D11, #9091, Cell Signaling Technology, Inc.), CD8 (clone C8/144B, M7103, Agilent Technologies) and Sox10 (clone SP267, 07560389001, ROCHE) in Envision Flex diluent (K800621-2 Agilent Technologies). Targets were then linked using the OmniMap anti-rabbit (05269679001, ROCHE) and OmniMap anti-mouse (05269652001, ROCHE) HRP conjugated secondary antibodies. Visualization of the different targets was performed using Discovery Silver (07053649001, ROCHE), Purple (07053983001, ROCHE) and Yellow (07698445001, ROCHE) detection kits. Tissue slides were counterstained using Haematoxylin (05277965001, ROCHE) enhanced by Bluing reagent (05266769001, ROCHE) and mounted with xylene-based mounting medium (Sakura TissuTek Prisma, Sakura Finetek Europe B.V., The Netherlands). IHC slides were then digitized with a Panoramic 250 Flash II digital microscope (3DHISTECH, Budapest, Hungary) equipped with a Zeiss Plan-Apochromat 20x NA 0.8 objective and a CIS VCC-FC60FR19CL 4 megapixels CMOS sensor (unit cell size 5.5 x 5.5) mounted on a 1.6x optical adaptor, to achieve a scan resolution of $0.24\mu m$/pixel in the final whole slide image. A set of twenty-four whole-slide images from the clinical cohort was assembled for training and testing, including twelve images from each clinical site (Institut Universitaire du Cancer de Toulouse and Centre Hospitalier Universitaire de Bordeaux). Images were allocated in a randomized fashion into the training (n = 12) and testing (n = 12) cohorts prior to analysis. Statistical

Analysis All statistical analyses were performed using GraphPad Prism 9 and are two-sided unless otherwise indicated. Means are compared using a one-way ANOVA with multiple comparisons. Means compared against a single fixed value are evaluated using a one-sample t-test. Reported values in the text correspond to mean and standard deviation unless otherwise indicated ($*p < 0.05, **p < 0.01, ***p < 0.001$, ns = non-significant).

3.4 Extended Data

3.4.1 Extended Data Figure

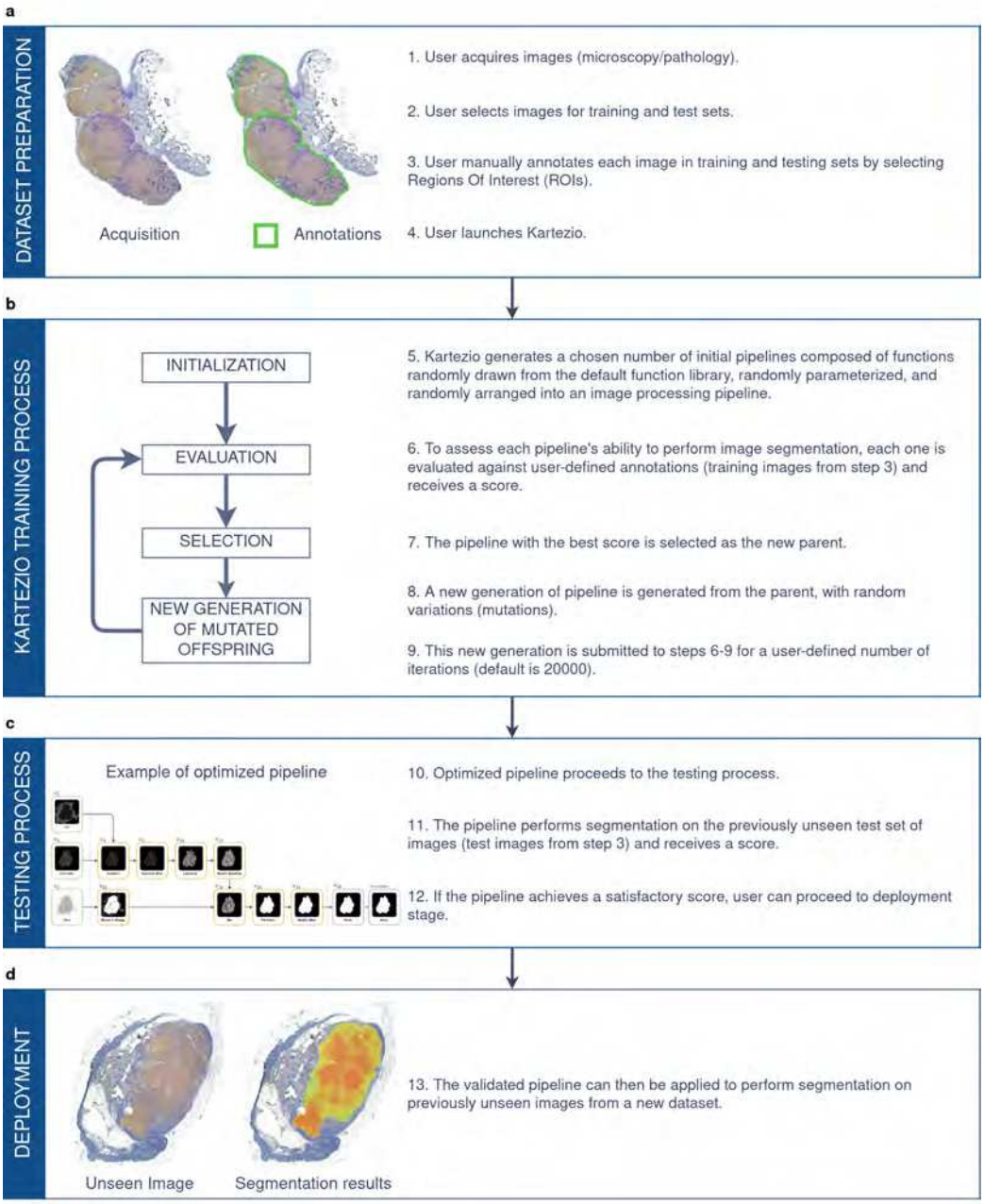


Figure 3.8: **Extended Data Figure 1 | Conceptual summary of Kartezio**
(a) Dataset preparation by experimentalist; **(b)** illustration of Kartezio training; **(c)** illustration of Kartezio testing; **(d)** deployment of Kartezio-generated pipeline.

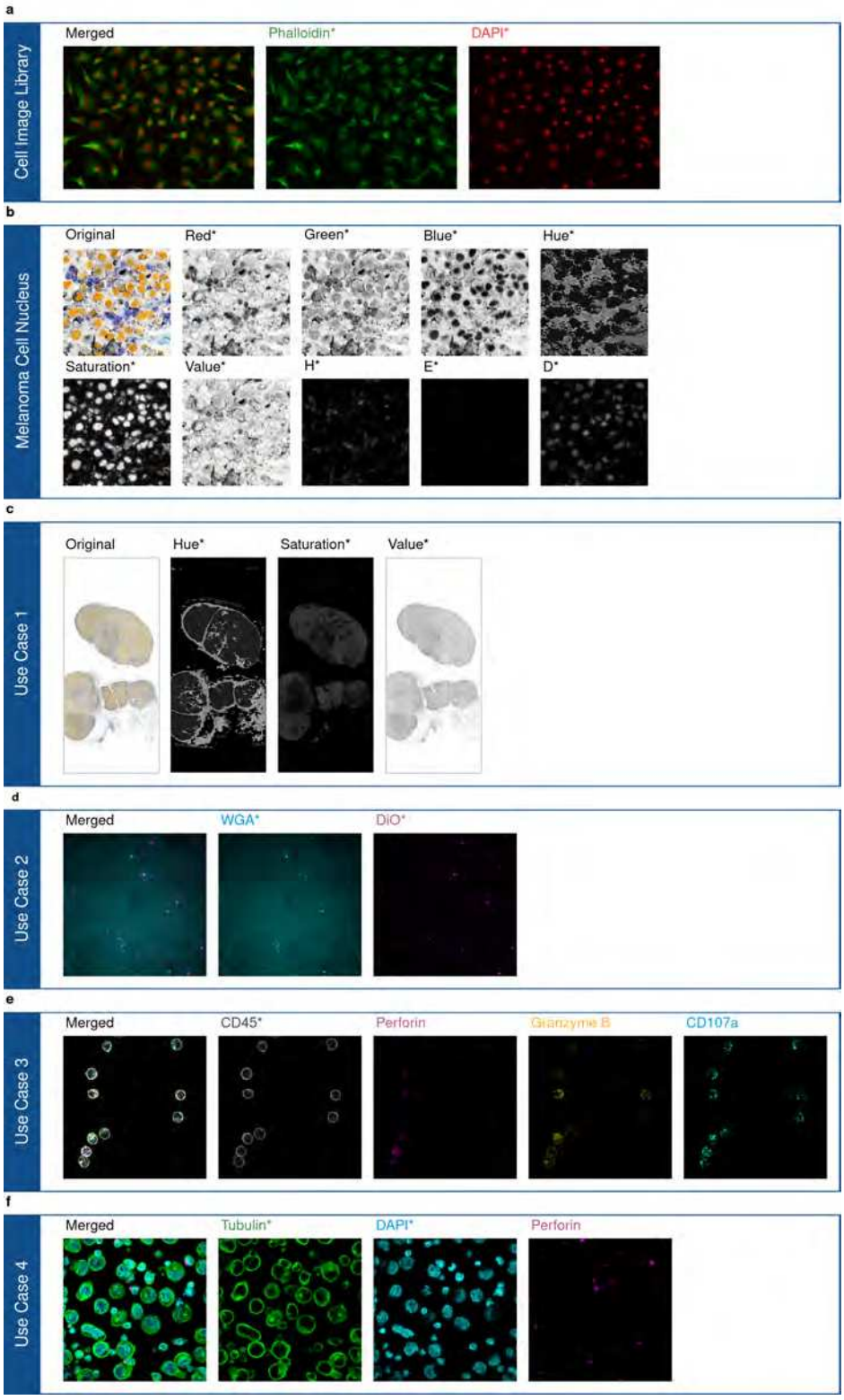

a
Cell Image Library
Merged
Phalloidin*
DAPI*
b
Melanoma Cell Nucleus
Original
Red*
Green*
Blue*
Hue*
Saturation*
Value*
H*
E*
D*
c
Use Case 1
Original
Hue*
Saturation*
Value*
d
Use Case 2
Merged
WGA*
DiO*
e
Use Case 3
Merged
CD45*
Perforin
Granzyme B
CD107a
f
Use Case 4
Merged
Tubulin*
DAPI*
Perforin

Figure 3.9: **Extended Data Figure 2 | Illustrative images from each Use Case**
(a-f) Representative images from each Use Case. Total numbers of images in the training and test datasets for each Use Case are summarized in **Extended Data Table 3**; numbers of in silico experiments performed for each Use Case (i.e. number of "runs") are shown in **Extended Data Table 4**. **(a)** Images from the previously published cell image library (described in **Fig. 2**) wherein in vitro neurons were stained with phalloidin (green) and DAPI (red) **(b, c)** Images from the melanoma cohort [2] comprised of fixed tissue slices stained for Sox10 (orange), CD8 (purple), CD107a (black) and counterstained with hematoxylin (blue). **(b)** Original high resolution images (evaluated in **Fig. 2**) were split into RGB, HSV or HED color spaces as shown. **(c)** Original low resolution images (described in **Fig. 3**) were split into HSV color space as shown. **(d)** Extracellular particles derived from polyclonal CTLs (shown in **Fig. 4**) were stained with WGA (cyan) and DiO (magenta) and imaged using TIRFM. **(e)** Human polyclonal CD8+ CTLs (shown in **Fig. 5**) were stained with antibodies directed against CD45 (gray), perforin (magenta), granzyme B (yellow) and CD107a (cyan). **(f)** A mixed population of cells containing both human clonal CTLs and target cells (**Fig. 6**) was stained with DAPI (cyan) and antibodies directed against α-tubulin (green) and perforin (magenta). * indicates channel was used as a model input for the corresponding Use Case.

a

```python
class ModelMelanoma(CodeModel):
    def __init__(self):
        super().__init__(endpoint=EndpointWatershed())


    def _parse(self, x):
        x_2 = x[1]
        x_3 = x[2]
        node_4 = self.call_node("bitwise_and", [x_3, x_2], [159, 219])
        node_5 = self.call_node("open", [node_4], [57, 55])
        node_6 = self.call_node("inrange", [node_5], [139, 253])
        node_7 = self.call_node("threshold", [node_5], [88, 119])
        z_1 = node_7
        z_2 = node_6
        Z = [z_1, z_2]
        return Z
```

Figure 3.10: **Extended Data Figure 3 | Automatically generated Python classes by Kartezio from given genotype**
(a) The Python class automatically generated from the evolved genotype illustrated in **Fig. 1** and utilized in **Fig. 2d**. Source code image generated using the *Carbon* tool available at carbon.now.sh.

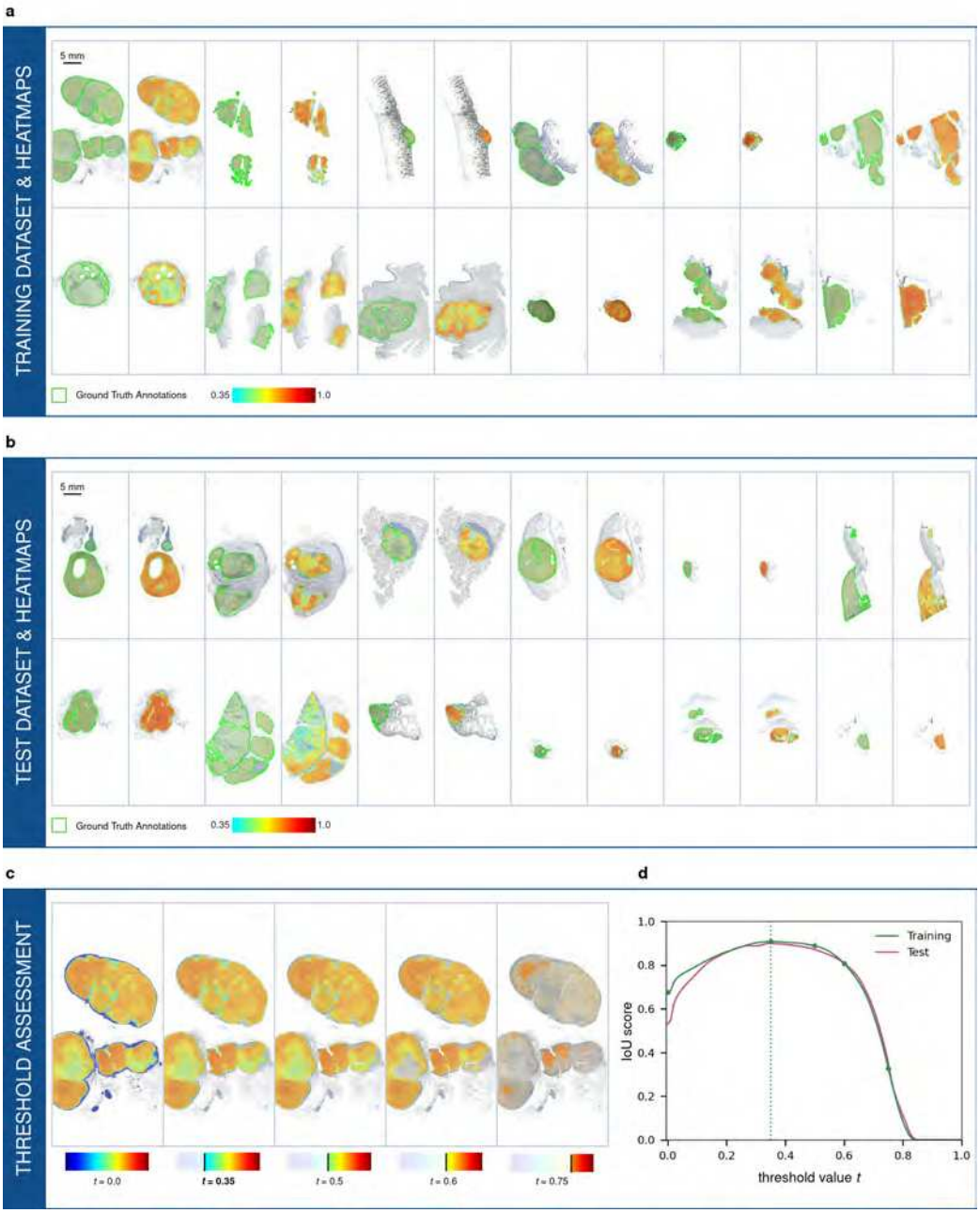

a
TRAINING DATASET & HEATMAPS
5 mm
Ground Truth Annotations
0.35
1.0
b
TEST DATASET & HEATMAPS
5 mm
Ground Truth Annotations
0.35
1.0
c
THRESHOLD ASSESSMENT
t = 0.0
t = 0.35
t = 0.5
t = 0.6
t = 0.75
d
Training
Test
IoU score
threshold value t
0.0
0.2
0.4
0.6
0.8
1.0

Figure 3.11: **Extended Data Figure 4 | Tumor nodule datasets and heatmap threshold analysis**
Tumor nodule datasets comprised of fixed tissue slices stained for Sox10 (orange), CD8 (purple), CD107a (black) and counterstained with hematoxylin (blue) were used for training **(a)** and testing **(b)** of Kartezio-generated semantic segmentation pipelines (n=100 models). Left panels in each pair depict tumor annotations as delineated by an expert pathologist (green lines). Right panels depict the predictions of 100 Kartezio-generated models whose outputs have been normalized and averaged to generate probability-based heatmaps of predicted tumor contours (minimum probability depicted is 0.35; maximum is 1.0). **(c)** A representative histology image with heatmap overlay and threshold set to different values (t = 0, 0.35, 0.5, 0.6, and 0.7) **(d)** Fitness (IoU score) of the model ensemble containing 100 Kartezio-generated pipelines as a function of threshold value (with increments of 0.01) for the training dataset (12 histopathology images, green) and testing dataset (12 histopathology images, magenta). Dotted green line indicates the optimal threshold value (t = 0.35) as determined using the training dataset.

3.4.2 Extended Data Table

3.4.3 Extended Data Movie

Extended Data Movie 1 | An illustrative example of an instance segmentation pipeline iteratively evolving using Kartezio

This is an illustrative example of an instance segmentation pipeline evolving using Kartezio, according to a $1 + \lambda$ evolution strategy wherein $\lambda = 2$. Two training images (denoted *Original Image* A and *Original Image* B) and their corresponding manual annotations (*Annotations A* and *Annotations B*) were selected from the original Cellpose specialist dataset [39] and provided to Kartezio, which generated an initial parent pipeline composed of functions randomly drawn from the default function library, randomly parame-

Function	Index	Symbol	Arity	Parameters	Sources	Domain
max	0	MAX	2	0	OpenCV	u8
min	1	MIN	2	0	OpenCV	u8
mean	2	MEAN	2	0	OpenCV	u8
add	3	ADD	2	0	OpenCV	u8
subtract	4	SUB	2	0	OpenCV	u8
bitwise_not	5	NOT	1	0	OpenCV	u8
bitwise_or	6	BOR	2	0	OpenCV	u8
bitwise_and	7	BAND	2	0	OpenCV	u8
bitwise_mask	8	ANDM	2	0	OpenCV	u8
bitwise_xor	9	BXOR	2	0	OpenCV	u8
sqrt	10	SQRT	1	0	OpenCV	f32
pow2	11	POW	1	0	OpenCV	f32
exp	12	EXP	1	0	OpenCV	f32
log	13	LOG	1	0	Numpy	u8
median_blur	14	BLRM	1	1	OpenCV	u8
gaussian_blur	15	BLRG	1	1	OpenCV	u8
laplacian	16	LPLC	1	0	OpenCV	f64
sobel	17	SOBL	1	2	OpenCV	f64
robert_cross	18	RBRT	1	1	OpenCV	f64
canny	19	CANY	1	2	OpenCV	u8
sharpen	20	SHRP	1	0	OpenCV	u8
gabor	21	GABR	1	2	OpenCV	u8
abs_diff	22	ABSD	1	2	OpenCV	u8
abs_diff2	23	ABS2	2	0	OpenCV	u8
fluo_tophat	24	FLUO	1	2	Handmade	f32
rel_diff	25	RELD	1	1	Handmade	f32
erode	26	EROD	1	2	OpenCV	u8
dilate	27	DILT	1	2	OpenCV	u8
open	28	OPEN	1	2	OpenCV	u8
close	29	CLSE	1	2	OpenCV	u8
morph_gradient	30	MGRD	1	2	OpenCV	u8
morph_tophat	31	MTHT	1	2	OpenCV	u8
morph_blackhat	32	MBHT	1	2	OpenCV	u8
fill_holes	33	FILL	1	0	Handmade	u8
rm_small_objects	34	RMSO	1	1	Skimage	bool
rm_small_holes	35	RMSH	1	1	Skimage	bool
threshold	36	TRH	1	2	OpenCV	u8
threshold_at_1	37	TRH1	1	1	OpenCV	u8
dt	38	DTRF	1	1	OpenCV	u8
dt_and_thresh	39	DTTR	1	2	OpenCV	u8
inrange_bin	40	BRNG	1	2	OpenCV	u8
inrange	41	RNG	1	2	OpenCV	u8

Table 3.3: **Extended Data Table 1 | Functions included in the default library $\mathcal{L}$ accessible by Kartezio**

Training Images	Median Training ROIs	Training Fitness	Test Images	Test ROIs	Test Fitness
89	4701	$\mathbf{0.853 \pm 0.016}$	11	521	$\mathbf{0.858 \pm 0.020}$
80	4168	$\mathbf{0.852 \pm 0.029}$	11	521	$\mathbf{0.853 \pm 0.028}$
70	3511	$\mathbf{0.851 \pm 0.018}$	11	521	$\mathbf{0.852 \pm 0.021}$
60	3180	$\mathbf{0.858 \pm 0.016}$	11	521	$\mathbf{0.856 \pm 0.019}$
50	2629	$\mathbf{0.857 \pm 0.015}$	11	521	$\mathbf{0.849 \pm 0.026}$
40	2184	$\mathbf{0.857 \pm 0.018}$	11	521	$\mathbf{0.850 \pm 0.021}$
30	1696	$\mathbf{0.861 \pm 0.023}$	11	521	$\mathbf{0.839 \pm 0.027}$
25	1306	$\mathbf{0.860 \pm 0.021}$	11	521	$\mathbf{0.840 \pm 0.025}$
20	1022	$\mathbf{0.855 \pm 0.020}$	11	521	$\mathbf{0.843 \pm 0.023}$
15	771	$\mathbf{0.867 \pm 0.028}$	11	521	$\mathbf{0.844 \pm 0.025}$
14	698	$\mathbf{0.864 \pm 0.024}$	11	521	$\mathbf{0.837 \pm 0.032}$
13	713	$\mathbf{0.873 \pm 0.028}$	11	521	$\mathbf{0.832 \pm 0.031}$
12	632	$\mathbf{0.876 \pm 0.028}$	11	521	$\mathbf{0.828 \pm 0.023}$
11	583	$\mathbf{0.865 \pm 0.031}$	11	521	$\mathbf{0.828 \pm 0.036}$
10	510	$\mathbf{0.869 \pm 0.030}$	11	521	$\mathbf{0.825 \pm 0.032}$
9	451	$\mathbf{0.872 \pm 0.031}$	11	521	$\mathbf{0.830 \pm 0.033}$
8	411	$\mathbf{0.888 \pm 0.036}$	11	521	$\mathbf{0.828 \pm 0.029}$
7	382	$\mathbf{0.880 \pm 0.042}$	11	521	$\mathbf{0.807 \pm 0.051}$
6	351	$\mathbf{0.882 \pm 0.031}$	11	521	$\mathbf{0.810 \pm 0.054}$
5	236	$\mathbf{0.886 \pm 0.036}$	11	521	$\mathbf{0.798 \pm 0.061}$
4	172	$\mathbf{0.890 \pm 0.035}$	11	521	$\mathbf{0.790 \pm 0.066}$
3	160	$\mathbf{0.895 \pm 0.044}$	11	521	$\mathbf{0.767 \pm 0.066}$
2	96	$\mathbf{0.918 \pm 0.044}$	11	521	$\mathbf{0.732 \pm 0.102}$
1	32	$\mathbf{0.933 \pm 0.058}$	11	521	$\mathbf{0.584 \pm 0.135}$

Table 3.4: **Extended Data Table 2 | Fitness scores achieved by Kartezio with descending numbers of training images**

	Use Case 1	Use Case 2		Use Case 3	Use Case 4
Use Case Name	Tumor Nodules	Extracellular Particles (SMAPs)		CTL Lytic Arsenal	Lytic Synapse
Imaging Type	2D IHC	2D IF (TIRF)		3D IF	3D IF
Image Processing Task	Semantic Segmentation	Instance Segmentation		Instance Segmentation	Instance Segmentation
Staining	Sox10, CD8 and CD107a Abs, hematoxylin dye	WGA and DiO probes		CD45, Perforin, Granzyme B and CD107a Abs	α-tubulin and perforin Abs, DAPI probe
Scale	1pixel = 46.8µm	1pixel = 0.107µm		1pixel = 0.264µm	1pixel = 0.264µm
Number of Datasets	1	2		1	1
Dataset Name	Tumor Nodules	WGA Particles	DiO Particles	CTL Cells	CTL/Target Cells
Training Images	12	$1 \xrightarrow{\text{split}} 4$	$1 \xrightarrow{\text{split}} 4$	5	8
Training ROIs	12	56	44	45	262
Testing Images	12	$1 \xrightarrow{\text{split}} 4$	$1 \xrightarrow{\text{split}} 4$	4	4
Testing ROIs	12	48	31	34	150
Post-Segmentation Analysis	Heatmap generation from the average of 100 models	Matching between masks using Intersection Over Union metric followed by feature extraction (MFI, particle size)		Feature extraction from instances (MFI) followed by 2D embedding and visualization using UMAP	Unsupervised machine learning classification using Gaussian Mixture Model followed by synapse detection

Table 3.5: **Extended Data Table 3 | Summary of Use Case dataset preparation and analysis**

	Dimension	Models	Runs	Name of Model	Model Inputs	Model Outputs	Aggregation	Model Endpoint
Use Case 1	2D	1	100	Nodule Model	$\iota = 3$, {Hue, Sat., Val.}	$o = 1$	-	Threshold
Use Case 2	2D	2	35	WGA Model	$\iota = 1$, {WGA}	$o = 1$	-	Local-Max Watershed
			35	DiO Model	$\iota = 1$, {DiO}	$o = 1$	-	Local-Max Watershed
Use Case 3	3D	1	35	CTL Model	$\iota = 1$, {CD45}	$o = 1$	Average	Local-Max Watershed
Use Case 4	3D	1	35	Cell Model	$\iota = 2$, {DAPI, Tubulin}	$o = 2$	Average	Marker-Controlled Watershed

Table 3.6: **Extended Data Table 4 | Summary of Use Case model parameterization**

	Training Min	Training Max	Training Average	Test Min	Test Max	Test Average
Use Case 1	**0.831**	**0.908**	$\mathbf{0.870 \pm 0.018}$	**0.637**	**0.888**	$\mathbf{0.846 \pm 0.047}$
Use Case 2 (WGA)	**0.658**	**0.872**	$\mathbf{0.805 \pm 0.037}$	**0.378**	**0.837**	$\mathbf{0.733 \pm 0.083}$
Use Case 2 (DiO)	**0.713**	**0.839**	$\mathbf{0.788 \pm 0.030}$	**0.399**	**0.847**	$\mathbf{0.762 \pm 0.090}$
Use Case 3	**0.975**	**1.000**	$\mathbf{0.977 \pm 0.007}$	**0.775**	**0.889**	$\mathbf{0.825 \pm 0.028}$
Use Case 4	**0.723**	**0.805**	$\mathbf{0.755 \pm 0.022}$	**0.576**	**0.729**	$\mathbf{0.667 \pm 0.034}$

Table 3.7: **Extended Data Table 5 | Summary of Use Case fitness scores**

terized, and randomly arranged into an image processing pipeline. This parent pipeline and two mutated offspring (*Child 1* and *Child 2*) were evaluated against the user defined annotations for Image A and B. The best pipeline was selected to proceed through the evolutionary selection process, during which the existing parent pipeline was randomly mutated either through changing the functions, the order in which they were arranged in the pipeline, or their parameters. With each iteration (generation), this process was repeated. When a new child pipeline outperformed the parent pipeline, it replaced the parent and the evolutionary process continued for a set number of generations (default = $20,000$). Of note, in generation 1, all three graphs (parent and two offspring) are randomly generated, while in each subsequent generation, the offspring are generated by mutating the parent pipeline.

***Extended Data Movie 1** has also been made available online at `https://youtu.be/r74gdzb6hdA`

3.5 Conclusion

This Chapter presents the foundation, the core of my entire thesis and work. The first few years of my thesis have seen the development of a powerful and versatile stable model. The application cases presented in this article are real, but are still only the first steps in the development and refinement of this approach.

Furthermore, since it is based on a modular architecture, it's possible to create an arsenal of extensions in a variety of fields. First of all, in the EC community, with a first exploration of *Lexicase Selection* [3] and the possibility of finding/modifying evolution operators (presented in next **Chapter 4**). It is also possible to adapt genome representations to fully harness CGP in the IP domain (see **Chapter 5**). On the CV side, the Preprocessing, Endpoint and function library are completely modular, allowing the integration of existing functions or the creation of handmade filters.

On the other hand, all the models generated in this study can be exploited. In **Chapter 4**, I take a closer look at genotypes to find visualization and quantification tools to study the programs produced. From these analyses, statistical models can be derived, enabling the generation of future models. The models can also be used together in a post-recombination tool to exploit biodiversity and produce new and better individuals (briefly mentioned in **Chapter 6**).

Chapter 4

Genotype Introspection

To extend the article proposed in **Chapter 3**, this Chapter looks in greater depth at two lines of research based on the Kartezio model for IP filter creation. For this study, I chose to use training data from the Cell Image Library dataset (see **Figure 3.9 - Extended Data Figure 2**). The dataset is made up of RGB images, where two channels are used (phalloidin and DAPI) and the Endpoint chosen is the MCW (2 outputs required), due to the presence of nuclei that can be used as markers. **Figure 4.1** illustrates one of the generated programs and the way it is a "white-box" model. In addition, the evolution curves of the models are shown in **Figure 4.2**, confirming the stability and speed of convergence of all models. Of the 35 models, only one performs slightly less well than the others. The rapid evolution of the curve at the start of training already gives an idea of the expected result and allows to experiment different settings before running official and longer training. The two following sections are focusing on two important aspects regarding genotypes. First, as with any ML model, a study of the hyperparameters is

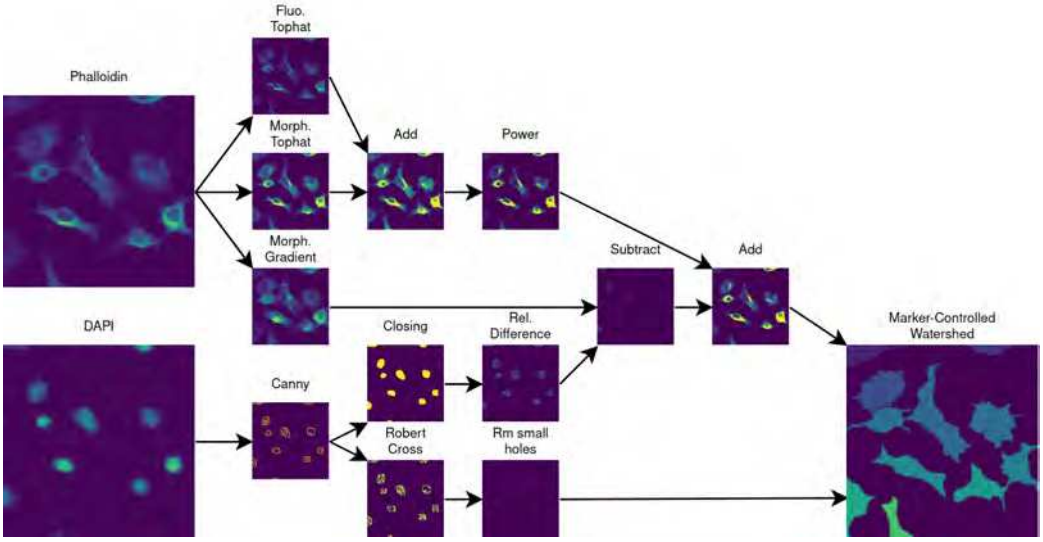


Figure 4.1: **Visualization of Kartezio-generated graph**
Visualization of one program (as a DAG) generated after 20,000 iterations, showing the transparency of the models by depicting the effect of each node, from inputs to IS output.

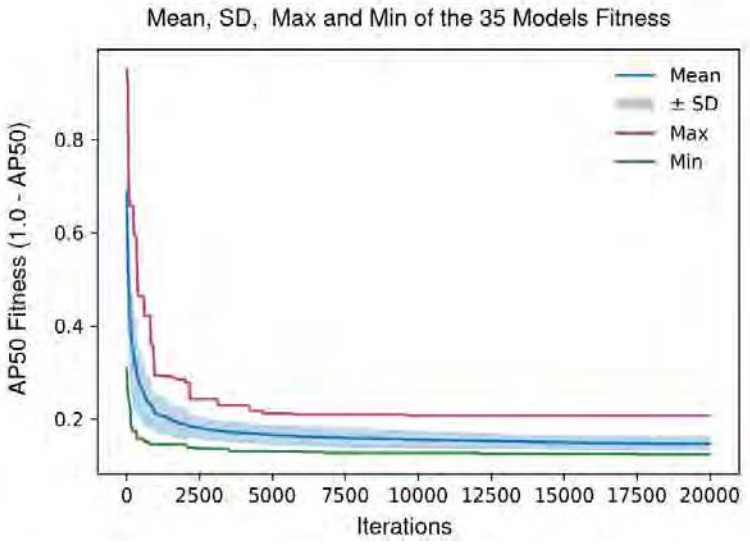


Figure 4.2: **Fitness curve along Evolution**
Visualization of the average (±SD) fitness curves ($n = 35$) over 20,000 iterations on Cell Image Library dataset (89 images). Max and Min values for a given iteration are also reported.

necessary. Secondly, I take the time to extract data on the composition of programs, focusing on the functions used, their frequency, their position in the genome, their combinations, etc.

4.1 Hyperparameters Study

Four main Hyperparameters are studied on a sub-dataset of 8 images in order to drastically reduce calculation time, as a total of 756 parameter combinations, detailed in **Table 4.1**. The number of children λ, the number of nodes η, the mutation rate μ and the output mutation rate ν are assessed. Each combination of parameters is executed 5 times, through 5 different samples of 8 images. Since we're studying the λ parameter, a total number of genotype evaluations is set at $100,000$ to fit with the article values ($\lambda = 5$ and $n_{iter} = 20,000$), thus determining the total number of iterations. **Table 4.3** reports global results on 3 different metrics: the performances on the test set (11 images), the average time to process one image, the graph length (number of activated nodes).

4.1.1 Observations and Perspectives

Overall, from **Figures 4.3 and 4.5**, the λ and η parameters do not seem to have a strong impact on the results obtained. However, there is a gap between $\eta = 5$ and the other values. For this dataset of 8 images, we need to leave room for a minimum of functions, i.e. $\eta \geq 10$. On the other hand, mutation rates, as expected, should remain fairly low, with a clearly visible gradient that increases as mutation rates decrease.

Parameters	Symbol	Assessed Values	Cumul. Runs
Children	λ	$[1, 2, 3, 5, 7, 10, 20]$	7
Nodes	η	$[5, 10, 20, 30, 50, 100]$	42
Mut. rate	μ	$[0.1, 0.15, 0.2, 0.3, 0.4, 0.5]$	252
Output Mut. Rate	ν	$[\mu + 0.0, \mu + 0.05, \mu + 0.10]$	**756**
n seeds	-	5	3780

Table 4.1: **Studied Hyperparameters**
Tested values and the cumulative number of combinations (756 in total), very combination is assessed by 5 runs.

In terms of graph size, from **Figures 4.3 and 4.6**, the larger η is, the larger the active graph. Additionally, the lower the mutation rates, the longer the graph generated. If we relate graph size to performance, we obtain the graph in the middle of **Figure 4.4**, which shows a slight trend: a longer graph performs better.

In perspective, this motivates to explore even smaller values for mutation rates and, above all, to integrate a variable similar to DL's *Learning Rate Decay* in order to decrease these rates over the course of the evolution. This would result in a very low rate at the end, which may enable to refine graphs that are already well constructed.

4.2 Function Distribution

Because of the syntactic nature of the graphs generated, it is possible to analyze the programs with simple tools. The aim of this section is to explore analysis tools to improve genotype representation, or to develop models for

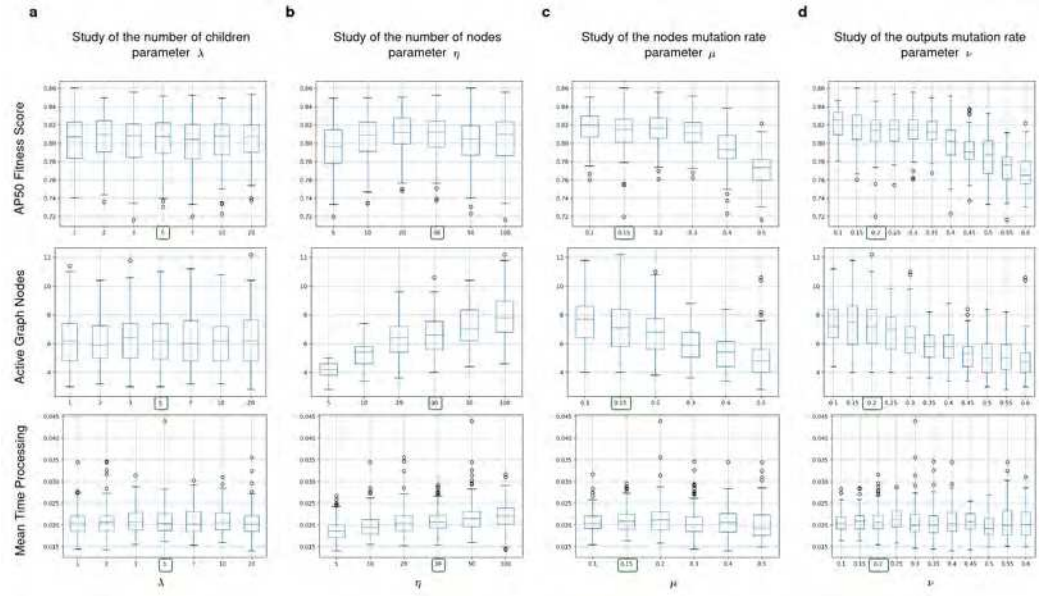


Figure 4.3: **Impact of Four Common Hyperparameters on Kartezio** The Cellpose dataset was utilized to test the impact of **(a)** the number of children λ, **(b)** the number of nodes η, **(c)** the mutation probability for nodes μ and **(d)** the mutation probability for outputs ν across a range of values, for a total of 756 different hyperparameter combinations. For each set of hyperparameter values, Kartezio was trained on eight images drawn at random from the Cellpose training dataset, with five runs performed per condition. The five pipelines thus generated were tested on the Cellpose 11-image test dataset as previously described. AP50 fitness score (top), number of active graph nodes (middle) and mean processing time (bottom) were quantified for each hyperparameter value. For each value of a given hyperparameter on the x-axis, the associated mean and standard deviation represent aggregate data from all other hyperparameters. Values inscribed in green squares represent the hyperparameter values used in the accepted article.

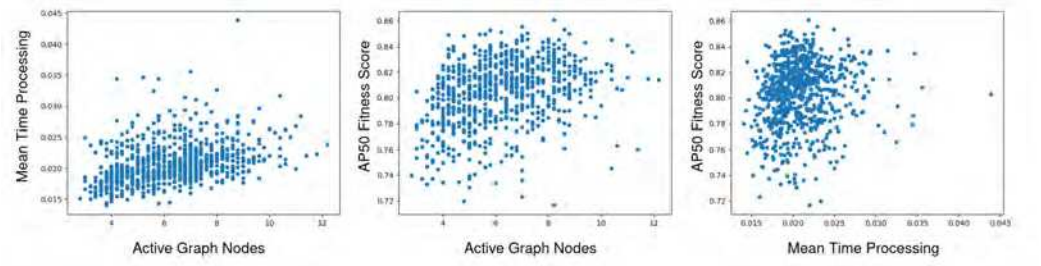


Figure 4.4: **Tendency of one metric against another**

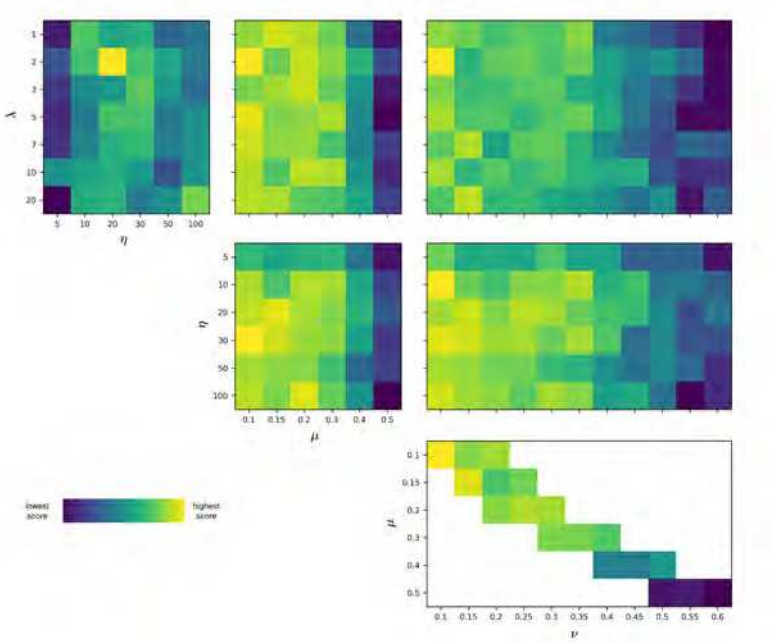


Figure 4.5: **Relative Test AP50 visualization with 2 fixed parameters**

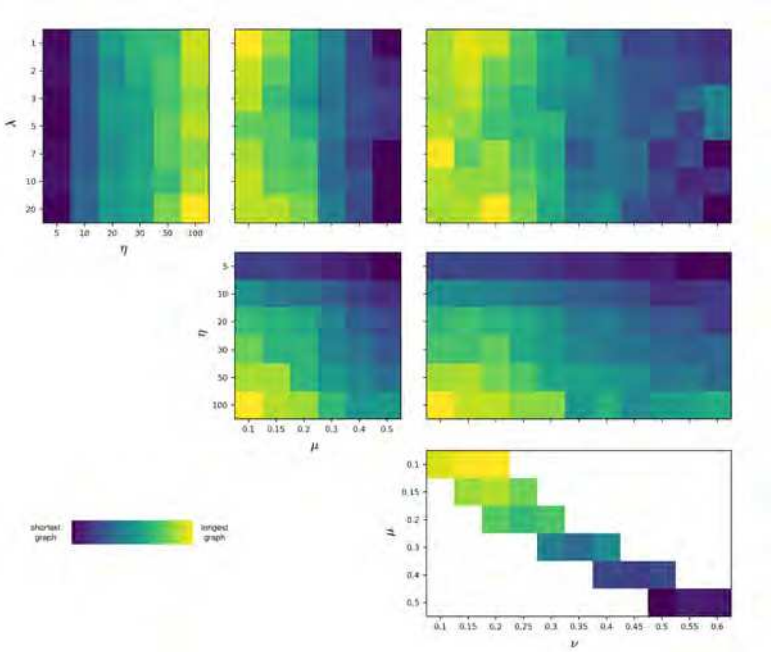


Figure 4.6: **Relative Graph Length visualization with 2 fixed parameters**

scaling learning. It could also be a preliminary step towards introducing Automatically Defined Functions (ADFs) to the library. The more models you train, the better the meta model improves. This section is based on the 35 models trained on Cell Image Library (with 89 images, see **Chapter 3**). A history of the parent genotypes, saved every 20 iterations, makes it possible to study the evolution of the functions used during training.

4.2.1 Function Histogram

After 20,000 iterations, by decoding the genotype and studying the active graph, it is possible to count each call to a function. By repeating the operation for each trained model, we obtain a histogram that can be sorted with the most frequently used functions first (see **Figure 4.7**, top left). The account information alone gives a fairly binary idea of the usefulness or otherwise of a function for processing the given set. In the case shown in **Figure 4.7**, I find the case of 2 unused functions interesting: *Sobel*, a very classic filter, is never used here. This is presumably because the filter's implementation only calculates the horizontal or vertical gradient and not the magnitude gradient, which combines the two. This is an assumption I made when implementing the functions: they are all symmetrical in XY except for *Sobel*. It's a function that's becoming too complex to integrate into the graph or that has no relevance to cell segmentation. A second function, *Gabor*, which allows you to convolve with much more complex filters, similar to those learned by DL, is never used. I suspect that it's difficult to find the right filter (and therefore the right combination of parameters) without a guided method

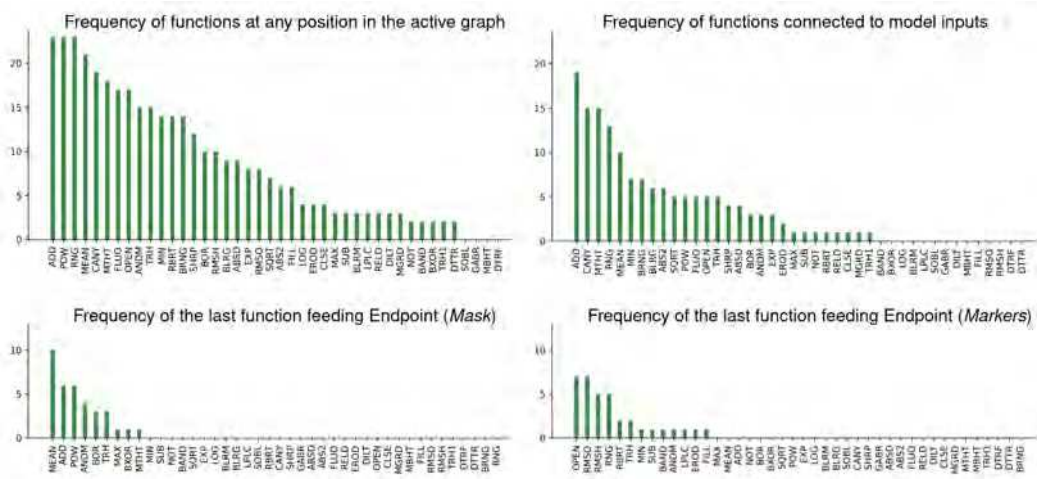


Figure 4.7: **Histograms showing function frequency among active graphs**
Four cumulative histograms build with $n = 35$ models trained on Cell Image Library.

such as DL backpropagation. Histograms can be restricted to the study of a specific position in the graph, such as functions that take at least one input channel as argument (top right), or those that are directly connected to outputs (bottom panels). In the case illustrated here, it is satisfying to see the relevance of the functions used for the outputs, in particular: functions acting on the signal (*Mean*, *Add*, *Power*) for the *Mask* heuristic and signal erosion or debris/holes removal functions (*Open*, *Remove Small Objects*, *Remove Small Holes*, *Range*, *Threshold*) for the *Markers*.

We can also study the evolution of this histogram over the course of 20,000 generations (**Figure 4.8**): with a chaotic distribution at the 20th iteration and a histogram with a profile similar to that at the end, at 10,000 iterations. Note that functions based on 2 inputs (*Add* and *Mean* from the top 4) take more iterations to integrate into the graph, certainly because adding al-

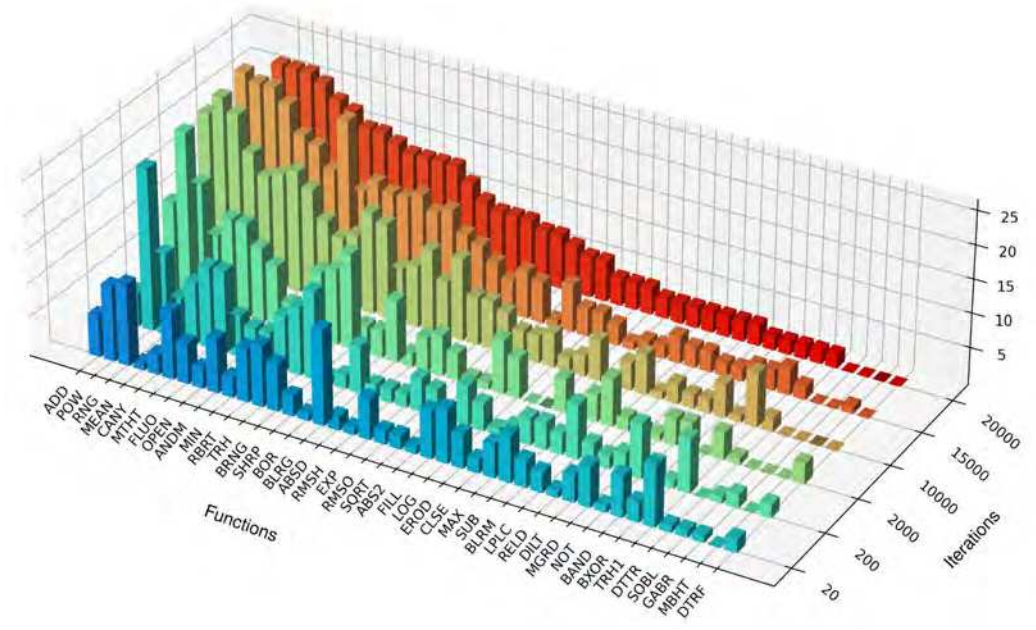


Figure 4.8: **Function histogram evolution over iterations**

gebraic complexity without already-tuned functions can be difficult at first. By combining this information, we can, for example, integrate the *Canny* function in preprocessing and add an input channel to the model. Similarly, it is possible to adapt the Endpoint by forcing a *Mean* of 2 heuristics to form the *Mask* gradient and forcing an *Open* on the marker heuristic before proceeding with the MCW. The model's task is then simply to optimize the 3 outputs. And since the Endpoint allows great flexibility, we can even generalize the average to n heuristics for calculating the topology map. Of course, adding more outputs also adds complexity (due to additional branches) to interpretability and may slow down/restrict evolution.

4.2.2 Bi-grams

N-grams are also used to analyze syntactic data. They count the number of occurrences of a series of n elements. In our case, a 2-gram (bi-gram) would be the count of all pairs of functions. In IP, this is very important data, since filters are not assembled randomly. N-grams are often used to compose probabilistic models that predict the next element from a sample. **Figure 4.9** illustrates the pair count as an heatmap: for instance, *Roberts* received 8 times the result of *Canny*. We can also see that *Canny* is used 15 times directly on the *DAPI* channel and 0 times on the *Phalloidin* channel, which led to the idea of adding Canny(i_1) as a 3rd input. The Robert(Canny(x)) pair can also be added to the function library for direct use. An external meta model can find these patterns and assemble ADFs.

4.2.3 Function location

This last section aims to study the localization of functions in the genotype. Unfortunately, due to the original CGP algorithm, the distribution is not uniform: as connections always mutate to a predecessor, active nodes are more numerous at the beginning of the genome. This can be seen in **Figure 4.10** (left), where the genotype is divided into segments (2 segments of 15 nodes for the first and 6 segments of 5 nodes for the second). By normalizing the occurrences, we obtain a probability map that is much more interesting to interpret and more in line with reality. The higher values in *S2* than *S1* (on the 3rd panel) for *OPEN*, *POW*, *MEAN*, clearly show that these are functions that tend to appear at the end of the program (confirmed

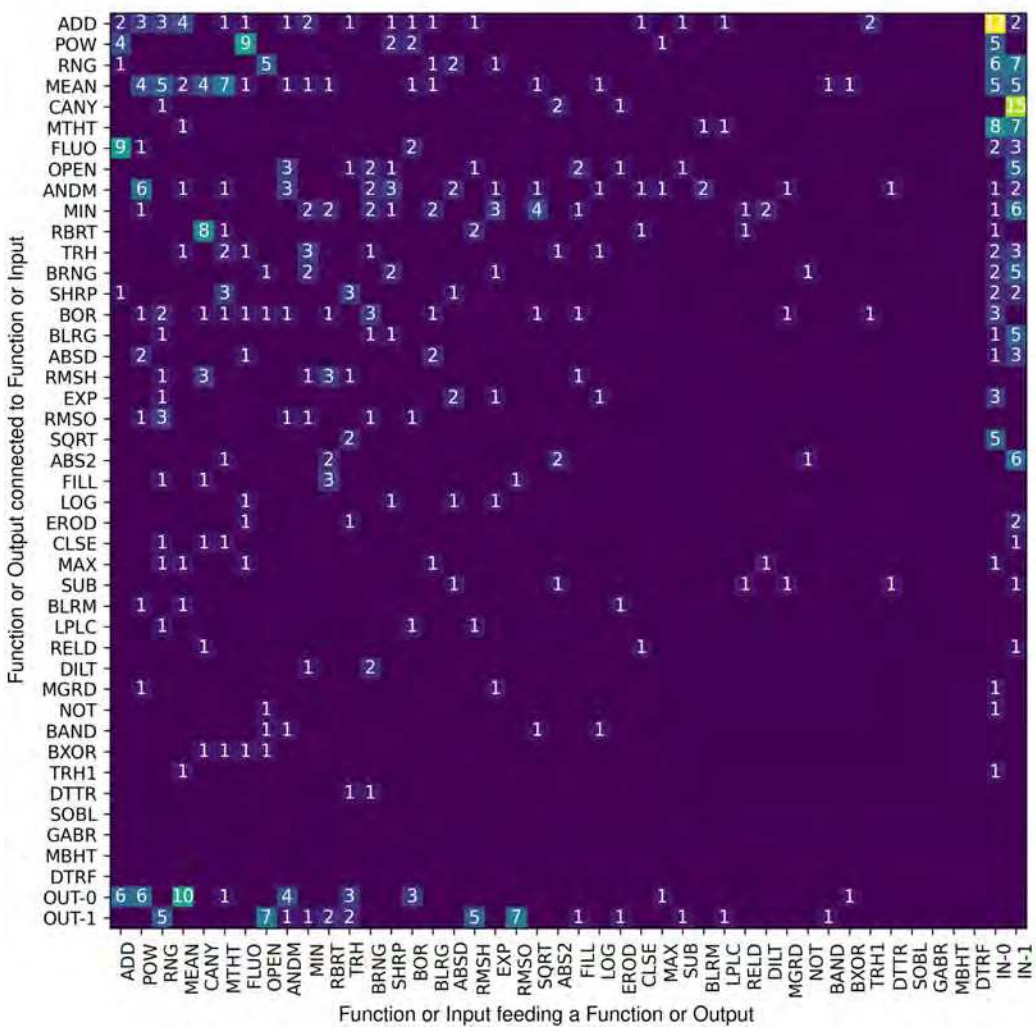


Figure 4.9: **Function bi-grams visualization**
Reading by line: the function *POW* is used 9 times using the result of *FLUO* and 5 times directly on IN-0 (Phalloidin channel). Reading by column: The output of *OPEN* function feeds 5 times *RNG* function and 7 times the second graph output OUT-1 (*Markers*).

in **Figure 4.7**) as well as they appear in the second half of the genotype. In the panel on the right, for example, we can see that among the nodes active on the 5th segment, there's a 22% chance of seeing a *MEAN* function. In my opinion, this is an important representation that would enable a more complete profile of the algorithms if the active nodes were better distributed. Having a more aligned distribution would also favor the possibility of crossovers or recombinations (as briefly presented in **Chapter 6**). This is an aspect I find promising and I'm going to investigate this distribution with a continuous parameter ω allowing a more or less strong distribution. In the same way, with a more spread-out distribution of functions, we could investigate which nodes find their final configuration fastest (rather the first, the last, the 2 extremities, the center, no distinction...), and thus try to determine the order of construction of the algorithms, always with a view to improving the scalability of future trainings.

4.3 Preliminary conclusion

To conclude this chapter, the study of hyperparameters (only 4) shows us that the main impact is given by mutation rates. These must remain low to avoid offspring decimation but it also makes the emergence of a high-performance child rare. This study therefore underlines the importance of introducing more advanced mutation tools. I'm thinking of 2 parameters **i)** the first would be a *Mutation Rate Decay* (or *Scheduler*) allowing mutation rates to decrease over the course of evolution. **ii)** the second would reduce the impact of a mutation over the course of evolution, notably by reducing the

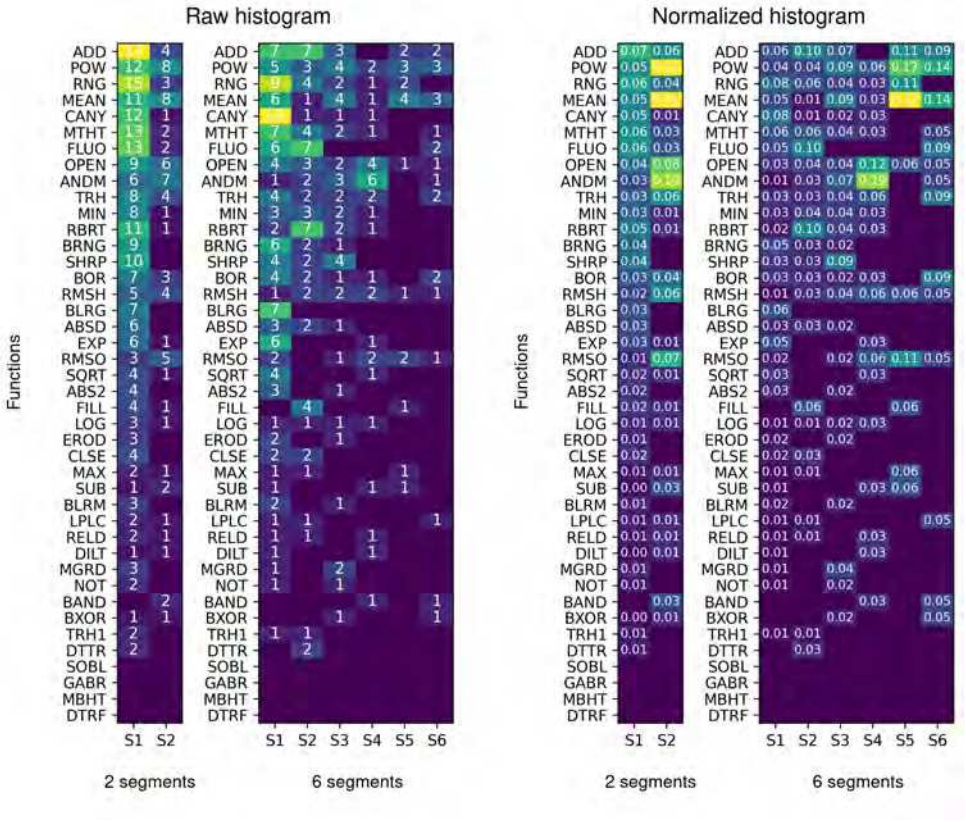


Figure 4.10: **Function position in partitioned Genotype**

range of possible values close to the current value, thus making it easier to finish fine-tuning the parameters. We could also transpose this refinement to the functions used if we manage to create a distance map between functions. This is a point I'm taking up in the discussion, particularly when encoding functions in a Transformer model. Finally, connections can also be refined using statistical data such as bi-grams, to favor connections that make sense. In fact, the study of functions could lead to the creation of probabilistic meta-models that facilitate learning through the creation of ADFs or more "intelligent" mutations. Moreover, I think that genotype partitioning is a good idea for the composition of IP algorithms and that it would be worth testing a new genotype representation with functions reserved for certain segments of the genotype. In the same way, balancing function positions within the genotype can be beneficial for the creation of meta-models.

Finally, unlike DL, hyperparameters are not vital to the convergence of learning, and more importantly, user do not have to worry about finding an architecture that works to get a result. CGP does it all, and that is a huge advantage in terms of accessibility to the general public.

Chapter 5

Enhancing Kartezio: Multimodal Adaptive Graph Evolution

After demonstrating the power of a tool like Kartezio and CGP models, we must investigate beyond the foundational principles and explore groundbreaking extensions. This Chapter explores a major conceptual advancement that, while still in its early stage, shows immense promise in redefining the landscape of CGP.

This enhancement revolves around a holistic extension of the Kartezio model. Recognizing the vast and varied spectrum of the CV domain, it becomes imperative to accommodate an array of diverse types within a single genotype. In its original version, Kartezio only works with input and output images (as masks) and the parameters of the functions are constants optimised during the course of the evolution but which it could

be interesting to transform into "variables" dependent on characteristics in the image (e.g. contrast, lightness...). The introduction of Preprocessing and the Endpoint was the first way of breaking this constraint by having nodes outside the graph that could interoperate with a variety of types. Strongly-Typed Genetic Programming (STGP) [209] already tried to bring an answer to this issue for the TGP. By finding a way to generalize the use of a large number of types within CGP, the generated programs could be even more accurate and address a larger panel of CV and IP needs.

5.1 Multimodality for Cartesian Genetic Programming

5.1.1 Introduction and Context

Building upon the discussion in **Chapter 3**, the current implementation of MT-CGP offers a compelling framework, particularly suited for multimodal challenges such as those in CV and IP. In such domains, diverse types are indispensable for the formulation of a holistic algorithm. This was already an issue raised in the realm of GP in which STGP [209] was proposed to allow the integration of diverse types into the generated trees. An agent playing games was already evolved using STGP [210].

To assert the need for multiple types, consider the convolution operation in IP. It mandates an image (essentially an Array $\mathcal{A}$) to be filtered, in conjunction with a kernel (k) to process the image. This association is high-

lighted in the subsequent function signature: $Convolution(\mathcal{A}, k) \rightarrow \mathcal{A}$. An image histogram is also an important component to discern intensity characteristics. The *Otsu Threshold* method exemplifies this by determining the threshold value predicated on the image's histogram. Here, we can represent the histogram as a vector $\mathcal{V}$, with the Otsu method producing the threshold value as a scalar $\mathcal{S}$: $Otsu(\mathcal{V}) \rightarrow \mathcal{S}$ where the histogram is obtained with $Histogram(\mathcal{A}) \rightarrow \mathcal{V}$ for instance. Geometrical entities such as Points $\mathcal{P}$ or Lines $\mathcal{L}$ are part of several CV tasks as well. Tasks such as sketching lines contingent on two points ($DrawLine(\mathcal{A}, \mathcal{L}) \rightarrow \mathcal{A}$ where the line is obtained through $Line(\mathcal{P}, \mathcal{P}) \rightarrow \mathcal{L}$) or with point paired with an angle ($Line(\mathcal{P}, \mathcal{S}) \rightarrow \mathcal{L}$), can be cited as examples. Additionally, a point can serve as the pivot for image rotation operations:$Rotate(\mathcal{A}, \mathcal{P}, \mathcal{S}) \rightarrow \mathcal{A}$. Extending the scope beyond individual points, polygons (denoted as Contours $\mathcal{C}$) enrich the computational toolbox, paving the way for operations like ConvexHull [211] ($ConvexHull(\mathcal{C}) \rightarrow \mathcal{C}$ where contours are obtained with $Contours(\mathcal{A}) \rightarrow \mathcal{C}$) or deducing the centroid of a shape($Centroid(\mathcal{C}) \rightarrow \mathcal{P}$). Many IP pipelines are constructed by integrating a diverse range of data types, which poses a significant challenge when trying to replicate them using the current MT-CGP model. Consider a select set of types: Array, Kernel, Vector, Scalar, Point, Line, and Contour, and an arity of $\alpha = 2$, each function within the library would require $7^2 = 49$ signatures. The *Kartezio* model, as discussed in **Chapter 3**, already overcomes some of these constraints by incorporating nodes capable of handling different types both upstream (Preprocessing) and downstream (Endpoint) of the CGP graph. The major objective now is to extend this versatility across the entire graph by intro-

ducing a novel genotype representation.

The conceptual framework of this research aims to empower the creation of graphs under the name of *"Multimodal Adaptative Graph Evolution"*.

This work was submitted to the GECCO 2024 conference and is the product of several contributors: I designed the concept of this model and the application for the IS and Camilo De La Torre did most of the application work, demonstrating the usefulness of this model in comparison with the existing MT-CGP. It has been made under the supervision of Prof. Sylvain Cussat-Blanc and Dr. Dennis G. Wilson.

5.1.2 Abstract

The problem of program synthesis is to automatically find a function based on some evaluation criteria, for example, how well the generated function matches known input, output pairs. CGP has been used, with success, for a variety of function synthesis problems, such as circuit design, pattern analysis, and game playing. However, it has mostly been applied to problems which treat a single type of data, from boolean values to entire images. While building a model of data most of the time requires to deal with various kinds of data, dealing with a variety of data types for a given problem reaches CGP limits. In 2012, MT-CGP, an extension of CGP in which the function library can process different data types has been proposed but quickly reached limitations in term of search space and complexity to build the function library. In this work, we propose MAGE: Multimodal Adaptive Graph Evolution, a generalization of CGP that extends its core representation to natively include functions of different domain types by grouping functions by type and constraining graph mutation based on type. We compare Multimodal Adaptive Graph Evolution (MAGE) to standard CGP and MT-CGP on the Program Synthesis Benchmark and an image segmentation task, showing that the representation and mutation constraints of MAGE aid in the search of multimodal functions.

5.1.3 Introduction

In modern scientific data analysis, methodologies increasingly rely on integrating and interpreting diverse data types, spanning from numerical integers

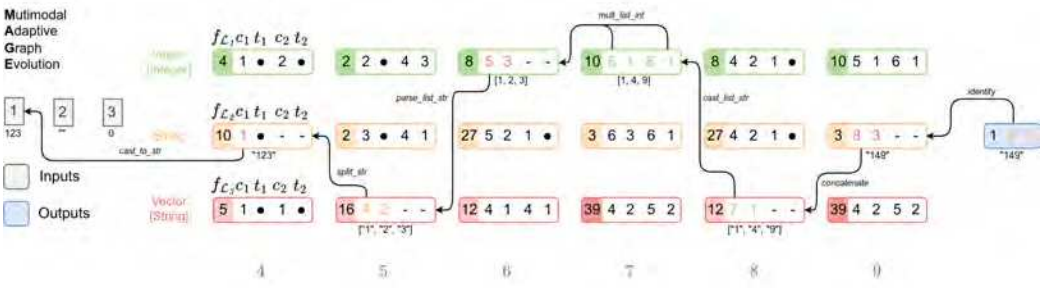


Figure 5.1: Multimodal Adaptive Graph Evolution illustrative model

to textual strings, and extending to complex multi-channel images. While human-designed computer programs can manage this variety of data, incorporating multimodality into ML algorithms poses a more challenging task. GP, which seeks to automatically generate computer programs, emerges as a potential solution for addressing multimodality.

GP has been applied to program synthesis across a spectrum of applications, including medical image analysis [1], game playing [5], cryptography [212], and machine learning [213]. GP algorithms evolve populations of individuals that encode a representation of computer programs, retaining the best-performing individuals at each iteration to minimize a fitness function. Given GP's inherent ability to generate computer programs, its utilization and advancement in program synthesis applications are natural progressions. However, the majority of GP approaches for program synthesis focus on single-type problems, where inputs, operations, and outputs involved in the GP evolution all utilize a single type of data, thereby limiting the applicability of such algorithms to a broader range of problems. PushGP [128, 129] and MT-CGP [141] have been employed to address multimodal problems. Both approaches have demonstrated promising results across various tasks,

spanning from computer vision to control. PushGP is a stack-based GP algorithm that inherently accommodates multimodal data by pushing results from an instruction to its corresponding stack, which depends on the type of the result. Similarly, the inputs for an instruction are extracted from their respective stacks based on the function's signature to execute. On the other hand, MT-CGP extends CGP by intelligently expanding the library of functions available for evolution, thereby facilitating the utilization and return of different types while maintaining the same graph representation.

We introduce Multimodal Adaptive Graph Evolution (MAGE), an extension of MT-CGP, which seamlessly integrates multiple function libraries, each specialized for a distinct return type. In contrast to MT-CGP, MAGE adopts a multi-chromosome representation, such as a Cartesian grid, where function libraries are divided and organized, assigning each genome one of these libraries. This restructuring yields two primary advantages. Firstly, it reduces genome size, thereby enhancing optimization. MT-CGP has demonstrated in practice the necessity for lengthy genomes to accommodate the correct combination of functions. However, as the number of data types increases, longer genomes can impede optimization or hinder computational efficiency. Secondly, the division of function libraries, tailored to specific return types, decreases the search space by leveraging function signatures to generate useful individuals—graphs wherein all operands align with the expected types for calling functions. This accelerated convergence and enhanced exploration of the search space enable MAGE to excel in scenarios where MT-CGP struggles to reach an optimal solution. Additionally, the framework's structured approach enhances adaptability for problems involving new data types, as

the integration of a new function library can be achieved without the need for extensive re-engineering of existing libraries.

In this paper, we first introduce the main concepts of GP and Program Synthesis (PS), with a particular focus on CGP methods. Following this, we provide a detailed description of the MAGE algorithm. Subsequently, we evaluate MAGE against MT-CGP across various problems included in the Second Program Synthesis Benchmark Suite (PSB2) [214], demonstrating that MAGE consistently converges faster and achieves superior optima compared to MT-CGP when utilizing a shared set of functions. Furthermore, we apply the MAGE framework to a medical image segmentation task, showcasing its practicality and efficacy in real-world scenarios. Finally, we conclude by outlining future research directions and offering some closing remarks.

5.1.4 Background

GP entails the process of discovering the most efficient computer programs for a given task by evolving the ordering and parameters of mathematical functions [123]. This optimization method typically involves selecting the fittest computer program candidate at each iteration. The exploration of the search space is achieved through mutations and crossover operations between individuals, mimicking the principles of natural selection.

Given its intrinsic ability to generate computer programs, GP has found widespread applications in Program Synthesis (PS) [215, 216]. In Program Synthesis, algorithms automatically discover routines, in this case, through an evolutionary approach, rather than relying on manually crafted procedural

steps to achieve the desired program. Various evolutionary methods have been employed for this purpose, including stack-based GP, such as PushGP [128], as well as grammar-based GP and LGP [215].

Program synthesis problems manifest in various forms. Within the domain of GP, Symbolic Regression (SR) problems stand out, aiming to derive a mathematical formulation that accurately models given data, typically comprising scalar inputs and outputs. In scenarios where the true mathematical formula is known, the algorithm must not only approximate a solution but also identify the correct formula. The SRBench compilation of SR problems encompasses over 200 numerical datasets, including 130 instances with known ground truth regression problems, serving as a primary benchmark for GP methods [217, 218].

While SR problems primarily involve scalar data, other program synthesis challenges necessitate interactions between multiple data types, extending beyond numerical inputs alone. The Program Synthesis Benchmark Suite presents a range of such problems, where both training data and solutions exhibit mixed-type characteristics [214, 219].

Classification, particularly multi-label classification, naturally falls into the realm of mixed-type problems if at least one of the features is non-categorical. Multi-label classification tasks have been addressed using GP, either by preprocessing the problem to achieve a uniform representation or by independently handling categorical and numerical data, merging the outcomes through a logical layer [220].

PushGP has been employed to tackle a diverse array of multimodal problems [214, 215]. PushGP, built upon the Push family of programming languages,

operates as a stack-based evolutionary framework. It inherently accommodates multimodal data by assigning a dedicated stack for each data type. Instructions within the 'exec' stack push and pop items from these typed-stacks in accordance with the instruction's signature. Upon completion of the evolutionary process, one can extract the last element(s) from the stack(s) corresponding to the desired types and designate them as the output from the individual.

MT-CGP has been utilized to tackle problems involving attributes of varying types [141]. In this framework, primitive functions are crafted to accommodate combinations of different types, such as a custom broadcast operation between a list and a scalar. While MT-CGP introduces this capability, other aspects of the evolutionary process remain identical to standard CGP. This study focuses on MT-CGP, hence, we delve into explaining CGP in the subsequent paragraphs.

CGP, initially proposed by Miller et al. [153], represents computer programs—genetic material—in a directed and acyclic graph structure. Within CGP, the hierarchical composition of operations, which yields the computer program, is attained by decoding the network of functions mapping inputs to outputs.

Similar to classical GP [123], CGP addresses the complexity required by specific problems by developing an extensive library of sophisticated functions. The selection of the function library is pivotal, often tailored to the specifics of the problem at hand. For instance, computer vision applications typically necessitate functions capable of image processing and feature extraction [1, 5, 143].

In CGP, a pool of candidates is typically generated at each generation, following the *1+λ* Evolutionary Strategy (ES) [153]. This process entails evaluating candidates using a defined fitness function and employing an elitist selection procedure to identify the fittest individual. The subsequent pool of candidates comprises mutated offspring of the previous fittest individual and the parent individual itself. Thus, the exploration of the search space primarily involves mutating individual genes within the genome of the elite. Notably, the utilization of the crossover operation in CGP remains an open question, and consequently, it is not commonly employed in practice, although some proposed solutions utilize a multi-chromosome genome similar to ours [153]. Traditionally, CGP has been applied to mono-type problems, where inputs and outputs are expected to be of the same type. However, many problems and supervised tasks involve different types, necessitating multimodal functions to exploit the information and connections between inputs. Moreover, even mono-type problems may benefit from the incorporation of multimodal functions, as a simpler solution may lie within this new space of possible programs.

Furthermore, the incorporation of multimodal functions enables the introduction of higher-order functions, significantly simplifying the genome. For instance, in the scenario outlined by Harding et al.[141] concerning image processing, if an optimal program requires calculating the average of an image (such as the average of all pixel values), it would be exceedingly challenging for a monotype CGP algorithm to perform this calculation based solely on scalar pixel input values. This challenge prompted Harding et al. to extend CGP to the multimodal setting, resulting in MT-CGP [141]. MT-

CGP maintains the standard CGP structure but incorporates multimodal functions alongside the existing "mono-modal" functions. Additionally, it is common in practice to manually devise rules that govern the behavior of functions based on the type combination received as input. Thus, MT-CGP facilitates the interaction of multimodal data. Moreover, MT-CGP streamlines the evolution process by eliminating the need to reimplement known higher-order functions from scratch, assuming they are required. MT-CGP has been applied to solve various problems across different domains, such as designing sound synthesizers from audio signals and scalars [221] or learning Atari game strategies from vector and scalar inputs [5].

Despite the aforementioned benefits, MT-CGP is accompanied by conceptual and practical drawbacks. One significant challenge is that all functions may need to be connected to all possible combinations of types. To address this issue, practitioners of MT-CGP have proposed three main strategies, each with its own limitations. The first strategy involves engineering functions to accommodate all types. While this approach offers flexibility, it necessitates manual adjustments for every function, particularly when dealing with problems involving new types. The second strategy entails writing functions for every combination of types and employing heuristics to dispatch them. However, this approach encounters a combinatorial problem as the number of types increases. Lastly, functions can be wrapped to return a default value in case of a mismatch between signature and input types. Unfortunately, this approach sacrifices all previous information in the genome. In the next section, we introduce Multimodal Adaptive Graph Evaluation, a generalization of MT-CGP that offers solutions to the aforementioned problems.

5.1.5 MAGE

MAGE fundamentally extends the principles of CGP. Whereas MT-CGP concentrates on integrating common types, MAGE serves as a framework where all types can contribute provided the requisite type bridges are established. As MAGE represents an abstraction of CGP, it accommodates a diverse range of types across the three primary stages of a typical CGP program: inputs, operations, and outputs.

Due to the combinatorial nature of tuples (*function - operand types*) and the corresponding remedies, MT-CGP becomes less effective as the number of types or functions increases. We contend that these remedies fail to address the primary issue with MT-CGP: the use of a single vector representation for all types, which heightens the risk of encountering type mismatches at runtime. In MAGE, we have expanded the CGP representation to accommodate a broader range of types. Consequently, a MAGE genome consists of multiple chromosomes, each representing a distinct return type, mirroring the approach of PushGP in segregating stacks by types. In other words, every output originating from a node within one of the chromosomes is guaranteed to be of the same type. This assurance stems from each vector being associated with a specific function library, ensuring its corresponding return type. With access to the return type of nodes and the functions' signatures without runtime evaluation, we can guide the mutation operation and expedite the transformation from genotype to phenotype.

Consequently, a MAGE genome comprises as many chromosomes as there are libraries. Similar to CGP, each node in the chromosome links a function

to certain inputs in the graph. However, what sets MAGE apart is that inputs can explicitly originate from other types (i.e., other chromosomes), thereby fostering composition, interaction, and dynamic parameterization of functions. This seamless interconnection capability empowers MAGE to integrate multiple types at the input, program, and output levels.

Formally, a MAGE genome $\mathcal{G}$ is composed of multiple Chromosomes $\mathcal{C}_i$, each ensuring a single return type, hence associated with a single function library $\mathcal{L}_i$. A Chromosome $\mathcal{C}_i$ is a vector composed of nodes $\mathcal{N}^*_{i,j}$ where i and j denote the horizontal and vertical position, respectively. We distinguish between 3 kinds of nodes: Input nodes $\mathcal{N}^I_{i,j}$, Program nodes $\mathcal{N}^P_{i,j}$, and Output nodes $\mathcal{N}^O_{i,j}$. All nodes are vectors composed of the smallest value holders, which we denote as the Allele. An Allele $a_{(i,j,k)}$ picks a number in a certain range $\mathcal{R}_{(i,j,k)}$. For alleles that represent functions, $a^f_{(i,j,k)}$, we have : $\mathcal{R}_{(i,j,k)} = \mathcal{R}_i = \{0, ..., |\mathcal{L}_i|\}$. In order to maintain the acyclic properties, for alleles that represent a connection to another node, $a^c_{(i,j,k)}$, we have $\mathcal{R}_{(i,j,k)} = \mathcal{R}_j = \{0, ..., j-1\}$. So far, this representation is fairly standard in classical CGP. Then, each connection is accompanied by an index specifying its type, for that type of allele, $a^t_{(i,j,k)}$, we have $\mathcal{R}_{(i,j,k)} = \{0, ..., |\{\mathcal{C}_0, ..., \mathcal{C}_n\}|\}$, i.e, from 0 to the number of types in the MAGE Genome. Quite simply, Input Nodes just hold the input values for the program and it's type is analyzed at runtime $\mathcal{N}^I_{i,j} = \{\mathbf{x}\}$. Gathering all together, for a fixed arity of 2, Program nodes are composed of one *function allele*, and two tuples (*connection allele - connection's type allele*) : $\mathcal{N}^P_{i,j} = \{a^f_{(i,j,0)}, a^c_{(i,j,1)}, a^t_{(i,j,2)}, a^c_{(i,j,3)}, a^t_{(i,j,4)}\}$. With this representation, we have all the information needed to determine whether a given node is in a correct or erroneous state based on whether the function signature (encoded

in $a^f_{(i,j)}$) accepts the operand types (encoded in $a^t_{(i,j,\cdot)}$). Therefore, the availability of the inputs node types, without the need to evaluate them, enables us to perform *safe mutations* during the evolution process. That is, when dealing with the mutation of a node $\mathcal{N}^P_{i,j}$ we can know the types of the inputs by looking at the current value of the type alleles ($a^t_{(i,j,*)}$), consequently we can check the accordance of the function encoded in $a^f_{(i,j,1)}$ and the types of the potential inputs. In case of a mismatch, we reiterate the mutation operation until a correct mutation is achieved. In doing so, we believe that we assist evolution, explore more efficiently the search space, by means of producing meaningful candidates, and thus have higher chances of finding a better solution than with MT-CGP.

In **Figure 5.1**, we present a visual example demonstrating MAGE's explicit encoding approach. The key components of MAGE are depicted, including the variety of function libraries, each linked to a distinct chromosome, and the interconnections spanning across types. We will now proceed to elucidate these concepts from left to right.

The inputs. The first three nodes represent the inputs for this particular learning problem. They are only represented on the x axis since their concrete type is resolved at runtime. This allows the use inputs for which their type is not actually one of the chromosome's types. To better understand why this is the case, in **Figure 5.1**, we depict a MAGE solution for the *Squared Digits* problem in PSB2, where the input is an positive integer (e.g. 123), accompanied by some extra inputs (i.e. "", 0), which are not used in this case. In this problem, the program has to :

- cast the input *Int* to *String* (e.g. ”123”, by using the *String* library).
- split the *String* characters (e.g. {”1”,”2”,”3”}, by using the *Vector*{*String*} library).
- parse the *Vector*{*String*} to a *Vector*{*Int*} (e.g. {1,2,3}, by using the *Vector*{*Int*} library).
- element-wise square that vector (e.g. {1,4,9}, by using the *Vector*{*Int*} library)
- cast it again, but this time to a *Vector*{*String*} (e.g. {”1”,”4”,”3”}, by using the *Vector*{*String*} library).
- finally, concatenate the *Vector*{*String*} into a single *String* (e.g. ”149” by using the *String* library). This will be the output from the program.

All the functions involved in this program use the following libraries: *String*, *Vector*{*String*} and *Vector*{*Int*}, you may notice that the *Int* library was not required even though the input used is an *Int*.

The program. The program nodes depicted in **Figure 5.1** illustrate network connections with an arity of 2, meaning each function has at most two inputs. When a directed arrow connects one node to another, it signifies that the left node serves as an input to the right node. If no arrows appear between connected nodes, it indicates that, based on the function signature, there's no need to establish a connection because the function accepts only one operand, not two. This examination of the known arity of the function

enhances genome decoding and phenotype evaluation speed by reducing the active graph to reflect only the nodes that are genuinely utilized.

Connections can extend vertically across types (denoted by $a^t_{\dots}$) and horizontally across positions (denoted by $a^c_{\dots}$). For instance, the active node in position 8 (e.g., {12,7,1,-,-}) indicates that the function with index 12, from the Vector{String} library (i.e., *cast_list_str*), is to be applied to the inputs of that node. By examining the function signature of *cast_list_str*, it was determined that the function takes a single input, thus, the last connection and type indices are not depicted. The sole input for *cast_list_str* will be the node at position 7 of type *String*. As the function and input types in node 8 align, this node is considered correct. Indeed, after each mutation, MAGE verifies the node's state and iterates until the state is correct. Specifically, when the function signature, describing the n number of operands and their allowed types, matches the first n types encoded in the node ($a^t_{\dots}$) or the runtime-discovered true type in the case of a connection to an input node (as seen in active node 4 of type *String*).

In our implementation of MAGE, we utilized the Julia programming language [222] which offers a robust type system, facilitating the authoring of generic functions that capitalize on the covariance of certain types.

The output. Output nodes function similarly to program nodes; they associate a function with inputs resolved by (i, j) position in the case of program nodes, or only by i position if they are input nodes. The "safe mutation" operator applies to output nodes since all signature and type information is also available. However, the output node depicted in **Figure 5.1** illus-

trates how we employed fixed output nodes in our experiments (for both MAGE and MT-CGP). We opted for fixed output nodes to eliminate the output node mutation rate hyper-parameter and based on recent analyses that have shown improved results in the context of MT-CGP. To implement this, we fixed the *function allele* to point to an identity function (i.e., the first function in the library by convention), the *connection allele* to the last node in the chromosome, and the *connection's type allele* to the desired type, chromosome index, for the problem.

In addition to the *safe mutation* operator introduced by MAGE, we incorporated an *active-node-material-only* mutation. With this configuration, to exit the iterative mutation loop, a node not only must be in a correct state but also the active part of the node cannot remain the same as before the mutation. In other words, when a node is mutated, it must change its active components while maintaining a correct state, thereby enforcing genuine changes in the active node material. This type of mutation becomes increasingly important as we expand the maximum arity of a node, since it is common for the majority of functions to have low arity.

In the following section, we compare MT-CGP and MAGE across a range of Program Synthesis tasks extracted from the PSB2 suite [214]. To our knowledge, this is the first attempt to utilize a CGP-derived method for program synthesis tasks. We observed that MAGE converges faster to a local optimum, significantly reducing the number of generations required to achieve a certain fitness score. While MT-CGP may reach a similar average fitness value for some problems, it does not converge as effectively as MAGE for others. Moreover, when both methods find absolute solutions, MAGE

does so more consistently, indicating better generalization to unseen data and presenting less variance between independent runs.

5.1.6 Multimodal function synthesis

We utilized the Second Program Synthesis Benchmark (PSB2) [214] to conduct a comparative analysis between our approach and MT-CGP. PSB2 comprises a meticulously curated collection of 25 program synthesis problems primarily derived from coding challenges. In the context of MAGE, PSB2 presents an intriguing benchmark due to the inherently multimodal nature of most problems, wherein input types differ from output types. As demonstrated in the PSB2 paper, supplementary functions may be introduced for each problem, even if the type is absent in the inputs or outputs (e.g., the boolean type for facilitating control flow).

We manually crafted functions that could be assembled to tackle all problems in PSB2, except for the *Bowling* problem, which we found to be excessively specific and requiring a multitude of problem-specific functions. With the knowledge that a solution exists within the realm of possible programs for each problem, we positioned MAGE and MT-CGP on equal footing and assessed the generalization potential of MAGE (i.e., how many runs were able to converge to a generalized solution). Table 5.1 illustrates the typed function libraries we designed. In contrast to PSB2, we did not develop libraries for the *Boolean* and *Char* types. Additionally, we introduced two more complex types: *Vector*{*Tuple*{*Int,Int*}} and a corresponding one with strings. The number of functions in the *Float* and *Integer* libraries is identical

because they share the same function set. The same principle applies to the *Vector*{*Tuple*{ ·, · }} libraries and the majority of functions dealing with vectors.

Our MAGE implementation for the program synthesis tasks was written in the Julia programming language [222], leveraging the multiple dispatch paradigm and the type system. This allowed us to create specialized and optimized methods by reusing generic methods, thus eliminating the need to rewrite identical functions. For each library, we employed a final *caster* function. To illustrate, let's consider the *add* function in the *Float* and *Int* libraries. This function accepts two inputs of type *Number* and returns another *Number*. When used in the *Float* library, the result of the *add* function is cast to *Float*. Conversely, when utilized in the *Int* library, the result is truncated to an *Int*, thereby ensuring the appropriate return type. Remarkably, the same function definition may be employed for all four combinations of inputs (*Int*, *Float*) without requiring any further adjustments.

As MAGE represents a generalization of both CGP and MT-CGP, all three frameworks can utilize the same function libraries. In CGP, unlike MAGE, the genome comprises a single chromosome and employs a single function library to ensure a consistent return type for all functions. Conversely, MT-CGP allows the use of multiple function libraries within the same genome, granting nodes access to a diverse range of functions of varying types. Notably, in MT-CGP, the mutation operation and decoding step do not impose any constraints. Runtime errors in nodes trigger a default (fallback) return value, contingent upon the function's library (e.g., 0 for *Int*, "" for *String*). These fallback values are also utilized in MAGE in the event of a runtime

Library	**Number of functions**
Int	22
Float	22
String	35
Vector{Int}	59
Vector{Float}	59
Vector{String}	49
Vector{Tuple{Int,Int}}	6
Vector{Tuple{String,String}}	6

Table 5.1: Function Libraries used for MAGE and MT-CGP

Parameter	**MAGE**	**MT-CGP**
Generations	20000	20000
Lambda	10	10
Nodes	30	30*types
Mutation rate	1	1
Output Mutation rate	-	-
Arity	3	3
n train samples	200	200
n test samples	2000	2000

Table 5.2: Experimental Parameters

exception, whereas in MT-CGP, they can arise from a type discrepancy.

We compared MT-CGP and MAGE across the first 10 problems in the PSB2 benchmark, excluding the *Bowling* problem, utilizing the fitness functions described in [214]. Each algorithm underwent evaluation through 20 independent runs, totaling 400 runs. Shared parameters were maintained constant for both MAGE and MT-CGP, with the list of parameters provided in Table **Table 5.2**. To minimize variability in mutation probabilities, we introduced a *real-numbered* mutation operation, which mutates precisely that number of alleles in the active graph. If the active graph contains fewer active nodes than the specified mutation parameter, we mutate all active nodes. For all experiments, a mutation parameter of 1 was used, indicating that only one allele in the active graph is mutated at each mutation step.

Notably, three operations distinguish MT-CGP from MAGE. Firstly, in MAGE, we employ a *safe mutation* operation, whereas in MT-CGP, nodes are mutated without restrictions. Additionally, mutations are performed solely on the active part of the node in MAGE, ensuring that modifications in the encoding representation result in genuine changes. Secondly, prior to the first iteration, all nodes in the initial MAGE genome are corrected (i.e., mutated until they are repaired). Thirdly, the genome size for MAGE was fixed at 30 for every problem. However, a MAGE genome of size 30 with 2 function libraries comprises a total of 60 nodes. To maintain consistency, MT-CGP is allocated 30 nodes per type to match the total number of nodes present in the MAGE configuration.

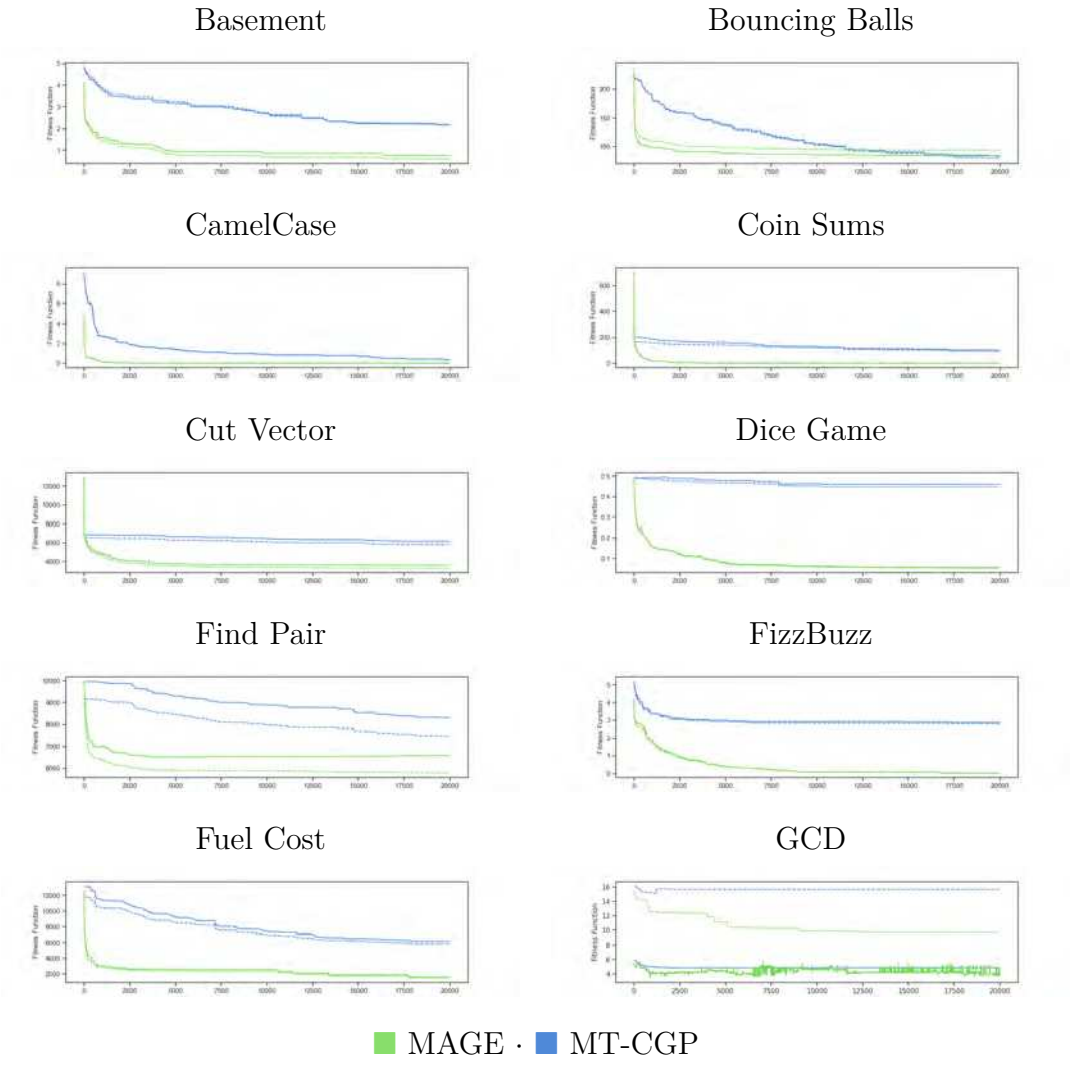


Figure 5.2: Comparison of the convergence of MAGE and MT-CGP on the PSB2 suite. Fitness is based on error (lower is better). Dashed lines indicate training data; solid lines indicate test data.

Results

In **Figure 5.2**, we compare the fitness over training, and in **Table 5.3**, we present the number of problems solved from the 10 PSB2 experiments, comparing MT-CGP and MAGE. MAGE outperforms MT-CGP on all metrics for almost all problems, with MT-CGP achieving similar error rates only in the Bouncing Balls problem. MAGE also converges much more rapidly than MT-CGP, requiring thousands fewer iterations to reach a certain score. Additionally, MAGE solves more problems than MT-CGP, and does so more frequently and consistently. In 4 out of the 6 problems where MAGE produces at least one complete solution, over 50% of the independent runs successfully solve the problem, indicating strong signs of generalization and reduced individual variance. When MT-CGP also solves a PSB2 problem at least once, it does so over a much longer period and with fewer successes.

These differences between MAGE and MT-CGP stem from the more directed and assisted search in MAGE. With the multiplication of possible combinations between inputs of different types and functions involved in the usage of multiple function libraries, the search space of MT-CGP becomes challenging to navigate. In MT-CGP, there is always a high likelihood that mutation introduces a discrepancy between function and input types, which can break or hinder its final output. Conversely, MAGE only produces individuals with consistent types, allowing evolution to proceed to relevant solutions rapidly. Furthermore, we believe that even better results can be achieved for both methods by exploring a wider range of parameter values. The fixed parameter set applied in this study may not be optimal for every problem, par-

ticularly concerning mutation rate and genome length. Additionally, the response of MAGE to different parameter values warrants further exploration. Moreover, alternative fitness functions may enhance algorithm convergence, especially for problems with vector-type outputs, where the current fitness function heavily penalizes length mismatches. Modifications such as penalizing solutions with incorrect vector lengths or adopting a different selection mechanism like lexicase selection could address these issues. Such adjustments can complement MAGE's alterations to the CGP representation and mutation process without affecting the selection mechanism.

5.1.7 Medical Image Segmentation

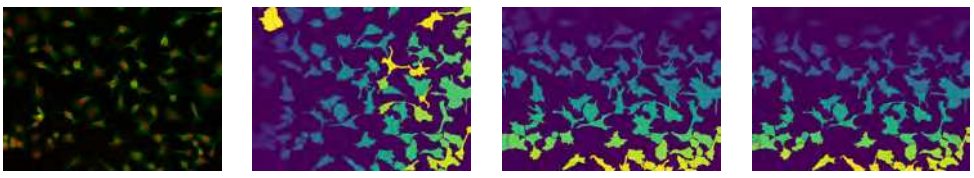
Many computer vision applications can be considered multimodal because the interaction between arrays, vectors, kernels, and scalar pixel values can be crucial for discovering simple, generalizable, and explainable pipelines. Therefore, we assess the ability of MAGE to perform image segmentation on biomedical images, as illustrated in **Figure 5.3**.

For this purpose, we integrate MAGE into the Kartezio framework [1], an image processing library based on CGP. Although this implementation is separate from that discussed in **Section 5.1.6**, the algorithmic details remain consistent. In this implementation, a minor modification involves using the function signature to determine input types, rather than encoding this information in the genome.

As Kartezio operates on CGP, its default setup employs a single function library and a single data type. In this configuration, many functions require

	Solved		Generation	
Problem	**MAGE**	**MT-CGP**	**MAGE**	**MT-CGP**
Basement	12	1	4278	18339
CamelCase	20	15	461	10514
Coin Sums	11	0	10684	-
FizzBuzz	17	0	6763	-
Fuelcost	6	0	1191	-
GCD	5	0	2217	0

Table 5.3: Solved: the number of runs which fully solved the given PSB2 problem, out of 20. Generation: the average generation at which the problem was solved.



Input Label CGP MAGE

Figure 5.3: An example of instance segmentation, with the labels provided by CGP and MAGE.

parameter values; for instance, the threshold function utilizes a parameter to establish a threshold on the input image ($Threshold(\mathcal{A}, p_1) \rightarrow \mathcal{A}$). Since these parameters are scalars, they cannot be directly used as inputs in CGP and must instead be encoded in the genotype. However, with MAGE, we can create a scalar chromosome to compute these parameters as part of the evolved program. In the threshold function example, the threshold parameter can now be set by a function: $Threshold(\mathcal{A}, \mathcal{S}) \rightarrow \mathcal{A}$. Therefore, we compare MAGE directly to CGP, where CGP employs constant, evolved parameters and MAGE potentially calculates dynamic scalars as parameters. The hyperparameters are identical: a mutation rate set to $\mu = 0.05$ for the function genes, $\nu = 0.1$ for the output genes, $\eta = 30$ nodes, and $\lambda = 5$ children.

In Kartezio, CGP utilizes an image processing function library composed of 43 functions, including image filters and transformations. The MAGE model incorporates two additional function libraries: $\mathcal{L}_a$, with 44 functions, returning arrays, and $\mathcal{L}_s$, with 17 functions, returning scalars. $\mathcal{L}_a$ is based on the original Kartezio image processing library, but it replaces most parameters with a scalar input. Additionally, the MAGE model includes two default scalar inputs: 0.0 and 1.0. Both models take two channels as inputs, Phalloidin and DAPI, and utilize a Watershed transform on the program output. This transformation requires two inputs: the markers and the topological mask, to produce the final IS prediction as described in [1].

To evaluate MAGE, we utilized the P2043 Cell Image Library dataset, generously provided by the Cellpose authors [39]. This dataset comprises 89 training images and 11 test images, as exemplified in **Figure 5.3**. To compare

the two models, we randomly selected 8 images from the training set (with the same sampling for both models) and conducted 10 independent runs. The fitness function employed is a composite of Average Precision (with a threshold of 0.5) (AP50) and Intersection over Union (IoU) to smooth the fitness landscape: $fitness = 1.0 \times \text{AP50} + 0.1 \times \text{IoU}$.

In **Table 5.4**, we present the AP50 scores achieved by MAGE on both the training and test datasets. MAGE consistently outperforms CGP across the test data, demonstrating superior generalization capabilities. As depicted in **Figure 5.3**, we showcase the labels generated by CGP and MAGE. Notably, MAGE successfully separates some closely situated cell clusters that CGP struggles to distinguish, resulting in enhanced accuracy. Additionally, in **Figure 5.4**, we illustrate the evolved graph for CGP, while in **Figure 5.5**, we display the evolved graph for MAGE. Although the utilization of scalar parameters is minimal, involving only two scalar functions, we posit that these dynamic parameters contribute to the observed improvement in generalization and the ability to discern nearby cells.

5.2 Conclusion

Multimodal problems are ubiquitous in real-world applications. While CGP has shown promise, its applicability was hindered by the challenges posed by multimodality. MT-CGP was introduced as a solution, adapting CGP for the multimodal setting, particularly in the realm of vision. While this represented progress, it soon became apparent that MT-CGP had its limitations, especially as problems grew more complex and involved multiple

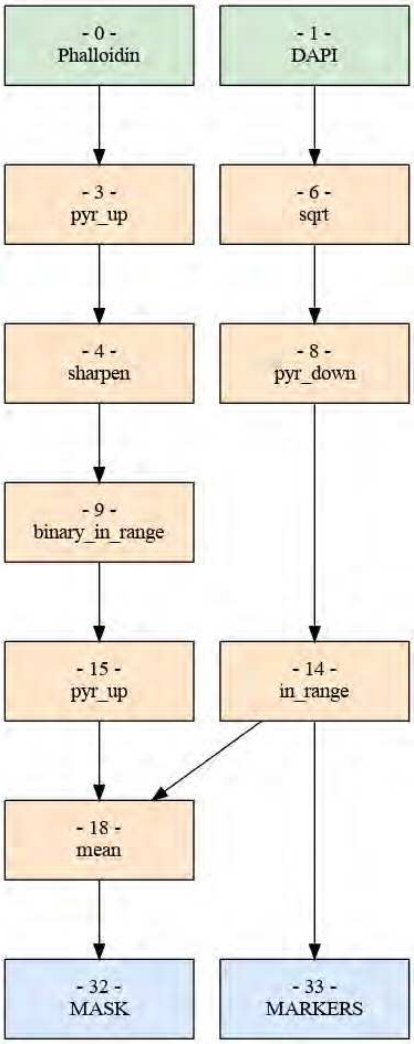


Figure 5.4: An example of evolved mono-typed CGP graph for Instance Segmentation

	Train (n=8)		Test (n=11)	
AP50	**CGP**	**MAGE**	**CGP**	**MAGE**
min	0.8440	0.8349	0.6607	**0.7766**
mean	0.8934	0.8905	0.8161	**0.8264**
max	0.9612	0.9621	0.8574	**0.8606**
SD	0.0357	0.0365	0.0547	**0.0281**

Table 5.4: Results on Cell Image Library dataset, a comparison between mono-typed CGP and MAGE model (10 runs).

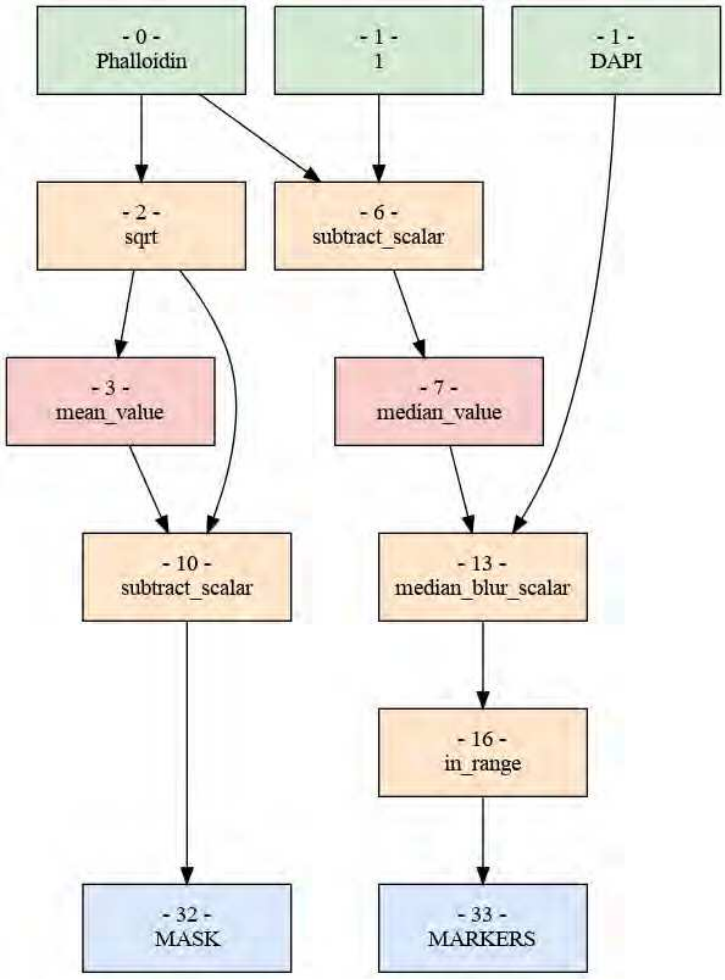


Figure 5.5: An example of evolved MAGE graph for Instance Segmentation

types. One major limitation was the inability to define functions to accommodate all possible cases as more types were introduced. This led MT-CGP genomes to often get trapped in suboptimal local optima, with limited means of escape due to the expanding search space. Additionally, the reliance on problem-dependent solutions hindered collaborative efforts in the domain.

In this paper, we introduced Multimodal Adaptive Graph Evolution (MAGE), a generalized approach based on CGP that separates genome chromosomes akin to how PushGP separates stacks. This separation by types enables several optimizations, notably improving convergence by allowing the mutation operation to assess node states more effectively. Pre-fixing individuals before evaluation significantly enhances their performance.

We conducted a comparative study between MT-CGP and MAGE for multimodal program synthesis tasks, marking the first time a CGP method has been applied in this context. Using the first 10 problems from the PSB2 benchmark with identical function libraries and parameters for both methods, we found that MAGE consistently outperformed or matched MT-CGP across all metrics. It exhibited superior generalization capabilities and tended to converge faster, irrespective of MT-CGP's performance.

In conclusion, we posit that constraining and assisting evolution is crucial in GP evolution, aiming to produce individuals that effectively explore the search space. MAGE represents a significant advancement in this direction, offering a more efficient and robust approach for solving multimodal problems in various domains. In the applicative domain of biomedical data analysis, MAGE could constitute a significant contribution by offering access to a novel type of machine learning algorithm capable of handling a wide array of data

types, including biological, clinical, and image data. In this critical domain, the MAGE algorithm generates interpretable graphs, which hold paramount importance for both physicians and industry professionals seeking verifiable and certifiable solutions.

Chapter 6

Discussion and Conclusion

After nearly four years spent conceptualizing, designing, and exploring around CGP, I am honored and humbled by the opportunity to publish this work in **Nature Communications** [1], an interdisciplinary journal which reflects the complex nature of this project and the wide range of domain in which Kartezio may prove future utility. To me, Kartezio represents a versatile core (which tends towards "Hexagonal Architecture"), with the capacity to be deployed to a myriad of important contemporary challenges, beyond the medical realm. As a proof of concept for Kartezio's versatility, I utilized it for the "Minecraft Open-Endedness Challenge" and managed to construct towers within the Minecraft game, and I even tried to segment manga dialogue bubbles with former intern students by always using the same core. The heart of Kartezio was designed to be modular, i.e. a set of "blocks" where every component can be replaced to adapt the whole behavior. This is similarly seen in the Keras library developed by François Chollet [223], with all the DL components (Layers, Loss, Optimizers...).

In my opinion, the success of Keras and its community acceptance are attributed to this modularity. DL (as discussed in **Chapter 2**) has been an incredible source of inspiration for this project, especially regarding its vast range of architecture over the past decade.

Throughout my PhD, I have been fortunate to be fully integrated within a team of biology researchers and receive continuous support from my computer science peers. It is this symbiosis between Biology and CS that led to the culmination of this interdisciplinary endeavor, presenting a panel of potential applications for my team at CRCT and the broader field of biology. Beyond that, I have also contributed to the world of EC, specifically extending CGP for Instance Segmentation and 3D image processing (as seen in **Chapter 3**). I have also initiated further enhancements of Kartezio (and by extension CGP), particularly by enabling the inclusion of an "unlimited" number of types within the program, and not merely externally (as discussed in **Chapter 5**). Even if this extension is currently only in its embryonic stage, it has clear utility within the ecosystem in the near future. Lastly, I had the privilege to participate in the inception of the next projects, especially with Yuri Lavinas, who is jointly supervised by Sylvain Cussat-Blanc and pathologists at IUCT, focusing on breast cancer. With a background heavily leaning towards EC, Yuri Lavinas aims to explore *Lexicase Selection* [3] and *Active Learning* through the co-evolution of dataset and model. Additionally, he is working on a "Human in the loop" model designed to assist doctors in real-time automatic annotation of ROIs.

Following the initial development of Kartezio and its deployment on the

four Use Cases provided in the **Nature Communications** article, the next step is to dedicate substantial time and engineering resources to thorough optimization of the system and development of a user interface for non-experts to facilitate uptake of Kartezio in the biomedical community. Having completed the publication of our recent manuscript and the writing of this thesis, it is now possible to dedicate my time towards developing around the public Python Package (with Jax [224], Haiku [225] and Numba [125] libraries). The knowledge accumulated over many training sessions should be collected, shared and made accessible on a public platform to optimize energy consumption.

The following sections offer hints of possible cooperations with DL and put Kartezio's contribution to the initial challenges into perspective.

6.1 Perspectives for Computer Science

As described above, Kartezio is very flexible and versatile, a strength that should be exploited to find applications for other problems. I'm thinking in particular of the extension to image classification (supplemented by other data) with endpoints using existing classifiers, such as SVM or DT. On the other hand, medical imaging is not limited to microscopy, and there are plenty of other areas to explore. I'm thinking in particular of MRI and radiology, both already under exploration in the REVA team.

One of the main interests I have for the future is to find a way of recombining experts with each other, in a post-evolution context, in order to compose better performing models, while remaining similar. Although very preliminary,

this is an area I've started to explore and I'll go into more detail in the next section.

For the rest of this part, I'm mainly focusing on ideas involving cooperation between both CGP and DL. Three major axes emerge: **i)** CGP enhancing DL by providing external processing, **ii)** the inclusion of ANN in CGP-generated programs, **iii)** DL enhancing CGP (qualified as *Model Induction* in **Figure 1.2**). It is also important to note that only case **iii)** preserves key points such as XAI while case **i)** could lead to a reduction of the computational cost.

One of the greatest motivations I have developed from the outset of my interaction with CGP is to provide and maintain a form of experience that it acquires through various training sessions. A bit like a human engineer capable of writing programs, who gains experience and assembles functions better and better with a certain intuition. This motivation is supported by the fact that, with each generation, the offspring is often not only less good than the parent, but in most cases completely unusable. Perspectives addressing this issues are discussed in **Sections 6.1.8 and 6.1.7**.

6.1.1 Post-Evolution recombination research

In the realm of TGP, a point crossover globally entails the random fusion of two trees, achieved by cutting and inversely grafting their branches. This operation typically involves two parent trees. Contrastingly, in CGP, the prevalent $1 + \lambda$ ES employs a singular population and the crossover operator is effectively discarded. In my opinion, the crossover strategies in

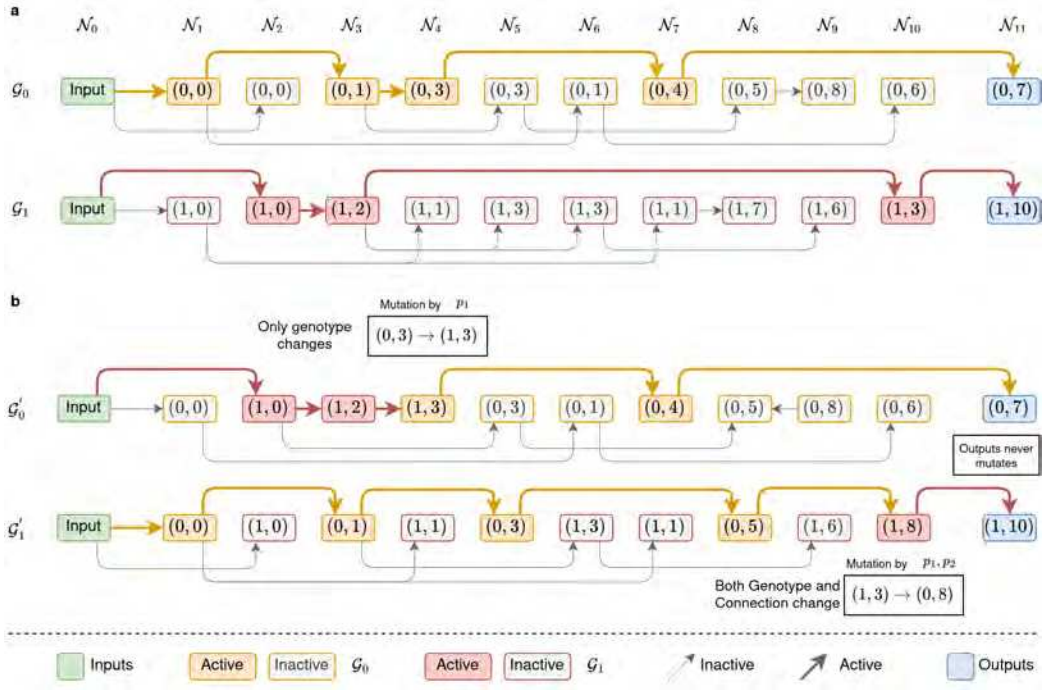


Figure 6.1: **Illustrative example of Post-Recombination between 2 genotypes**

CGP [226] often appear underserved, primarily because we tend to seek graph operations [138] by drawing inspiration directly from TGP or by only crossing genotypes [137]. A more recent approach called "Block-Crossover" was introduced by Husa and Kalkreuth [226] and uses both active graph and ADF-like blocks to reorganise the offspring genotypes. For me, this is the most promising approach, since it exploits at the same time the resources of the active graph and the beauty of the core feature of CGP [124, 133] that always produces valid DAG from a given genotype.

Minimal Algorithm Overview

Based on this work, I also propose an algorithm mixing both active graph and genotype and extend CGP crossover to more than 2 parents. To this end, I suggest stacking at least two genotypes on a 3rd dimension: $\{\overset{dim_1}{Genotypes}, \overset{dim_2}{Nodes}, \overset{dim_3}{Genes}\}$ and follow-up the alterations by changing randomly some connections using the new dimension. As an illustrative example, two genotypes $\mathcal{G}_0$ and $\mathcal{G}_1$ and their corresponding active/inactive graph are represented in **Figure 6.1**. Only the connection value C is represented here, under the form of a tuple $C = (g, c)$ where g is the genotype index and c the connection node index. For this specific operation, all the connection values C_i are encoded with coordinates $C_i = (i, c)$, i being the index of the current genotype itself (see **Figure 6.1a**). In other words, by default, all nodes are connected within the same genotype. In this example, the node $\mathcal{N}_4$ from the first genotype $\mathcal{G}_0$ belongs to the active graph and is initially pointing to coordinates $(0, 3)$, corresponding to $\mathcal{N}_3$ of $\mathcal{G}_0$. In **Figure 6.1b**, $\mathcal{N}_4$ first value has mutated to 1: $(0, 3) \rightarrow (1, 3)$, now pointing to $\mathcal{N}_3$ of $\mathcal{G}_1$; thus inducing crossover operation between $\mathcal{G}_0$ and $\mathcal{G}_1$, with the former inheriting a segment of the latter. Consequently, the remaining part of $\mathcal{G}_0$ ($\mathcal{N}_3$ and its predecessors) are supplanted by the active nodes of $\mathcal{G}_1$, giving a novel genotype $\mathcal{G}_0^{'}$ (visible in **Figure 6.1b**). On $\mathcal{G}_1$ $\mathcal{N}_{10}$, both coordinates has been affected by mutations ($\mathcal{N}_{10} : (1, 3) \rightarrow (0, 8)$); thus providing a second offspring: $\mathcal{G}_1^{'}$ (visible in **Figure 6.1b**). If any of the new latent genotype $\mathcal{G}_i^{'}$ performs better than their correspond previous version $\mathcal{G}_i$, then the new one is kept, otherwise the former one is retrieved.

Iteration are made a certain number of times (n_iter) to try to combine several genotypes.

With such a minimalist algorithm, two problems arise: **i)** the too-high probability of altering genotypes by completely destroying the original graph, **ii)** the possibility that all genotypes will converge towards the same model by reconnecting directly to the best genotype, thus eliminating all diversity. These issues are addressed in the following subsections.

Exploiting the active graphs and "vestiges"

Active graph, as the backbone of the generated genotype stores the real resource and program information. Therefore, any attempts at refining and combining CGP must respect and exploit this active graph. In the example shown in **Figure 6.1a**, nodes $\mathcal{N}_7$,$\mathcal{N}_4$,$\mathcal{N}_3$, and $\mathcal{N}_1$ are constituting the active graph of $\mathcal{G}_0$, and each has a probability p_1 of seeing its index genotype mutated as it is the case for $\mathcal{N}_4$. Inactive nodes ("vestiges" of evolution) cannot mutate and are therefore ignored by the algorithm. $\mathcal{N}_4$ is now pointing $\mathcal{N}_3$ of $\mathcal{G}_1$ that is already part of the $\mathcal{G}_1$ active graph($\mathcal{N}_{10}$, $\mathcal{N}_3$, $\mathcal{N}_2$). The process continues and no mutation was performed on $\mathcal{N}_3$ or $\mathcal{N}_2$ thus ending the composition of $\mathcal{G}_0'$. On the other hand, $\mathcal{G}_1$ is initially composed of $\mathcal{N}_{10}$, $\mathcal{N}_3$, $\mathcal{N}_2$ active nodes and $\mathcal{N}_{10}$ mutates its genotype index (respecting probability p_1) and its node index (respecting probability $p_1 \times p_2$, only if genotype index mutation occurred). $\mathcal{N}_{10}$ is now pointing to $\mathcal{N}_8$ of $\mathcal{G}_0$ that do not belong to the current active graph. $\mathcal{N}_8$ is forced to become active and a whole path is activated ($\mathcal{N}_8$, $\mathcal{N}_5$, $\mathcal{N}_3$, $\mathcal{N}_1$) using the connection indices. This is an example where potential vestiges can re-emerge and be

reintegrated into the active graph. p_2 is introduced to provide a change to hook different node positions since functions are not aligned in the genotype as demonstrated in **Chapter 4**.

Excluding output connections

The second important point is to lock the output connections. By doing this, we force the preservation of the last node of each genotype, which prevents all outputs from pointing to the last node of the best genotype. In this case, all genotypes would have the best score but would be identical, which is absolutely not the initial objective.

Preliminary Study and Results

A very basic test was performed to confirm that this algorithm could improve at least one genotype. So I re-exploited the genotypes I already have from the article and the use cases presented in **Chapter 3**. Use Case 1 (see **Figure 3.4**) was used as it is the most simple, simply performing semantic segmentation (with one single output). **Figure 6.2** shows the scores of 10 models (among 100) before and after this process. Iterations are made using the training dataset only, and final new models are assessed on both training (left) and test (right) sets. After 1000 iterations and $p_1 = 0.25$, $p_2 = 0.25$, only 6 improvements has been made. The average score risen from 0.8641 to 0.8687 for the training and 0.8563 to 0.8646 for the test (illustrated by horizontal bars).

A second experiment was made to assert the possibility to use a large number of genotypes. It was achieved by pulling the 100 models together and perform

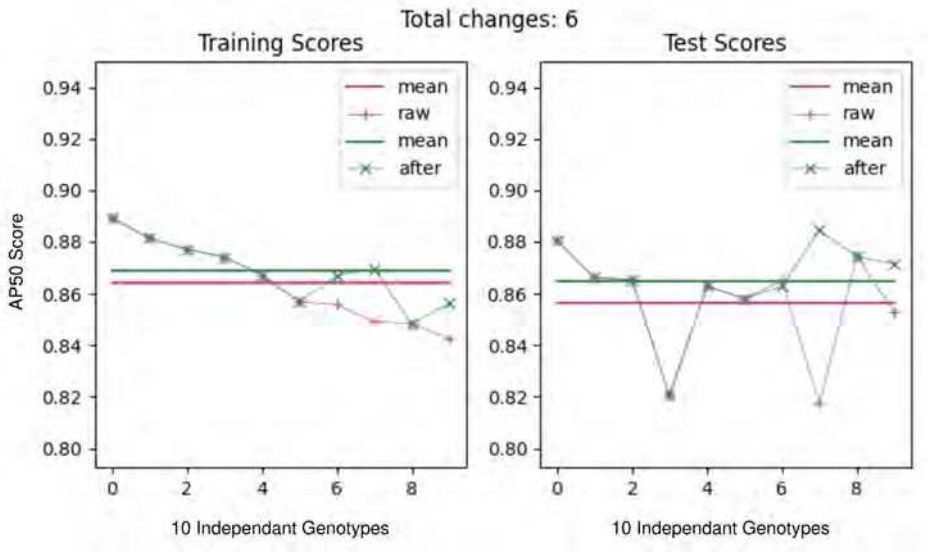


Figure 6.2: **Improvement after Post-Training recombination**
The experiment is made if 10 models (x-axis) over $n_iter = 1000$ iterations. A total of 6 changes (enhancements) were made and accepted, thus giving the new scores in green.

the same experiment with $p_1 = 0.25$, $p_2 = 0.25$. As the visualization of 100 models is more complex, all results are summarized in **Table 6.1**.

Conclusion

This process has been first used as a post-evolution method, where a bundle of genotypes is already trained, and the entire bundle forms a new population to recombine. The process described before is not reserved for 2 parents for a crossover but could is extended to any size, which seems to be an advantage regarding the results (**Table 6.1**). The first step would be to investigate the impact of the p_1 and p_2 parameters and adapt Kartezio to allow parallel evaluations. The current implementation also has the advantage of never causing models to explode in size, and therefore doesn't suffer from any

n	p_1	p_2	n_iter	Valid	Train R	Train A	Test R	Test A
10	0.25	0.25	1000	6	0.8641	0.8687	0.8563	0.8646
100	0.25	0.25	100	24	0.8696	0.8720	0.8462	0.8485
100	0.25	0.25	200	54	0.8696	0.8738	0.8462	0.8482
100	0.25	0.25	300	87	0.8696	**0.8760**	0.8462	**0.8491**

Table 6.1: **Parameters and Results summary**
n: number of genotypes recombined all together; *Valid*: number of valid recombinations, inducing an increase of fitness; *Train R*: raw training score; *Train A*: training score after process; *Test R*: raw test score; *Test A*: test score after process

bloating effect. So even when recomposing the genetic material of 100 models, the programs generated at the end are of similar complexity.

6.1.2 Evolution of Convolution kernels

As the convolution filter is a key element for CNNs, it is necessarily an important filter to keep for CGP. In Kartezio, convolution is used through several well-known kernels (such as Sobel, Roberts, Laplacian...). We should get closer to CNN and find a way of evolving kernels directly with CGP. One can see two approaches: **i)** first evolve a kernel as an image and apply the convolution as Endpoint, for a specific task such as edge detection, noise reduction... Then add these kernels to the function library for future use. **ii)** utilize MAGE model which is more flexible option. It allows the use of kernels within the graph with dedicated operators and, above all, the possibility of adapting them to certain features of the image: a given kernel depending on the average intensity of the image, or a variable SE size depending on the average area of the blobs detected in the image, for example. In the

latter case, the filters would be adaptive and could improve the segmentation results.

6.1.3 Towards If-Then-Else statements for CGP

An appealing progression to the MAGE model (**Chapter 5**) would be the incorporation of the **If-Then** statement as a function potentially with a signature like: $IfThen(\mathcal{B}, \mathcal{A}) \rightarrow \mathcal{A}$, here, $\mathcal{B}$ (standing for $\mathcal{B}oolean$) might emerge as a novel type resulting of comparison test for instance. If the test is *True*, then it returns the $\mathcal{A}rray$, otherwise it returns $Zero$ (though the efficacy of One could be assessed in a comparative study).

By taking full advantage of the model's generic and composite properties, we can create a $Choice_{\mathcal{A}}$ type to associate two $\mathcal{A}$rrays ($CreateChoice(\mathcal{A}_1, \mathcal{A}_2) \rightarrow Choice_{\mathcal{A}}$). From there, it is easy to implement the **If-Then-Else** statement with: $IfThenElse(\mathcal{B}, Choice_{\mathcal{A}}) \rightarrow \mathcal{A}$.

In this enhanced version, the parsing operation might undergo iterative processing. Initially, elements excluding $Choice$ would be decoded, and the B values computed. Based on these B outcomes, the relevant branch would be decoded and executed. If another $Choice$ node is encountered, the cycle repeats until the entire graph is parsed.

6.1.4 Cartesian Genetic Programming for Neural Architecture Search

CGP has already been employed for NAS (CGPANN [227, 228]) and I think a multimodal version such as MAGE could help in this line of research.

We would first have to check that it's possible to reproduce and encode existing architectures within a genotype (by hand). Also, it is possible to concentrate on smaller components such as memory units (LSTM, GRU), which are very similar to electronic circuits and for which CGP could be a good ally. In this situation, we are clearly in case i) where the final model remains an ANN. In my opinion, the main challenge lies in respecting the tensor dimensions that need to be arranged throughout the graph, starting from the end. It can then be difficult to reconnect to the inputs correctly. This may be a case where the graph decoding might be helped by the addition of a path-finding algorithm or by a bidirectional decoding approach.

6.1.5 Cartesian Genetic Programming for generalist Deep Learning models

Some Artificial Neural Networks (ANN) are versatile in their implementation and are trained on various data. This is the case of Cellpose [39] for example, an architecture that integrates "image-style" into the latent space to refine segmentation. In this way, Cellpose offers a generalist model, whereas Kartezio favors the training of specialized models. In this context, I can see Kartezio being used as a preprocessing tool for generalist models. All datasets can then use the same ANN model, but the preprocessing is specific to each set. This is a very easy implementation by simply placing the ANN as Endpoint. Kartezio then optimizes the preprocessing so that the ANN generates the best result. It is a cooperative approach that can save energy

by avoiding the need to re-train hundreds of ANNs. It may also be a way of improving current ANN performances by removing some noise in images.

6.1.6 Specialist Artificial Neural Networks as primitives

In a somewhat similar manner, an *image-to-image* ANN can play the role of a filter and thus be integrated into the CGP graph itself. This could be done easily via primitives calling for inference from a specific model: "Holistically-Nested Edge Detection" [24] model could be used to detect edges, Cellpose [39] to produce cell probability map, Stardist [69] for convex blob detection... For me, this is the closest case to the hybridization between CGP and ANN. On the other hand, we lose transparency, we lose frugality, and the computation time required to train the model can become very disruptive. I see this hybridization as possible only if new ANNs are trained to serve as advanced filters while remaining very light and accessible. At the end, this idea also raises the possibility of simply evolving IP programs to perform specific functions and then including them as primitives (allowing a composite architecture).

6.1.7 Evolution as a Chess Game

The first approach that could be considered is based on Monte-Carlo Tree Search (MCTS). Indeed, I see successive generations as a chess game with a function that can change (a piece to play) or connections/parameters (the next position of the piece). The new state is given by the mutation oper-

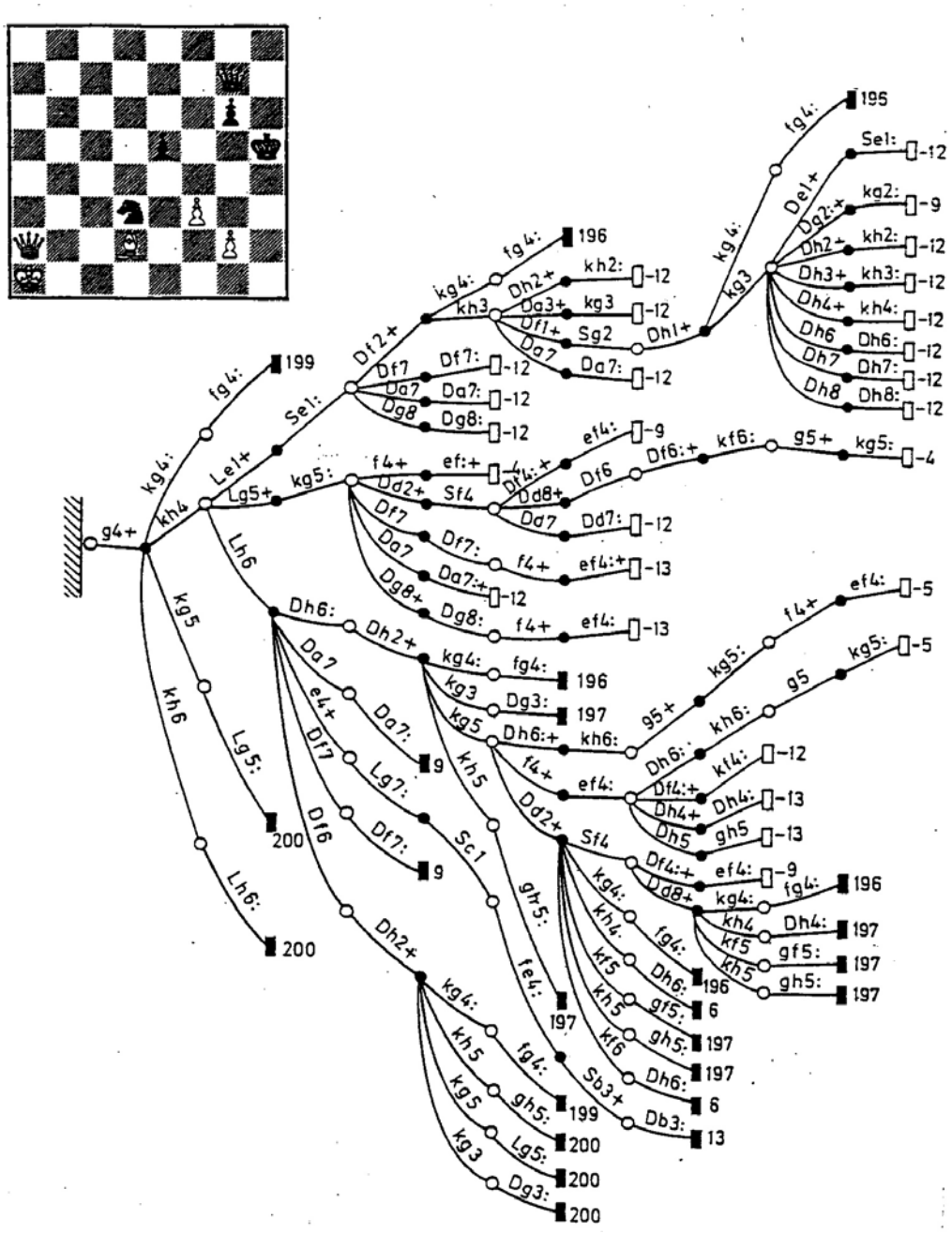


Figure 6.3: **The decision-tree of a chess study by M. Botvinnik and S. Kaminer (white to play and win).**
The start position is given in the top left-hand corner.
Source: Figure 3 (p12) [229]

ator (as the mathematical model described by Botvinnik [229], illustrated in **Figure 6.3**). Of course, the number of combinations and trees to explore are absolutely colossal in GP. So I think it could be interesting to use an external model capable of pre-filtering possible mutations and discarding irrelevant ones. This model could be a MCTS [230] method guided by DNN. This is exactly the essence behind the work of *Google DeepMind* and the "Alpha series". Firstly, by allowing *AlphaGo* [231], to be trained by Deep Reinforcement Learning (DRL) to play *Go*, a board game with *EXPTIME*-class combinatorics. Later, it has been extended to Chess and Shogi with *AlphaZero* [232]. In the medical field, *AlphaFold* [233] has become the master of Critical Assessment of protein Structure Prediction. Then, in the field of mathematics, *AlphaTensor* [234] has made it possible to find faster matrix multiplication algorithms, a field that has been extensively studied by human mathematicians for decades. Collectively, this work suggest that a hybrid approach with GP in general might be possible. Thus, an hypothetical "*AlphaGP*" could emerge with a model reducing the search space. The idea is not to destroy randomly generated diversity, but to channel and above all avoid the generation of algorithms that make no sense (3 binary thresholds in a row for example). A very recent article from DeepMind presented *AlphaCode* [235] that discovers faster sorting algorithms in assembly. In my opinion, this is an excellent basis for considering Transformers [104] as encoders for primitives and therefore, by extension, of algorithms and graphs (in the case of CGP). Given a Genotype, a mutation would consist in choosing randomly one bundle of mutations among a pre-selected one.

6.1.8 Genotype Correction

In the same vein as the previous section, and with the aim of using an external model to promote good offspring, I propose here the idea of a genotype corrector. This operation would occur after the mutation process with the objective to make as few modifications as possible and improve genotype fitness. The active graph (phenotype) is mathematically a sentence where a function is a word. This sentence may contain grammar mistakes, word/letter omissions or additions that can be corrected with a spellchecker-like model. Some words could be replaced by synonyms to make the sentence even better. These corrections can be made by DL (where graph representation encoding is well implanted [236, 237]) and, in particular, LLM. One could envision a Transformer model to encode the genotype, correct it and return the corrected version (a spelling corrector tranformer [238]). So we need to answer 2 questions: How does the LLM learn? How do we encode a function and its parameters, like word embedding? To answer the first question, I would investigate DRL Transformers [239] as an intuitive direction where the reward could be a weighted formula including the difference between the genotype before and after (to preserve the original found by random mutation) and a second term about the impact of the change as the score difference. A high reward would involve tiny changes and consequent fitness improvement. I already try to address the second point with very preliminary search, by using VAE model on the MNIST [240] dataset. Where the IP function is given directly into the latent space as a weight-vector that must be learnt by backpropagation. The models input are the original image, the *One-Hot En-*

coding of the function and the parameter values, the latent space is stacked with the learned embedding in order to decode and apply the IP filter to the image. All filters and parameters can be embedded into a vector and used by Transformer model.

6.2 The impact of Kartezio on Biomedical Image Analysis

As detailed in the **Chapter 3**, Kartezio has features that make it a game-changer in the field. The main strengths of Kartezio can be summarized in the following five points.

6.2.1 Explainable AI

One of the main characteristics of Kartezio is that it is fully transparent and interpretable by humans. This feature is highly valuable in biomedicine. In biological sciences it is important that investigators are aware of the process followed by the AI to obtain a given result in order to exclude biases and errors of interpretation. One of the most important limitations of the currently available automated image analysis software is that its leave a very small margin to users to understand how a given analysis pipeline was generated and how image segmentation results were obtained. I am currently working on a user-friendly application wrapping Kartezio for scientists all over the world. The application I am creating will be initially tested by members of our team and European partners. It should allow biomedical scientists with no experi-

ence in CS to rapidly produce IP pipelines, specialized for a given task. The programs used for IP could be, in the future, included in the material and methods section of biomedical science articles, which for me would prove interest of our approach. In clinical practice the advantages of Kartezio-based programs are even more immediately evident. As discussed in our accepted article, our computational strategy generates IP pipelines that are transparent, a feature that is becoming necessary to track the logic of decisions made by algorithms. This is in contrast with "black-box" DL based models, where the lack of interpretability is a potentially important drawback, particularly in the regulatory context, where the process of extracting information from medical images must be fully verifiable. This concept is also in line with the main principles of the *European Union AI Act*, currently under discussion by the European Council, that regulates the use of AI in sensitive fields such transportation and health care [189]. On this line, the advancement of Kartezio is even faster than in biology. As said above, Kartezio is currently applied to breast cancer samples digitalized images by Yuri Lavinas. The postdoctoral fellow is applying for the first time a Kartezio-derived software to hematoxylin-eosin clinical routine samples and is starting to develop the procedure of "Human in the loop" (see below) that only a couple of months ago we proposed as potentially feasible (see accepted article, **Chapter 3**). We are planning, in collaboration with clinicians at the IUCT to test the application of the MAGE model to melanoma patients in order to integrate multimodal data (clinical data, tumor tissue sample imaging, composition of tumor microenvironment, *ex vivo* biological data, prognostic markers, etc.) into pipelines attempting to predict which patients are more likely to re-

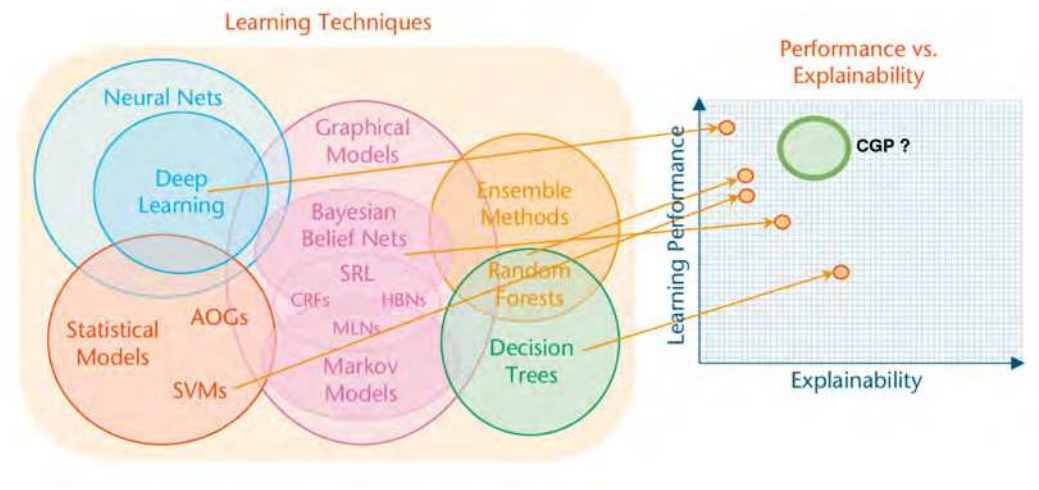


Figure 6.4: **CGP-based methods and XAI**
Possible placement of CGP-based methods for biomedical segmentation, in terms of Learning Performance and Explicability. The green circle suggests the position of CGP-based methods: performance similar to DL, but considerably more explainable.
Original Figure: DARPA's XAI Program, Figure 1 [6]

spond to immunotherapy. Immunotherapy is indeed very expensive for the national health system, heavy for patients and can potentially cause serious and irreversible side effects. It would be a major clinical achievement if a MAGE model might help to better stratify melanoma patients undergoing immunotherapy. In addition, we are considering to offer Kartezio-based analysis services to the IUCT clinical trial office that is very active in developing phase one clinical trials and constantly interested to ameliorate the analytical procedures related to these trials. Discussion will soon start with clinicians and hospital officers. In perspective this approach can be applied to multi-center clinical trials and to other clinical applications.

6.2.2 Few-Shot Learning

Kartezio is trainable on very small datasets. This second characteristic is precious both in biology and medicine. In biology, the availability of images is typically limited because of the issues related to the culture of healthy donor and patient derived cells and the complexity and cost of the experiments that are behind each data set. The capacity of Kartezio to provide reliable analysis on small datasets that can be easily made available to readers in scientific articles can strongly contribute to the accuracy, transparency and repeatability of biological image analyses. Moreover, it must be considered that biology labs are much less equipped in instruments and human resources when compared to big clinical departments. They therefore cannot effort expensive commercial software for image analysis. Indeed, they might strongly profit of Kartezio-derived models or applications. The few-shot learning allowed by Kartezio can have major impact in clinical routine too. In many clinical situations, annotations must be performed by expert pathologists who have limited time to dedicate to non-diagnostic activity. It is therefore crucial to minimize experts' effort by designing ML algorithms that can handle the smallest datasets possible, while keeping adequate generalization capacity. To address this point, our study utilizes CGP to generate solutions that leverage the cumulative expertise of human CV experts. In other words, CGP produces pipelines by assembling algorithms that integrate decades of existing research and development in CV. The possibility to upgrade Kartezio to an active "Human in the loop" training mode to further reduce the size requirement of the training dataset and improve performance is expected

to further improve efficiency solution. In such a configuration, intermediate results from Kartezio-generated pipelines are displayed while the expert annotates. At a given point, the expert does not annotate anymore, but simply amends the automatically generated annotations. In turn, expert-operated adjustments are informative to pursue the training of Kartezio [1].

6.2.3 Customization

A key characteristic of Kartezio is that it is easily customizable even by biologists and clinicians not able to code. This characteristic dramatically augments the potential of integrating human knowledge into the various pipelines that will be generated starting from CGP-based algorithms. As mentioned above, it is tempting to speculate that the interpretability and customizability of Kartezio might favor the development of a new community of image analysis experts (working at the frontier between computer science, biology and medicine) sharing pipelines, primitives, operators, preprocessing, endpoints, and ideas.

6.2.4 Versatility

As shown in Paper Figure 3 of the submitted article our CGP-based algorithm is adaptable to different imaging techniques, scales, and resolutions. This is an essential feature of Kartezio that makes it a potential ideal partner for pathologists. One important limitation of neuronal network-based analysis of tissue samples is that staining (texture, shadows of color, intensity, contrast) vary from one laboratory to another. ANN therefore requires additional

training time and adjustments when handling heterogenous cohorts [241]. As shown in our submitted article, Kartezio is intrinsically able to overcome this important limitation a feature that makes it an ideal tool to handle sample heterogeneity in clinical routine. It is difficult to forecast the real impact that automated analysis will have in the future in routine pathology, this depends on many parameters that are difficult to evaluate at present. However, in principle, CGP-based analysis appear to have a strong advantage over more extensive and less versatile DL algorithms.

6.2.5 Frugality

Finally, it is important to underline a characteristic of Kartezio that is very relevant nowadays: its ecological potential. It is clear that when compared to DL based approaches CGP-based algorithms are more frugal since they require a much lower energy and hardware power. A precise estimation of the reduced energy costs of Kartezio when compared to NN-based AI is presently elusive. However, it is conceivable the energetic gain might become increasingly important overtime. Another feature of CGP strictly related to its ecological potential is that it is broadly accessible with basic hardware (1 CPU is enought). This feature obviously contributes to make this IP procedure accessible, embedded, cost-effective and ecological. In fact, REVA team recently collaborated with a company to equip city lighting installations with intelligent sensors, embedding CGP-IP-based filters [144, 242]. Multi-objective optimization was used to find a balance between performances and computational cost.

6.3 Conclusion

In this period of evolution and hype surrounding AI, the focus is increasingly - and often obsessively - on improving accuracy rates, sometimes by very small margins. This desire, while natural, often masks a fundamental question regarding the right metrics to optimize within this technology. The research presented in this thesis around CGP is more than just a presentation of an illustrative use of frugal, accessible and explainable AI; it suggests that, in our principally DL-driven search, we may have overlooked simpler, more elegant solutions, such as the discovery of new IP operators and their optimization. Discovering new kind of algorithms, such as Mathematical Morphology or the design of specific convolution kernels, could offer more efficient paths than the maintenance of "gradient quality" during backpropagation or the pushing of CGPs' capabilities to their limits.

This leads to my final point: accessibility. AI and its potential benefits should not be reserved for those with extensive computing resources. More frugal and cost-efficient models enable more people, regardless of their technical or financial capabilities, to harness the power of AI. This approach also aligns with today's urgent calls for more sustainable approaches to human activity. By consuming less energy and requiring less hardware, such models tend to be more respectful of the environment. As such, we present Kartezio to the community as a highly effective, flexible, accessible, and more holistically sustainable approach to image analysis.

References

1. Cortacero, K., McKenzie, B., Müller, S., Khazen, R., Lafouresse, F., Corsaut, G., Van Acker, N., Frenois, F.-X., Lamant, L., Meyer, N., *et al.* Evolutionary design of explainable algorithms for biomedical image segmentation. *Nature communications* **14,** 7112 (2023).
2. Filali, L., Puissegur, M.-P., Cortacero, K., Cussat-Blanc, S., Khazen, R., Van Acker, N., Frenois, F.-X., Abreu, A., Lamant, L., Meyer, N., *et al.* Ultrarapid lytic granule release from CTLs activates Ca2+-dependent synaptic resistance pathways in melanoma cells. *Science Advances* **8,** eabk3234 (2022).
3. Lavinas, Y., Cortacero, K. & Cussat-Blanc, S. *Evolving Graphs with Cartesian Genetic Programming with Lexicase Selection* in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (2023), 1920–1924.
4. Cortacero, K., Fischer, T. & Demiris, Y. *RT-BENE: A dataset and baselines for real-time blink estimation in natural environments* in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops* (2019).
5. Wilson, D. G., Cussat-Blanc, S., Luga, H. & Miller, J. F. *Evolving Simple Programs for Playing Atari Games* in *Proceedings of the Genetic and Evolutionary Computation Conference* (Association for Computing Machinery, Kyoto, Japan, 2018), 229–236. ISBN: 9781450356183. https://doi.org/10.1145/3205455.3205578.
6. Gunning, D. Explainable artificial intelligence (xai). *Defense advanced research projects agency (DARPA), nd Web* **2,** 1 (2017).
7. Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., *et al.* Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion* **58,** 82–115 (2020).

8. McKenzie, B. & Valitutti, S. Resisting T cell attack: Tumor-cell-intrinsic defense and reparation mechanisms. *Trends in Cancer* (2023).

9. McKenzie, B., Khazen, R. & Valitutti, S. Greek fire, poison arrows, and scorpion bombs: how tumor cells defend against the siege weapons of cytotoxic T lymphocytes. *Frontiers in Immunology* **13** (2022).

10. Das, J. & Jayaprakash, C. *Systems immunology: An introduction to modeling methods for scientists* (CRC Press, 2018).

11. O'Donnell, J. S., Teng, M. W. & Smyth, M. J. Cancer immunoediting and resistance to T cell-based immunotherapy. *Nature reviews Clinical oncology* **16,** 151–167 (2019).

12. Boole, G. *The mathematical analysis of logic: being an essay towards a calculus of deductive reasoning* (Macmillan, Barclay, & Macmillan, Cambridge, 1847).

13. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *nature* **521,** 436–444 (2015).

14. LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. & Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural computation* **1,** 541–551 (1989).

15. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86,** 2278–2324 (1998).

16. Krizhevsky, A., Sutskever, I. & Hinton, G. E. *ImageNet Classification with Deep Convolutional Neural Networks* in *Advances in Neural Information Processing Systems* (eds Pereira, F., Burges, C., Bottou, L. & Weinberger, K.) **25** (Curran Associates, Inc., 2012). `https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

17. Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM* **60,** 84–90 (2017).

18. Zhao, G., Huang, G., He, H., He, H. & Ren, J. Regional spatiotemporal collaborative prediction model for air quality. *IEEE Access* **7,** 134903–134919 (2019).

19. Robbins, H. & Monro, S. A stochastic approximation method. *The annals of mathematical statistics,* 400–407 (1951).

20. Cauchy, A. *et al.* Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris* **25,** 536–538 (1847).

21. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *nature* **323,** 533–536 (1986).

22. Xu, L., Lu, C., Xu, Y. & Jia, J. *Image smoothing via L 0 gradient minimization* in *Proceedings of the 2011 SIGGRAPH Asia conference* (2011), 1–12.

23. Cho, H., Lee, H., Kang, H. & Lee, S. Bilateral texture filtering. *ACM Transactions on Graphics (TOG)* **33,** 1–8 (2014).

24. Xie, S. & Tu, Z. *Holistically-nested edge detection* in *Proceedings of the IEEE international conference on computer vision* (2015), 1395–1403.

25. Roberts, L. *Machine perception of 3-D solids, optical and electro-optical information processing* 1965.

26. Irwin, F. *et al.* An isotropic 3x3 image gradient operator. *Presentation at Stanford AI Project* **2014** (1968).

27. Prewitt, J. M. *et al.* Object enhancement and extraction. *Picture processing and Psychopictorics* **10,** 15–19 (1970).

28. Kirsch, R. A. Computer determination of the constituent structure of biological images. *Computers and biomedical research* **4,** 315–328 (1971).

29. Canny, J. A Computational Approach To Edge Detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **PAMI-8,** 679–698 (Dec. 1986).

30. Deriche, R. Using Canny's criteria to derive a recursively implemented optimal edge detector. *International journal of computer vision* **1,** 167–187 (1987).

31. Paillou, P. Detecting step edges in noisy SAR images: a new linear operator. *IEEE Transactions on Geoscience and Remote Sensing* **35,** 191–196 (1997).

32. Matheron, G. & Serra, J. *The birth of mathematical morphology* in *Proc. 6th Intl. Symp. Mathematical Morphology* (2002), 1–16.

33. Serra, J. & Soille, P. *Mathematical morphology and its applications to image processing* (Springer Science & Business Media, 2012).

34. Beucher, S. & Meyer, F. in *Mathematical Morphology in Image Processing* 433–481 (CRC Press, 1993).

35. Guo, Q., Wang, Y., Yang, S. & Xiang, Z. A method of blasted rock image segmentation based on improved watershed algorithm. *Scientific Reports* **12,** 7143 (2022).

36. Zhang, Y. & Xu, D. Q. *Improved watershed algorithm for cell image segmentation* in *Advanced Materials Research* **546** (2012), 464–468.

37. Grau, V., Mewes, A., Alcaniz, M., Kikinis, R. & Warfield, S. Improved watershed transform for medical image segmentation using prior information. *IEEE Transactions on Medical Imaging* **23,** 447–458 (2004).

38. Bai, M. & Urtasun, R. *Deep watershed transform for instance segmentation* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 5221–5229.

39. Stringer, C., Wang, T., Michaelos, M. & Pachitariu, M. Cellpose: a generalist algorithm for cellular segmentation. *Nature methods* **18,** 100–106 (2021).

40. Hough, P. V. *Machine analysis of bubble chamber pictures* in *International Conference on High Energy Accelerators and Instrumentation, CERN, 1959* (1959), 554–556.

41. Duda, R. O. & Hart, P. E. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM* **15,** 11–15 (1972).

42. Ballard, D. H. Generalizing the Hough transform to detect arbitrary shapes. *Pattern recognition* **13,** 111–122 (1981).

43. Xie, Y. & Ji, Q. *A new efficient ellipse detection method* in *2002 International Conference on Pattern Recognition* **2** (2002), 957–960.

44. Zahn, C. T. & Roskies, R. Z. Fourier descriptors for plane closed curves. *IEEE Transactions on computers* **100,** 269–281 (1972).

45. Persoon, E. & Fu, K.-S. Shape discrimination using Fourier descriptors. *IEEE Transactions on systems, man, and cybernetics* **7,** 170–179 (1977).

46. Nixon, M. S. & Aguado, A. S. in *Feature Extraction and Image Processing for Computer Vision (Fourth Edition)* (eds Nixon, M. S. & Aguado, A. S.) Fourth Edition, 339–398 (Academic Press, 2020). ISBN: 978-0-12-814976-8. `https://www.sciencedirect.com/science/article/pii/B9780128149768000075`.

47. Belongie, S., Malik, J. & Puzicha, J. *Matching shapes* in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001* **1** (2001), 454–461.

48. Freeman, W. T. & Roth, M. *Orientation histograms for hand gesture recognition* in *International workshop on automatic face and gesture recognition* **12** (1995), 296–301.

49. Freeman, W. T., Tanaka, K.-i., Ohta, J. & Kyuma, K. *Computer vision for computer games* in *Proceedings of the second international conference on automatic face and gesture recognition* (1996), 100–105.

50. Lienhart, R. & Maydt, J. *An extended set of haar-like features for rapid object detection* in *Proceedings. international conference on image processing* **1** (2002), I–I.

51. Viola, P. & Jones, M. *Rapid object detection using a boosted cascade of simple features* in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001* **1** (2001), I–I.

52. Viola, P., Jones, M. J. & Snow, D. Detecting pedestrians using patterns of motion and appearance. *International journal of computer vision* **63,** 153–161 (2005).

53. Karami, E., Prasad, S. & Shehata, M. Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images. *arXiv preprint arXiv:1710.02726* (2017).

54. Lowe, D. G. *Object recognition from local scale-invariant features* in *Proceedings of the seventh IEEE international conference on computer vision* **2** (1999), 1150–1157.

55. Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* **60,** 91–110 (2004).

56. Bay, H., Tuytelaars, T. & Van Gool, L. Surf: Speeded up robust features. *Lecture notes in computer science* **3951,** 404–417 (2006).

57. Rublee, E., Rabaud, V., Konolige, K. & Bradski, G. *ORB: An efficient alternative to SIFT or SURF* in *2011 International conference on computer vision* (2011), 2564–2571.

58. Dalal, N. & Triggs, B. *Histograms of oriented gradients for human detection* in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)* **1** (2005), 886–893.

59. Vapnik, V. *The nature of statistical learning theory* (Springer science & business media, 1999).

60. Turing, A. Intelligent machinery (1948). *B. Jack Copeland,* 395 (2004).

61. Samuel, A. L. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development* **3,** 210–229 (1959).

62. Werbos, P. J. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences* PhD thesis (Harvard University, 1974).

63. Zeiler, M. D. & Fergus, R. *Visualizing and Understanding Convolutional Networks* 2013. arXiv: 1311.2901 [cs.CV].

64. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

65. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. *Going deeper with convolutions* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), 1–9.

66. Long, J., Shelhamer, E. & Darrell, T. *Fully convolutional networks for semantic segmentation* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), 3431–3440.

67. Shelhamer, E., Long, J. & Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **39,** 640–651 (2017).

68. Ronneberger, O., Fischer, P. & Brox, T. *U-net: Convolutional networks for biomedical image segmentation* in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18* (2015), 234–241.

69. Schmidt, U., Weigert, M., Broaddus, C. & Myers, G. *Cell detection with star-convex polygons* in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II 11* (2018), 265–273.

70. He, K., Zhang, X., Ren, S. & Sun, J. *Deep residual learning for image recognition* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 770–778.

71. Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M. & Yang, S. *Adanet: Adaptive structural learning of artificial neural networks* in *International conference on machine learning* (2017), 874–883.

72. Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. *Densely connected convolutional networks* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 4700–4708.

73. Chollet, F. *Xception: Deep learning with depthwise separable convolutions* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 1251–1258.

74. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. & Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

75. Zhang, X., Zhou, X., Lin, M. & Sun, J. *Shufflenet: An extremely efficient convolutional neural network for mobile devices* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 6848–6856.

76. Zoph, B., Vasudevan, V., Shlens, J. & Le, Q. V. *Learning transferable architectures for scalable image recognition* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 8697–8710.

77. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L.-C. *Mobilenetv2: Inverted residuals and linear bottlenecks* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 4510–4520.

78. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A. & Le, Q. V. *Mnasnet: Platform-aware neural architecture search for mobile* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), 2820–2828.

79. Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., *et al. Searching for mobilenetv3* in *Proceedings of the IEEE/CVF international conference on computer vision* (2019), 1314–1324.

80. Girshick, R., Donahue, J., Darrell, T. & Malik, J. *Rich feature hierarchies for accurate object detection and semantic segmentation* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), 580–587.

81. Uijlings, J. R., Van De Sande, K. E., Gevers, T. & Smeulders, A. W. Selective search for object recognition. *International journal of computer vision* **104,** 154–171 (2013).

82. Girshick, R. *Fast r-cnn* in *Proceedings of the IEEE international conference on computer vision* (2015), 1440–1448.

83. Ren, S., He, K., Girshick, R. & Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems* **28** (2015).

84. Dai, J., Li, Y., He, K. & Sun, J. R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems* **29** (2016).

85. He, K., Gkioxari, G., Dollár, P. & Girshick, R. *Mask r-cnn* in *Proceedings of the IEEE international conference on computer vision* (2017), 2961–2969.

86. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. & Berg, A. C. *Ssd: Single shot multibox detector* in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14* (2016), 21–37.

87. Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. *You only look once: Unified, real-time object detection* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 779–788.

88. Redmon, J. & Farhadi, A. *YOLO9000: better, faster, stronger* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 7263–7271.

89. Redmon, J. & Farhadi, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

90. Bochkovskiy, A., Wang, C.-Y. & Liao, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934* (2020).

91. Xie, J., Xu, L. & Chen, E. Image denoising and inpainting with deep neural networks. *Advances in neural information processing systems* **25** (2012).

92. Cho, K. Boltzmann machines and denoising autoencoders for image denoising. *arXiv preprint arXiv:1301.3468* (2013).

93. Bourlard, H. & Kamp, Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics* **59,** 291–294 (1988).

94. Kingma, D. P. & Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

95. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* **79,** 2554–2558 (1982).

96. Schuster, M. & Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing* **45,** 2673–2681 (1997).

97. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9,** 1735–1780 (1997).

98. Bakker, B. Reinforcement learning with long short-term memory. *Advances in neural information processing systems* **14** (2001).

99. Cho, K., Van Merriënboer, B., Bahdanau, D. & Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

100. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

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

102. Pennington, J., Socher, R. & Manning, C. D. *Glove: Global vectors for word representation* in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), 1532–1543.

103. Cui, Z., Ke, R., Pu, Z. & Wang, Y. Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. *arXiv preprint arXiv:1801.02143* (2018).

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

105. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., *et al.* An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

106. Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M. & Luo, P. *SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers* in *Advances in Neural Information Processing Systems* (eds Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P. & Vaughan, J. W.) **34** (Curran Associates, Inc., 2021), 12077–12090. `https://proceedings.neurips.cc/paper_files/paper/2021/file/64f1f27bf1b4ec22924fd0acb550c235-Paper.pdf`.

107. Strudel, R., Garcia, R., Laptev, I. & Schmid, C. *Segmenter: Transformer for Semantic Segmentation* in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), 7262–7272.

108. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q. & Rush, A. *Transformers: State-of-the-Art Natural Language Processing* in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (Association for Computational Linguistics, Online, Oct. 2020), 38–45. https://aclanthology.org/2020.emnlp-demos.6.

109. Fel, T., Hervier, L., Vigouroux, D., Poche, A., Plakoo, J., Cadene, R., Chalvidal, M., Colin, J., Boissin, T., Béthune, L., *et al.* Xplique: A deep learning explainability toolbox. *arXiv preprint arXiv:2206.04394* (2022).

110. Rajpurkar, P., Chen, E., Banerjee, O. & Topol, E. J. AI in health and medicine. *Nature medicine* **28,** 31–38 (2022).

111. Drogt, J., Milota, M., Vos, S., Bredenoord, A. & Jongsma, K. Integrating artificial intelligence in pathology: a qualitative interview study of users experiences and expectations. *Modern Pathology* **35,** 1540–1550. ISSN: 0893-3952. https://www.sciencedirect.com/science/article/pii/S0893395222002204 (2022).

112. Baxi, V., Edwards, R., Montalto, M. & Saha, S. Digital pathology and artificial intelligence in translational medicine and clinical practice. *Modern Pathology* **35,** 23–32 (2022).

113. Weigert, M., Schmidt, U., Haase, R., Sugawara, K. & Myers, G. *Star-convex polyhedra for 3D object detection and segmentation in microscopy* in *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (2020), 3666–3673.

114. Mandal, S. & Uhlmann, V. *Splinedist: Automated cell segmentation with spline curves* in *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)* (2021), 1082–1086.

115. Chen, S., Ding, C., Liu, M., Cheng, J. & Tao, D. CPP-net: Context-aware polygon proposal network for nucleus segmentation. *IEEE Transactions on Image Processing* **32,** 980–994 (2023).

116. Graham, S., Vu, Q. D., Raza, S. E. A., Azam, A., Tsang, Y. W., Kwak, J. T. & Rajpoot, N. Hover-net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images. *Medical image analysis* **58,** 101563 (2019).

117. Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J. & Maier-Hein, K. H. nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods* **18,** 203–211 (2021).

118. Mei, Y., Chen, Q., Lensen, A., Xue, B. & Zhang, M. Explainable artificial intelligence by genetic programming: A survey. *IEEE Transactions on Evolutionary Computation* (2022).

119. Parziale, A., Senatore, R., Della Cioppa, A. & Marcelli, A. Cartesian genetic programming for diagnosis of Parkinson disease through handwriting analysis: Performance vs. interpretability issues. *Artificial Intelligence in Medicine* **111,** 101984 (2021).

120. Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

121. Lindenmayer, A. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of theoretical biology* **18,** 280–299 (1968).

122. Darwin, C. *On the Origin of Species by Means of Natural Selection* or the Preservation of Favored Races in the Struggle for Life (Murray, London, 1859).

123. Koza, J. R. Genetic programming as a means for programming computers by natural selection. *Statistics and computing* **4,** 87–112 (1994).

124. Miller, J. F. *et al. An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach* in *Proceedings of the genetic and evolutionary computation conference* **2** (1999), 1135–1142.

125. Lam, S. K., Pitrou, A. & Seibert, S. *Numba: A llvm-based python jit compiler* in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC* (2015), 1–6.

126. Koza, J. R. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems* (Stanford University, Department of Computer Science Stanford, CA, 1990).

127. McKay, R. I., Hoai, N. X., Whigham, P. A., Shan, Y. & O'neill, M. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines* **11,** 365–396 (2010).

128. Spector, L. & Robinson, A. Genetic Programming and Autoconstructive Evolution with the Push Programming Language. en. *Genetic Programming and Evolvable Machines* **3,** 7–40. ISSN: 1573-7632. `https://doi.org/10.1023/A:1014538503543` (2024) (Mar. 2002).

129. Spector, L., Klein, J. & Keijzer, M. *The Push3 execution stack and the evolution of control* in *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation* (Association for Computing Machinery, Washington DC, USA, 2005), 1689–1696. ISBN: 1595930108. https://doi.org/10.1145/1068009.1068292.

130. Banzhaf, W. *Genetic programming for pedestrians* in *ICGA* (1993), 628.

131. Brameier, M., Banzhaf, W. & Banzhaf, W. *Linear genetic programming* (Springer, 2007).

132. Alavi, A. H. & Sadrossadat, E. New design equations for estimation of ultimate bearing capacity of shallow foundations resting on rock masses. *Geoscience Frontiers* **7,** 91–99 (2016).

133. Miller, J. F., Thomson, P. & Fogarty, T. Designing electronic circuits using evolutionary algorithms. arithmetic circuits: A case study. *Genetic algorithms and evolution strategies in engineering and computer science,* 105–131 (1997).

134. Tackett, W. A. *Recombination, selection, and the genetic construction of computer programs* PhD thesis (University of Southern California Los Angeles, 1994).

135. Banzhaf, W. & Langdon, W. B. Some considerations on the reason for bloat. *Genetic Programming and Evolvable Machines* **3,** 81–91 (2002).

136. Turner, A. J. & Miller, J. F. *Cartesian genetic programming: Why no bloat?* in *Genetic Programming: 17th European Conference, EuroGP 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers 17* (2014), 222–233.

137. Clegg, J., Walker, J. & Miller, J. *A new crossover technique for Cartesian genetic programming* in (July 2007), 1580–1587.

138. Kalkreuth, R., Rudolph, G. & Droschinsky, A. *A new subgraph crossover for cartesian genetic programming* in *Genetic Programming: 20th European Conference, EuroGP 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings 20* (2017), 294–310.

139. Wilson, D. G., Miller, J. F., Cussat-Blanc, S. & Luga, H. Positional Cartesian Genetic Programming. *ArXiv* **abs/1810.04119** (2018).

140. Miller, J. & Harding, S. *GECCO 2010 tutorial: Cartesian genetic programming* in (July 2010), 2927–2948.

141. Harding, S., Graziano, V., Leitner, J. & Schmidhuber, J. *Mt-cgp: Mixed type cartesian genetic programming* in *Proceedings of the 14th annual conference on Genetic and evolutionary computation* (2012), 751–758.

142. Leitner, J., Harding, S., Förster, A. & Schmidhuber, J. *Mars Terrain Image Classification using Cartesian Genetic Programming* in (Sept. 2012).

143. Harding, S., Leitner, J. & Schmidhuber, J. in, 31–44 (Mar. 2013). ISBN: 978-1-4614-6845-5.

144. Biau, J., Wilson, D., Cussat-Blanc, S. & Luga, H. *Improving Image Filters with Cartesian Genetic Programming.* in *IJCCI* (2021), 17–27.

145. Paris, P. C. D., Pedrino, E. C. & Nicoletti, M. Automatic learning of image filters using Cartesian genetic programming. *Integrated Computer-Aided Engineering* **22,** 135–151 (2015).

146. Turner, A. J. & Miller, J. F. *Recurrent cartesian genetic programming* in *Parallel Problem Solving from Nature–PPSN XIII: 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings 13* (2014), 476–486.

147. Turner, A. J. & Miller, J. F. *Recurrent Cartesian genetic programming applied to famous mathematical sequences* in *Proceedings of the Seventh York Doctoral Symposium on Computer Science & Electronics* (2014), 37–46.

148. Turner, A. J. & Miller, J. F. *Recurrent Cartesian genetic programming applied to series forecasting* in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation* (2015), 1499–1500.

149. Kalkreuth, R. *Two New Mutation Techniques for Cartesian Genetic Programming.* in *IJCCI* (2019), 82–92.

150. Fang, W. & Gu, M. FMCGP: frameshift mutation cartesian genetic programming. *Complex & Intelligent Systems* **7,** 1195–1206 (2021).

151. Freese, E. The specific mutagenic effect of base analogues on phage T4. *Journal of Molecular Biology* **1,** 87–105 (1959).

152. Crick, F., Barnett, L., Brenner, S., Watts-Tobin, R. J., *et al.* General nature of the genetic code for proteins (1961).

153. Miller, J. Cartesian genetic programming: its status and future. *Genetic Programming and Evolvable Machines* **21** (June 2020).

154. Goldman, B. & Punch, W. *Reducing Wasted Evaluations in Cartesian Genetic Programming* in. **7831** (Apr. 2013), 61–72. ISBN: 978-3-642-37206-3.

155. Harding, S. L., Miller, J. F. & Banzhaf, W. *Self-modifying cartesian genetic programming* in *Proceedings of the 9th annual conference on Genetic and evolutionary computation* (2007), 1021–1028.

156. Yazdani, S. & Shanbehzadeh, J. Balanced Cartesian Genetic Programming via migration and opposition-based learning: application to symbolic regression. *Genetic Programming and Evolvable Machines* **16,** 133–150 (2015).

157. Simon, D. Biogeography-based optimization. *IEEE transactions on evolutionary computation* **12,** 702–713 (2008).

158. Izzo, D., Biscani, F. & Mereta, A. *Differentiable genetic programming* in *Genetic Programming: 20th European Conference, EuroGP 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings 20* (2017), 35–51.

159. Knezevic, K., Picek, S. & Miller, J. F. *Amplitude-oriented mixed-type CGP classification* in *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2017), 1415–1418.

160. Sekanina, L. *Image Filter Design with Evolvable Hardware* in. **2279** (Mar. 2002), 21–27. ISBN: 978-3-540-43432-0.

161. Martinek, T. & Sekanina, L. *An Evolvable Image Filter: Experimental Evaluation of a Complete Hardware Implementation in FPGA* in (Sept. 2005), 76–85. ISBN: 978-3-540-28736-0.

162. Sekanina, L. & Martinek, T. Evolving Image Operators Directly in Hardware. *Eurasip Book Series on Signal Processing and Communications* **8** (Jan. 2008).

163. Harding, S. *Evolution of image filters on graphics processor units using Cartesian Genetic Programming* in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)* (2008), 1921–1928.

164. Sekanina, L., Harding, S., Banzhaf, W. & Kowaliw, T. in, 181–215 (Sept. 2011). ISBN: 978-3-642-17309-7.

165. Leitner, J., Harding, S., Frank, M., Förster, A. & Schmidhuber, J. *Humanoid Learns to Detect Its Own Hands* in (June 2013).

166. Leitner, J., Förster, A. & Schmidhuber, J. *Improving robot vision models for object detection through interaction* in *2014 International Joint Conference on Neural Networks (IJCNN)* (2014), 3355–3362.

167. Barisoni, L., Lafata, K. J., Hewitt, S. M., Madabhushi, A. & Balis, U. G. Digital pathology and computational image analysis in nephropathology. *Nature Reviews Nephrology* **16,** 669–685 (2020).

168. Esteva, A., Chou, K., Yeung, S., Naik, N., Madani, A., Mottaghi, A., Liu, Y., Topol, E., Dean, J. & Socher, R. Deep learning-enabled medical computer vision. *NPJ digital medicine* **4,** 5 (2021).

169. Deng, S., Zhang, X., Yan, W., Chang, E. I.-C., Fan, Y., Lai, M. & Xu, Y. Deep learning in digital pathology image analysis: a survey. *Frontiers of medicine* **14,** 470–487 (2020).

170. Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence* **1,** 206–215 (2019).

171. Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S. & Yang, G.-Z. XAI—Explainable artificial intelligence. *Science robotics* **4,** eaay7120 (2019).

172. Banzhaf, W., Nordin, P., Keller, R. E. & Francone, F. D. *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications* (Morgan Kaufmann Publishers Inc., 1998).

173. Miller, J. F. *Cartesian genetic programming* 346. ISBN: 9783642173097. `http://dx.doi.org/10.1007/978-3-642-17310-3` (Springer, Berlin, Germany, 2011).

174. Miller, J. F., Job, D. & Vassilev, V. K. Principles in the evolutionary design of digital circuits—Part I. *Genetic programming and evolvable machines* **1,** 7–35 (2000).

175. Ahmad, A. M., Khan, G. M., Mahmud, S. A. & Miller, J. F. *Breast cancer detection using cartesian genetic programming evolved artificial neural networks* in *Proceedings of the 14th annual conference on Genetic and evolutionary computation* (2012), 1031–1038.

176. Suganuma, M., Shirakawa, S. & Nagao, T. Designing convolutional neural network architectures using cartesian genetic programming. *Deep Neural Evolution: Deep Learning with Evolutionary Computation,* 185–208 (2020).

177. Serge, B. *IMAGE SEGMENTATION AND MATHEMATICAL MORPHOLOGY* Generic.

178. Yuen, H., Princen, J., Illingworth, J. & Kittler, J. Comparative study of Hough transform methods for circle finding. *Image and vision computing* **8,** 71–77 (1990).

179. Yang, L., Ghosh, R. P., Franklin, J. M., Chen, S., You, C., Narayan, R. R., Melcher, M. L. & Liphardt, J. T. NuSeT: A deep learning tool for reliably separating and analyzing crowded cells. *PLoS computational biology* **16,** e1008193 (2020).

180. Bertrand, F., Müller, S., Roh, K.-H., Laurent, C., Dupré, L. & Valitutti, S. An initial and rapid step of lytic granule secretion precedes microtubule organizing center polarization at the cytotoxic T lymphocyte/target cell synapse. *Proceedings of the National Academy of Sciences* **110,** 6073–6078 (2013).

181. Stinchcombe, J. C., Bossi, G., Booth, S. & Griffiths, G. M. The immunological synapse of CTL contains a secretory domain and membrane bridges. *Immunity* **15,** 751–761 (2001).

182. Chang, H.-F., Schirra, C., Ninov, M., Hahn, U., Ravichandran, K., Krause, E., Becherer, U., Bálint, Š., Harkiolaki, M., Urlaub, H., *et al.* Identification of distinct cytotoxic granules as the origin of supramolecular attack particles in T lymphocytes. *Nature communications* **13,** 1029 (2022).

183. Bálint, Š., Müller, S., Fischer, R., Kessler, B., Harkiolaki, M., Valitutti, S. & Dustin, M. Supramolecular attack particles are autonomous killing entities released from cytotoxic T cells. *Science* **368,** 897–901 (2020).

184. McInnes, L., Healy, J. & Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).

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

186. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., *et al.* Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* **12,** 2825–2830 (2011).

187. Shamai, G., Livne, A., Polónia, A., Sabo, E., Cretu, A., Bar-Sela, G. & Kimmel, R. Deep learning-based image analysis predicts PD-L1 status from H&E-stained histopathology images in breast cancer. *Nature Communications* **13,** 6753 (2022).

188. Koelzer, V. H., Gisler, A., Hanhart, J. C., Griss, J., Wagner, S. N., Willi, N., Cathomas, G., Sachs, M., Kempf, W., Thommen, D. S., *et al.* Digital image analysis improves precision of PD-L1 scoring in cutaneous melanoma. *Histopathology* **73,** 397–406 (2018).

189. *Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL LAYING DOWN HARMONISED RULES ON ARTIFICIAL INTELLIGENCE (ARTIFICIAL INTELLIGENCE ACT) https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206* Report (European Commission, 2021).

190. OpenAI. *Introducing ChatGPT* Web Page. 2023. `https://chat.openai.com/`.

191. OpenAI. GPT-4 Technical Report. *arXiv:2303.08774* (2023).

192. Allal, L. B., Li, R., Kocetkov, D., Mou, C., Akiki, C., Ferrandis, C. M., Muennighoff, N., Mishra, M., Gu, A., Dey, M., *et al.* SantaCoder: don't reach for the stars! *arXiv preprint arXiv:2301.03988* (2023).

193. Shrivastava, D., Larochelle, H. & Tarlow, D. *Repository-level prompt generation for large language models of code* in *International Conference on Machine Learning* (2023), 31693–31715.

194. Le, T. T., Fu, W. & Moore, J. H. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics* **36,** 250–256 (2020).

195. Shen, Y., Song, K., Tan, X., Li, D., Lu, W. & Zhuang, Y. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580* (2023).

196. Blank, J. & Deb, K. Pymoo: Multi-objective optimization in python. *Ieee access* **8,** 89497–89509 (2020).

197. Deb, K. in *Multi-objective evolutionary optimisation for product design and manufacturing* 3–34 (Springer, 2011).

198. Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* **6,** 182–197 (2002).

199. Gerules, G. & Janikow, C. *A survey of modularity in genetic programming* in *2016 IEEE Congress on Evolutionary Computation (CEC)* (2016), 5034–5043.

200. Zuba-Surma, E. K. & Ratajczak, M. Z. Analytical capabilities of the ImageStream cytometer. *Methods in cell biology* **102,** 207–230 (2011).

201. Van Rossum, G., Drake, F. L., *et al. Python reference manual* (Centrum voor Wiskunde en Informatica Amsterdam, 1995).

202. Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., *et al.* Array programming with NumPy. *Nature* **585,** 357–362 (2020).

203. Van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E. & Yu, T. scikit-image: image processing in Python. *PeerJ* **2,** e453 (2014).

204. Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in science & engineering* **9,** 90–95 (2007).

205. Waskom, M. L. Seaborn: statistical data visualization. *Journal of Open Source Software* **6,** 3021 (2021).

206. Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C., Saalfeld, S., Schmid, B., *et al.* Fiji: an open-source platform for biological-image analysis. *Nature methods* **9,** 676–682 (2012).

207. Goldman, B. W. & Punch, W. F. *Length bias and search limitations in Cartesian genetic programming* in *Proceedings of the 15th annual conference on Genetic and Evolutionary Computation* (2013), 933–940.

208. Goode, A., Gilbert, B., Harkes, J., Jukic, D. & Satyanarayanan, M. OpenSlide: A vendor-neutral software foundation for digital pathology. *Journal of pathology informatics* **4,** 27 (2013).

209. Montana, D. J. Strongly typed genetic programming. *Evolutionary computation* **3,** 199–230 (1995).

210. Jia, B. & Ebner, M. *A strongly typed GP-based video game player* in *2015 IEEE Conference on Computational Intelligence and Games (CIG)* (2015), 299–305.

211. Barber, C. B., Dobkin, D. P. & Huhdanpaa, H. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)* **22,** 469–483 (1996).

212. Kuepper, J., Erbsen, A., Gross, J., Conoly, O., Sun, C., Tian, S., Wu, D., Chlipala, A., Chuengsatiansup, C., Genkin, D., *et al.* CryptOpt: Verified Compilation with Randomized Program Search for Cryptographic Primitives. *Proceedings of the ACM on Programming Languages* **7,** 1268–1292 (2023).

213. Real, E., Liang, C., So, D. & Le, Q. *Automl-zero: Evolving machine learning algorithms from scratch* in *International conference on machine learning* (2020), 8007–8019.

214. Helmuth, T. & Kelly, P. *PSB2: The Second Program Synthesis Benchmark Suite* arXiv:2106.06086 [cs]. June 2021. `http://arxiv.org/abs/2106.06086` (2024).

215. Sobania, D., Schweim, D. & Rothlauf, F. A Comprehensive Survey on Program Synthesis With Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* **27.** Conference Name: IEEE Transactions on Evolutionary Computation, 82–97. ISSN: 1941-0026. `https://ieeexplore.ieee.org/document/9743417` (2024) (Feb. 2023).

216. Gulwani, S., Polozov, O. & Singh, R. Program Synthesis. English. *Foundations and Trends® in Programming Languages* **4.** Publisher: Now Publishers, Inc., 1–119. ISSN: 2325-1107, 2325-1131. `https://www.nowpublishers.com/article/Details/PGL-010` (2024) (July 2017).

217. Cava, W. G. L., Orzechowski, P., Burlacu, B., de França, F. O., Virgolin, M., Jin, Y., Kommenda, M. & Moore, J. H. Contemporary Symbolic Regression Methods and their Relative Performance. *CoRR* **abs/2107.14351.** arXiv: `2107.14351`. `https://arxiv.org/abs/2107.14351` (2021).

218. Orzechowski, P., La Cava, W. & Moore, J. H. *Where are we now? a large benchmark study of recent symbolic regression methods* in *Proceedings of the Genetic and Evolutionary Computation Conference* (Association for Computing Machinery, Kyoto, Japan, 2018), 1183–1190. ISBN: 9781450356183. `https://doi.org/10.1145/3205455.3205539`.

219. Helmuth, T. & Spector, L. *General Program Synthesis Benchmark Suite* in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (Association for Computing Machinery, Madrid, Spain, 2015), 1039–1046. ISBN: 9781450334723. `https://doi.org/10.1145/2739480.2754769`.

220. Jabeen, H. & Baig, A. R. Two layered Genetic Programming for mixed-attribute data classification. *Applied Soft Computing* **12,** 416–422. ISSN: 1568-4946. https://www.sciencedirect.com/science/article/pii/S1568494611003127 (2012).

221. Macret, M. & Pasquier, P. *Automatic design of sound synthesizers as pure data patches using coevolutionary mixed-typed cartesian genetic programming* in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation* (Association for Computing Machinery, New York, NY, USA, July 2014), 309–316. ISBN: 978-1-4503-2662-9. https://doi.org/10.1145/2576768.2598303 (2024).

222. Bezanson, J., Edelman, A., Karpinski, S. & Shah, V. Julia: A Fresh Approach to Numerical Computing (July 2015).

223. Chollet, F. *et al.* Keras: The python deep learning library. *Astrophysics source code library,* ascl–1806 (2018).

224. Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S. & Zhang, Q. *JAX: composable transformations of Python+NumPy programs* version 0.3.13. 2018. http://github.com/google/jax.

225. Hennigan, T., Cai, T., Norman, T., Martens, L. & Babuschkin, I. *Haiku: Sonnet for JAX* version 0.0.10. 2020. http://github.com/deepmind/dm-haiku.

226. Husa, J. & Kalkreuth, R. *A comparative study on crossover in cartesian genetic programming* in *Genetic Programming: 21st European Conference, EuroGP 2018, Parma, Italy, April 4-6, 2018, Proceedings 21* (2018), 203–219.

227. Khan, M. M., Khan, G. M. & Miller, J. F. *Evolution of neural networks using cartesian genetic programming* in *IEEE congress on evolutionary computation* (2010), 1–8.

228. Turner, A. J. & Miller, J. F. *Cartesian genetic programming encoded artificial neural networks: a comparison using three benchmarks* in *Proceedings of the 15th annual conference on Genetic and evolutionary computation* (2013), 1005–1012.

229. Botvinnik, M. M. in *Computers, Chess and Long-Range Planning* 11–25 (Springer US, New York, NY, 1970). ISBN: 978-1-4684-6245-6. https://doi.org/10.1007/978-1-4684-6245-6_3.

230. Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S. & Colton, S. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* **4,** 1–43 (2012).

231. Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., *et al.* Mastering the game of Go with deep neural networks and tree search. *nature* **529,** 484–489 (2016).

232. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., *et al.* A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **362,** 1140–1144 (2018).

233. Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* **596,** 583–589 (2021).

234. Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., Novikov, A., R Ruiz, F. J., Schrittwieser, J., Swirszcz, G., *et al.* Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature* **610,** 47–53 (2022).

235. Mankowitz, D. J., Michi, A., Zhernov, A., Gelmi, M., Selvi, M., Paduraru, C., Leurent, E., Iqbal, S., Lespiau, J.-B., Ahern, A., *et al.* Faster sorting algorithms discovered using deep reinforcement learning. *Nature* **618,** 257–263 (2023).

236. Hamilton, W. L., Ying, R. & Leskovec, J. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).

237. Ibarz, B., Kurin, V., Papamakarios, G., Nikiforou, K., Bennani, M., Csordás, R., Dudzik, A. J., Bošnjak, M., Vitvitskyi, A., Rubanova, Y., *et al. A generalist neural algorithmic learner* in *Learning on Graphs Conference* (2022), 2–1.

238. Kuznetsov, A. & Urdiales, H. Spelling correction with denoising transformer. *arXiv preprint arXiv:2105.05977* (2021).

239. Li, W., Luo, H., Lin, Z., Zhang, C., Lu, Z. & Ye, D. A survey on transformers in reinforcement learning. *arXiv preprint arXiv:2301.03044* (2023).

240. Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* **29,** 141–142 (2012).

241. Syrykh, C., Abreu, A., Amara, N., Siegfried, A., Maisongrosse, V., Frenois, F. X., Martin, L., Rossi, C., Laurent, C. & Brousset, P. Accurate diagnosis of lymphoma on whole-slide histopathology images using deep learning. *NPJ digital medicine* **3,** 63 (2020).

242. Biau, J. *Nouvelles approches évolutionnaires pour des algorithmes interprétables de vision par ordinateur pour les systèmes embarqués* Theses (Université Toulouse le Mirail - Toulouse II, June 2022). `https://theses.hal.science/tel-04088467`.