

A Hybrid Neuroevolutionary Approach to the Design of Convolutional Neural Networks for 2D and 3D Medical Image Segmentation

Nivedha Ramesh

**A Thesis
in
The Department
of
Electrical And Computer Engineering**

**Presented in Partial Fulfillment of the Requirements
for the Degree of
Master of Applied Science (Electrical And Computer Engineering) at
Concordia University
Montréal, Québec, Canada**

February 2024

© Nivedha Ramesh, 2024

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Nivedha Ramesh**
Entitled: **A Hybrid Neuroevolutionary Approach to the Design of Convolutional Neural Networks for 2D and 3D Medical Image Segmentation**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical And Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

Dr. W. Hamou-Lhadj Chair

Dr. D. Goswami External Examiner

Dr. W. Hamou-Lhadj Examiner

Dr. N. Kharm Supervisor

Approved by _____
Dr. Yousef R. Shayan, Chair
Department of Electrical And Computer Engineering

_____ 2024

Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

A Hybrid Neuroevolutionary Approach to the Design of Convolutional Neural Networks for 2D and 3D Medical Image Segmentation

Nivedha Ramesh

This thesis highlights the development and evaluation of a hybrid neuroevolutionary approach for designing Convolutional Neural Networks (CNNs) for image classification and segmentation tasks. We integrate Cartesian Genetic Programming (CGP) with Novelty Search and Simulated Annealing algorithms to optimize the CNN architectures efficiently.

The challenge lies in reducing the computational demands and inefficiencies of traditional Neural Architecture Search (NAS) techniques. To address this, a flexible framework based on CGP is utilized for evolving network architectures. Novelty Search facilitates the exploration of varied architectural landscapes, promoting diversity of solutions. Simulated Annealing further refines these solutions, optimizing the balance between exploring new possibilities and exploiting known good solutions within the search space.

Our experiments, conducted on benchmark datasets such as DRIVE and MSD, demonstrate the method's effectiveness in generating competitive segmentation models. On the DRIVE dataset, our models achieved Dice Similarity Coefficients (DSC) of 0.828 and 0.814, and Intersection over Union (IoU) scores of 0.716 and 0.736, respectively. For the MSD dataset, our models exhibited DSC scores up to 0.924 for the Heart task, showcasing the potential of our method in handling complex segmentation challenges across different medical imaging modalities.

The significance of this research lies in its hybrid approach that efficiently navigates the search space for CNN architectures, thus reducing number of fitness evaluations while achieving near state of art performance. Future work will explore enhancing the algorithm's effectiveness through advanced data preprocessing techniques, and the exploration of more complex network layers. Our findings highlight the potential of evolutionary algorithms and local search in advancing automated CNN design for medical image segmentation, offering a promising direction for future research in the field.

Acknowledgments

I would like to express my sincere gratitude to Dr. Nawwaf Kharma, my research supervisor, for his unwavering support, invaluable guidance, and endless patience throughout the course of this research. His mentorship has been instrumental in shaping the direction of this work.

I am also deeply appreciative of the contributions made by Tabish Ashfaq, for lending his expertise and dedication to the experimental aspects of this study. His insightful feedback and collaborative spirit have been truly invaluable.

Finally, my deepest appreciation goes to my parents, brother and friends for their unwavering support and encouragement throughout my academic journey. Their love and belief in me have been a constant source of inspiration.

This research would not have been possible without the collective efforts and encouragement of these individuals, and for that, I am profoundly thankful.

Contents

List of Figures	vii
List of Tables	x
1 Introduction	1
1.1 Motivation	1
1.2 Problem Statement	2
1.3 Contribution	4
1.4 Thesis Aims and Outlines	4
2 Literature Review	6
3 Hybridized Neuroevolution for Image Classification	10
3.1 Methodology	11
3.1.1 Genetic Representation	14
3.1.2 Phenotypic Representation	15
3.1.3 Evolutionary Algorithm	16
3.1.4 Diversity Measures and Novelty Search	22
3.1.5 Local Optimization	23
3.2 Dataset	28
3.3 Experimental Setting	28
3.4 Results and Discussion	28
3.4.1 Recognition Error	28
3.4.2 Number of Fitness Evaluations	30
3.4.3 Comparison to related works	30
3.5 Conclusions and Future Work	32
4 Hybridized Neuroevolution for 2D and 3D Image Segmentation	34
4.1 Methodology	34
4.1.1 Genotypic and Phenotypic Representation	36
4.1.2 Evolutionary Algorithm	37
4.1.3 Novelty Computation using GED	42
4.1.4 Vector Representation	43
4.1.5 Local Optimization	46

4.2	Dataset	49
4.2.1	DRIVE Dataset (Digital Retinal Images for Vessel Extraction) . . .	49
4.2.2	Medical Segmentation Decathlon (MSD) Dataset	49
4.3	Experimental Setting	50
4.4	Results and Discussion	52
4.4.1	Segmentation Results	52
4.4.2	Novelty Optimization of the Proposed Algorithm	54
4.4.3	Number of Fitness Evaluations	55
4.4.4	Comparison with related works	56
4.5	Conclusions and Future Work	60
5	Conclusions and Future Work	61
	Bibliography	63

List of Figures

Figure 3.1	Illustration of the three distinct stages of our proposed hybrid algorithm - rich initialization, evolution and local optimization.	12
Figure 3.2	Representation of the individual's genotype as a two-dimensional matrix, where each row represents a specific layer within the CNN architecture. These rows comprise three columns each, denoting the type of layer function alongside two node IDs to which the layer is connected to. Notably, the row highlighted in gray, featuring an Average Pooling layer linked to nodes 1 and 4, signifies an inactive node. This particular layer is not expressed in the phenotype, which is the Keras CNN model.	13
Figure 3.3	An illustration of an example genotype and it's equivalent phenotype. We can see that each cell in the genotype contains a function id and two input connection id's.	15
Figure 3.4	Overview of an evolutionary algorithm	17
Figure 3.5	Mean novelty vs generations: evolutionary algorithm (top) and evolutionary algorithm with Novelty Search (bottom). In each case, the vertical line indicates the generation selected for local optimization.	25
Figure 3.6	Solutions from the sampled generations of highest mean novelty from (a) Population A and (b) Population B are sorted in ranks based on their novelty and fitness score by the NSGA II algorithm. The solutions in pink belong to rank 1, solutions in blue belong to rank 2 and solutions in yellow belong to further lower ranks.	26
Figure 3.7	Variation of fitness over number of fitness evaluations during the evolutionary process for Individual 1 from Populations A through D. The shaded red region represents the number of fitness evaluations during the rich initialization of the population and the green region represents the number of fitness evaluations during the evolutionary process until sampling is done.	31
Figure 4.1	Three distinct stages of our proposed hybrid algorithm - rich initialization, evolutionary phase and local optimization.	35
Figure 4.2	An illustration of an example genotype and it's equivalent phenotype. We can see that each cell in the genotype contains a function id and two input connection id's.	36
Figure 4.3	Overview of the evolution process	39

Figure 4.4 Runtime comparison of SimGNN with other benchmarked methods of GED computation over three datasets.	43
Figure 4.5 Mean novelty variation across generations during evolution of solutions for DRIVE dataset segmentation, with novelty scores on the y-axis and number of generations on the x-axis. The population with the highest mean novelty was sampled at the 28th generation, as indicated by the dotted line.	44
Figure 4.6 Mean novelty variation across generations during evolution of solutions for MSD Dataset Tasks, with novelty scores on the y-axis and number of generations on the x-axis. The blue curve represents the Hippocampus Model, and the population with highest mean novelty was sampled at the 30th generation. The green curve represents the heart model, with the population of highest mean novelty found at the 28th generation, while the purple curve represents the liver model with the highest mean novelty population sampled at the 14th generation. The pink curve represents the lung model with the population sampled from the 60th generation. Dotted Lines mark the discovery of generations with highest mean novelty.	45
Figure 4.7 Population for the DRIVE segmentation task as ranked by the NSGA II algorithm, based on their fitness and novelty score. The novelty score is presented on the y-axis while the fitness is presented on the x-axis. Solutions in blue represent Rank 1, red represent Rank 2 and yellow represents further lower ranks.	46
Figure 4.8 Population for the MSD Hippocampus segmentation task as ranked by the NSGA II algorithm, based on their fitness and novelty score. The novelty score is presented on the y-axis while the fitness is presented on the x-axis. Solutions in blue represent Rank 1, red represent Rank 2 and yellow represents further lower ranks.	47
Figure 4.9 Population for the MSD Heart segmentation task as ranked by the NSGA II algorithm, based on their fitness and novelty score. The novelty score is presented on the y-axis while the fitness is presented on the x-axis. Solutions in blue represent Rank 1, red represent Rank 2 and yellow represents further lower ranks.	47
Figure 4.10 Population for the MSD Lung segmentation task as ranked by the NSGA II algorithm, based on their fitness and novelty score. The novelty score is presented on the y-axis while the fitness is presented on the x-axis. Solutions in blue represent Rank 1, red represent Rank 2 and yellow represents further lower ranks.	48
Figure 4.11 Population for the MSD Liver segmentation task as ranked by the NSGA II algorithm, based on their fitness and novelty score. The novelty score is presented on the y-axis while the fitness is presented on the x-axis. Solutions in blue represent Rank 1, red represent Rank 2 and yellow represents further lower ranks.	48

Figure 4.12 Visualization of the (i) original slice of a input sample on the left with it's (ii) respective segmentation mask overlay on the right for the MSD dataset prior to preprocessing.	51
Figure 4.13 Slices from the (a) Heart, (b) Lung, (c) Liver task of the MSD dataset visualized after the preprocessing pipeline	52
Figure 4.14 Visualization of the Input slice(left) and predicted mask(right) for the different segmentation tasks	53
Figure 4.15 Fitness variation across iterations in the simulated annealing algorithm during optimization of the model for DRIVE dataset segmentation task, with dotted line marking the discovery of the optimal model at the 86th iteration.	56
Figure 4.16 Fitness variation across iterations in the simulated annealing algorithm during optimization of the models for MSD Dataset Tasks, with Dotted Lines Marking the Discovery of Optimal Models.The blue curve represents the Hippocampus Model, the red curve the heart model, the cyan curve the liver model and the orange curve the lung model.	57

List of Tables

Table 3.1	Node functions present in the CGP function set. F: number of filters (output channels), K: kernel size, N: number of units in a dense layer. . . .	11
Table 3.2	Summary of Parameters used for the evolutionary algorithm	21
Table 3.3	Summary of Parameters used for the Local Optimization Algorithm .	26
Table 3.4	Summary of the four fittest individuals obtained from each population. Individuals are sorted by test error. The best results are highlighted in bold.	29
Table 3.5	Comparison of the recognition error (%) on the MNIST test dataset to related works.	32
Table 4.1	CGP function set and range of different parameters to construct the genotype.	37
Table 4.2	Summary of Parameters used for the evolutionary algorithm	42
Table 4.3	Original and Augmented Dimensions for 2D slices of different tasks in the MSD dataset. The Hippocampus task has varying dimensions for every sample and hence the largest dimension is reported here.	50
Table 4.4	Performance of the top two models for each of the segmentation task. The best results for each task is shown in bold formatting.	54
Table 4.5	Number of fitness computations incurred in obtaining the best performing model for each of the segmentation tasks. F1 indicates the number of computations for the evolutionary phase and F2 indicates the number of computations for the local optimization phase.	55
Table 4.6	Results (%) of Retinal Vessel Segmentation on DRIVE dataset. The accuracy, dice score coefficient and the intersection of union metrics for all the benchmarked models are tabulated in decreasing order of DSC. The best results are in bold.	57
Table 4.7	Comparison of the DSC metric different tasks of the MSD dataset of the current state of the art models. Some of the models do not report the results for all the four tasks and are hence left blank	59

Chapter 1

Introduction

In the realm of computer science and statistics, machine learning (ML) stands as a cornerstone, distinguished by its ability to perform computational tasks through algorithms that learn from patterns in data rather than relying on explicit programming. These algorithms, central to automating decision-making, are foundational in the development of machine learning projects. Neural networks have been a foundational element in the field of artificial intelligence (AI) and machine learning, evolving significantly since their inception. The journey began with the perceptron, invented by [Rosenblatt \(1957\)](#) which was a simple model designed to mimic the decision-making process of the human brain but was limited to linear classification tasks. The introduction of the backpropagation algorithm by [Rumelhart, Hinton, and Williams \(1986a\)](#) allowed neural networks to adjust their internal parameters accurately. This method was crucial for training deep neural networks, leading to the development of more complex architectures. [LeCun, Bengio, and Hinton \(2015\)](#) developed Convolutional Neural Networks (CNNs), inspired by the visual cortex of animals which became fundamental in processing visual data, revolutionizing fields like image recognition. These networks, especially Long Short-Term Memory (LSTM) networks, were designed to handle sequences of data, making them ideal for tasks like speech recognition and natural language processing (NLP). The last decade has seen an explosion in deep learning, with neural networks becoming deeper and more complex. This era has been marked by the development of new architectures like Generative Adversarial Networks (GANs) and Transformers, significantly improving performance across a wide range of tasks.

1.1 Motivation

CNNs have shown exceptional results in handling tasks related to computer vision and image classification [Krizhevsky, Sutskever, and Hinton \(2012\)](#). Currently, they find application in diverse fields such as image processing [Ren, He, Girshick, and Sun \(2016\)](#), healthcare [Joshi and Singh \(2020\)](#), agriculture [Joshi, Mishra, Srivastav, and Goel \(2021\)](#), and cyber security [Ghillani \(2022\)](#). However, the effectiveness of a CNN largely hinges on the architecture's complexity, the training data it's exposed to, and how its hyperparameters

are chosen [Simonyan and Zisserman \(2014\)](#). The architecture of a CNN is composed of convolutional, pooling, and fully connected layers arranged in sequence. The design process of a CNN necessitates the network designer to undertake multiple critical decisions involving, selecting the number and types of layers, determining their sequential arrangement, and specifying the hyperparameters for each layer which include the dimensions, stride, and the quantity of receptive fields within the convolutional layers. The vast choices available in the design process renders the task exceedingly complex, making the thorough manual exploration of all possible configurations impractical.

While automated or semi-automated neural network design methods have been developed [Pinto, Doukhan, Dicarolo, and Cox \(2009\)](#) [Bergstra, Yamins, and Cox \(2012\)](#) [Domhan, Springenberg, and Hutter \(2015\)](#), the innovation of new CNN architectures or network design aspects predominantly depends on the theoretical knowledge and empirical insights of researchers. To streamline the design process and enhance the performance of neural networks, Neural Architecture Search (NAS) has emerged as a powerful solution. NAS automates the labor-intensive and time-consuming task of creating neural network designs, saving time and effort. Additionally, it has the potential to enhance the performance of neural networks by exploring different designs to find ones that outperform those created by humans.

1.2 Problem Statement

The evolution of neural networks has been characterized by a continual push towards more sophisticated and capable models, driven by advancements in algorithms, computing power, and data availability. NAS represents a cutting-edge frontier in this evolution, promising to accelerate the discovery of optimal neural network architectures and reduce the need for manual experimentation. NAS techniques are mainly categorized into two approaches: Evolutionary Algorithms (EA) based [Miikkulainen et al. \(2017\)](#), [Real et al. \(2017\)](#), [Xie and Yuille \(2017\)](#) and Reinforcement Learning (RL) based [Baker, Gupta, Naik, and Raskar \(2017\)](#), [Cai, Chen, Zhang, Yu, and Wang \(2018\)](#), [Zoph and Le \(2017\)](#), [Zoph, Vasudevan, Shlens, and Le \(2018\)](#). In the EA approach, each neural network configuration is represented as a string. The search process involves random mutations and recombinations of these strings. Subsequently, each model is trained and assessed on a validation set, with top-performing models producing "offsprings". On the other hand, the RL approach involves an agent executing a series of actions to determine the network's structure. After training, the model's performance on a validation set is used as feedback to refine the RL controller's decisions.

In assessing the weaknesses of EA and RL based methods in neural architecture search, it's crucial to understand the limitations inherent in each approach. For EAs, a primary concern is their computational intensity. The process of evaluating numerous architectures, demands substantial computational resources, which can be a limiting factor in resource-constrained research environments [De Jong \(2006\)](#), [Eiben and Smith \(2003\)](#). Additionally, the randomness in mutations and crossovers, while beneficial for exploring diverse solutions, may lead to a less directed and potentially inefficient search process.

On the other hand, RL-based methods, despite their more focused search trajectory, suffer from issues related to sample efficiency. These methods often require a large number of trials to effectively learn and optimize network architectures, making them time-consuming and computationally expensive. For instance, one RL-based method required training and evaluating 20,000 neural networks using 500 P100 GPUs over a period of four days. Furthermore, there is a risk of converging to local optima, particularly if the reward system and exploration strategies are not carefully designed [Sutton and Barto \(2018\)](#), [Silver et al. \(2016\)](#). Both methodologies, despite their proven capabilities in discovering high-performing architectures, present significant challenges that necessitate careful consideration, especially in the context of limited computational resources and the need for efficient search strategies in NAS.

Given these considerations, our focus shifts towards neuroevolution, particularly in the realm of medical segmentation tasks, and the application of CNNs within healthcare. This area demands our attention due to the critical requirement for high precision in medical image segmentation, especially concerning Regions of Interest (ROIs) like abnormalities and organs. The need for exact delineation of lesion boundaries and accurate measurement of lesion size is crucial, as even minor inaccuracies can significantly impact clinical practices, degrade the clinical environment experience, and increase risks associated with computer-aided diagnosis. The exhaustive and time-consuming nature of manually outlining lesions across various imaging modalities, coupled with the influence of clinician subjectivity and expertise on outcomes, highlights the necessity for developing advanced, reliable techniques for the efficient and precise segmentation of lesions or organs in medical images.

The unique challenges posed by medical imaging, such as the presence of various types of noise, artifacts, and variability in images, necessitate complex network structures incorporating mechanisms like attention to improve segmentation accuracy. Neuroevolution, integrating evolutionary computation with neural networks, presents a promising avenue for autonomously determining network structures. CNNs, with their sophisticated layers of linear and nonlinear processing units, excel at learning detailed representations for complex problems. Consequently, CNN architectures such as UNet [Ronneberger, Fischer, and Brox \(2015\)](#), UNet++ [Zhou, Rahman Siddiquee, Tajbakhsh, and Liang \(2018\)](#), UNet 3+ [H. Huang et al. \(2020\)](#), and DC-UNet [Lou, Guan, and Loew \(2021\)](#) have emerged as leading methods for image segmentation, renowned for their accuracy. The challenge of manually designing such networks, given the vast array of potential layer configurations and the lack of clear guidelines for network construction, is huge. Additionally, the availability of labeled data for supervised learning is a pivotal factor in training these networks, which itself is a resource-intensive process. Finding the right configurations often requires significant computational efforts, sometimes demanding the use of hundreds of GPU days for adequate training [Baldeon Calisto and Lai-Yuen \(2020\)](#); [Hassanzadeh, Essam, and Sarker \(2020\)](#); [Real, Aggarwal, Huang, and Le \(2019\)](#). This highlights the importance of exploring neuroevolution as a viable approach in this area.

1.3 Contribution

In this research, we explore an enhanced approach to neuroevolution, aiming to refine the process of neural architecture search while fostering the generation of a broad array of diverse solutions. Our investigation begins with the formulation of CNN architectures for image classification, using Cartesian Genetic Programming (CGP) [J. Miller \(2003\)](#), a variant of evolutionary algorithms. This hybrid methodology is designed to identify sub-optimal solutions from distinct regions of the fitness landscape utilizing a multi-objective optimization algorithm that leverages both the calculated novelty metric and fitness value. Subsequently, Simulated Annealing (SA) [Kirkpatrick, Gelatt, and Vecchi \(1983\)](#) is applied for a more focused optimization or exploitation of the local neighborhood of these sub-optimal solutions.

Based on the insights gained from the application of our algorithm in the realm of image classification, we extend our exploration to the domain of biomedical image segmentation. This task entails the pixel-wise classification of images to delineate and map out anomalies such as lesions and tumors within both 2D and 3D datasets. Additionally, we refine our approach to calculating the novelty metric employed during the image classification phase. Transitioning from the initial cosine similarity based metric, we embrace SimGNN [Bai et al. \(2020\)](#), a method that not only enhances accuracy but also improves the efficiency of computing graph edit distances.

The key results of our work can be summarized as below -

- (1) Propose a three stage hybrid neuroevolutionary algorithm that can evolve CNN architectures in image classification and segmentation tasks with performance comparable to the state-of-the-art models
- (2) A diverse set of CNN architectures are evolved by including novelty as an optimization metric during initialization and during switching between exploration and exploitation
- (3) The proposed algorithm is demonstrated to be computationally less intensive (quantified in terms of the number of fitness evaluations incurred here) compared to traditional evolutionary algorithms

Our results indicate that this adapted approach considerably lowers computational demands (quantified in terms of number of fitness evaluations) in comparison to traditional evolutionary methods, without sacrificing model performance. Moreover, the CNN architectures generated through our approach show competitive abilities in recognition tasks against other neuroevolutionary strategies, highlighting the efficacy of our modified neuroevolutionary method in creating efficient neural network architectures.

1.4 Thesis Aims and Outlines

This thesis explores the development of Convolutional Neural Networks (CNNs) through a hybrid neuroevolutionary approach, aimed at mitigating the substantial computational

demands inherent in traditional neuroevolution methods. By integrating Cartesian Genetic Programming (CGPs) with Novelty Search [Lehman and Stanley \(2011\)](#) and Simulated Annealing (SA), this research demonstrates how the proposed hybrid algorithm can reduce the number of computationally intensive fitness evaluations observed in a traditional neuroevolutionary algorithm while yielding CNN architectures that are diverse in structure and competitive in performance with current state-of-the-art models. The structure of this thesis is organized into five main chapters, designed to provide a comprehensive examination of the topic.

Chapter 1 introduces the foundational concepts of neuroevolution and CNNs, emphasising the computational challenges associated with conventional neuroevolutionary methods. This chapter sets the stage by defining the problem statement and the significance of pursuing a hybrid neuroevolutionary approach.

Chapter 2 delves into the existing literature and related work, offering a critical review of previous studies in the domain of neuroevolution and CNN architecture design. This review establishes the context for the proposed method by highlighting the gaps and limitations in current methodologies.

Chapter 3 describes the development of the hybrid algorithm. It details the integration of CGPs, Novelty Search, and SA into a cohesive strategy aimed at optimizing the search for effective CNN architectures. This chapter also includes an evaluation of the algorithm's performance on the MNIST dataset for image classification tasks, serving as a preliminary test of its efficacy.

Chapter 4 expands on the application of the hybrid algorithm beyond simple image classification, focusing on the complex and clinically significant task of medical image segmentation. This chapter presents a comparative analysis of the CNN models designed using the hybrid algorithm against the state-of-art models, showcasing the algorithm's ability to produce high-quality architectures tailored for the segmentation of both 2D and 3D medical images.

Finally, Chapter 5 provides a conclusive summary of the research findings, emphasising the efficiency and effectiveness of the hybrid neuroevolutionary method in designing CNN architectures. This chapter reflects on the implications of the study for the field of neuroevolution and CNN development, proposing directions for future research that could further refine and expand upon the contributions of this work.

Chapter 2

Literature Review

The intersection of evolutionary computation [De Jong \(2006\)](#) and machine learning has given rise to a hybrid field commonly known as Evolutionary Machine Learning (EML). This domain combines methodologies from both areas to enhance solution-seeking processes. EML techniques have found extensive applications across a range of real-world scenarios. The versatility of EML approaches in addressing diverse challenges in these fields highlights their significance and potential impact [Al-Sahaf et al. \(2019\)](#).

Neuroevolution [Floreano, Dürr, and Mattiussi \(2008\)](#), one of the widely used branch of EML, has played a pivotal role in shaping the design of neural networks across various domains and problem types. It introduces capabilities that often surpass those found in traditional gradient-based methods, such as stochastic gradient descent (SGD) [Rumelhart, Hinton, and Williams \(1986b\)](#). This advancement elevates automation beyond merely adjusting weights within predetermined network structures. The enhanced features of neuroevolution encompass the optimization of key hyperparameters, structural components, and potentially the learning rules themselves. This broader scope of optimization signifies a notable shift in the depth and range of automation in neural network design and training [K. Stanley, Clune, Lehman, and Miikkulainen \(2019\)](#). The Neuroevolution of Augmenting Topologies (NEAT) technique, pioneered by [K. Stanley and Miikkulainen \(2002\)](#), emerged as a highly influential approach. Building on the principles of NEAT, the HyperNEAT algorithm, introduced by [K. O. Stanley, D'Ambrosio, and Gauci \(2009\)](#), expanded these capabilities to facilitate the evolution of large-scale neural networks. This was accomplished through the use of indirect encoding via Compositional Pattern Producing Networks (CPPNs). However, it is important to note that while HyperNEAT showed effectiveness in certain areas, its application in image classification, particularly in feature learning with Convolutional Neural Networks (CNNs), was less successful. This limitation was highlighted by [Verbancsics and Harguess \(2015\)](#), who reported a 7.9% error rate on the MNIST dataset using this method.

Two notable extensions of the NEAT framework are DeepNEAT and CoDeepNEAT. DeepNEAT by [Miikkulainen et al. \(2019a\)](#) adapts the principles of NEAT to handle the complexities and depth of modern neural networks, making it suitable for more complex tasks and larger datasets. It handles the evolution of deep neural networks, which are inherently more complex and computationally expensive compared to simpler models. This

complexity can lead to increased computational resources and time required for training and evolution. Also, given its ability to evolve very deep networks, there's a potential risk of overfitting, especially when used on datasets with limited size or diversity.

CoDeepNEAT by [Miikkulainen et al. \(2019b\)](#), uses cooperative coevolution to evolve both the architecture and components of deep neural networks. It allows for more complex architectures by evolving separate modules that can be combined into larger networks. The framework however suffers from ensuring compatibility and optimal integration of these modules, as the evolved modules may not always perform well when combined.

Another noteworthy contribution is the Genetic CNN developed by [Xie and Yuille \(2017\)](#) which utilizes a Genetic Algorithm with a binary string representation. This method showed promising recognition capabilities on the MNIST dataset, although it adapted the fully-connected portion of the basic LeNet architecture [Lecun, Bottou, Bengio, and Haffner \(1998\)](#) without encoding it.

[Baldominos, Saez, and Isasi \(2018\)](#) in their study, investigated the evolution of CNNs using a Genetic Algorithm with binary Gray encoding and Grammatical Evolution with an integer-based encoding. Their approach was comprehensive, covering both convolutional and fully-connected layers and including parameters for network connectivity, allowing for the evolution of recurrent structures. They also introduced a novel similarity metric with a niching strategy. However, it's crucial to note that this metric might not be fully applicable in scenarios with varying numbers of convolutional and dense layers, as it produces a zero evaluation in such cases.

[Sun, Xue, Zhang, and Yen \(2018\)](#) made strides in automated CNN design with a Genetic Algorithm that encoded functional blocks in variable-length chromosomes, incorporating skip connections for better adaptability. In a similar context, [Suganuma, Shirakawa, and Nagao \(2017\)](#), presented a Cartesian Genetic Programming-based method that included both convolutional and residual units in its function set. While these methods effectively encoded the convolutional aspects of networks, they did not explore the measurement of structural similarity among different neural networks, highlighting a potential area for further research.

In recent years, the field of medical imaging has witnessed significant advancements across various modalities, including X-ray, computed tomography (CT), ultrasonography (USG), and magnetic resonance imaging (MRI). These advancements have led to improvements in acquisition time, image quality, and cost-effectiveness. However, these techniques are not without their limitations, including noise, motion artifacts, non-uniform contrasts, and registration errors, which can introduce uncertainties in clinical diagnoses. To address these challenges, medical image processing has emerged as a crucial area of research, employing problem-specific strategies and image processing algorithms.

Traditionally, image processing in the medical domain relied on conventional machine learning models, often requiring the manual extraction of handcrafted features by domain experts. However, these approaches had inherent limitations, including a heavy reliance on expert intervention and the presence of arbitrary parameters. To overcome these limitations, the field has increasingly turned to convolutional neural networks (CNNs).

Evolutionary Deep Convolutional Neural Networks (DCNNs) have seen application

in the field of medical image analysis. A notable instance is the AdaResU-Net [Baldeon-Calisto and Lai-Yuen \(2019\)](#), a multi-objective evolutionary model designed to identify compact yet accurate networks for medical image segmentation. This approach begins with a fixed U-Net-based structure, upon which an evolutionary model optimizes hyperparameters. The creators of AdaResU-Net also introduced Adaen-Net [Baldeon Calisto and Lai-Yuen \(2020\)](#), another evolutionary model tailored for medical image segmentation. In their more recent research, they employed an evolutionary model to design complete network structures for both 2D and 3D imaging. These models were trained individually for the same dataset to develop corresponding 2D and 3D networks. Notably, the 3D model exhibited lower accuracy than its 2D counterpart, leading to the utilization of an ensemble model for final predictions.

EvoU-Net [Hassanzadeh et al. \(2020\)](#) represents another evolutionary approach in this domain, employing a genetic algorithm to evolve DCNNs for medical image segmentation. Unlike previous models, EvoU-Net is capable of determining the entire network structure and a portion of its training parameters for the specific use case. These evolutionary models showcase the potential of applying genetic algorithms and evolutionary strategies in refining neural network structures for complex tasks such as medical image segmentation. However, the need to train and evaluate numerous candidate architectures across generations in EvoU-Net demands significant computational resources.

Lately, the remarkable achievements of transformers in the field of natural language processing (NLP) [Vaswani et al. \(2023\)](#) have sparked interest in their application to the vision domain [Carion et al. \(2020\)](#), [Dosovitskiy et al. \(2021\)](#), [Touvron et al. \(2021\)](#), [Z. Liu et al. \(2021\)](#). This adaptation has resulted in consistent advancements across a range of vision-related tasks, highlighting the substantial promise of vision transformers in this area. A key focus among these developments is enhancing the effectiveness of models, particularly by refining their fundamental component: attention. This shift towards incorporating transformer architectures in vision tasks represents a significant stride in exploring new methodologies for improving model performance.

In crafting our hybrid neuroevolutionary algorithm, we drew inspiration from the literature on evolutionary computation and neural network design, particularly focusing on neuroevolution and its applications in deep learning and medical image analysis. Our approach combines Cartesian Genetic Programming (CGP) with Novelty Search and Simulated Annealing, aiming to optimize CNN architectures for complex tasks in 2D and 3D medical image segmentation. This methodology addresses the computational and architectural challenges identified in the works of [Miikkulainen et al. \(2019a\)](#), [Miikkulainen et al. \(2019b\)](#) on DeepNEAT and CoDeepNEAT, and [Suganuma et al. \(2017\)](#) on CGP-based CNN design. By integrating these strategies, we sought to enhance the efficiency and efficacy of CNN design processes, specifically tackling the high computational demands and the quest for architectural innovation as highlighted by EvoU-Net [Hassanzadeh et al. \(2020\)](#) and AdaResU-Net [Baldeon-Calisto and Lai-Yuen \(2019\)](#) and [Baldeon Calisto and Lai-Yuen \(2020\)](#). Unlike AdaResU-Net and EvoU-Net, which primarily focused on hyperparameter optimization within a fixed architecture, our method leverages CGP to evolve both the structure and parameters of CNNs, offering a more holistic approach to network

design. The algorithm’s development was guided by the goal of overcoming the limitations of previous evolutionary approaches in the medical imaging domain, aiming to deliver a robust, computationally less intensive solution capable of generating optimized neural network architectures for advanced medical image segmentation tasks.

Furthermore, addressing the limitations highlighted in previous works, such as the Genetic CNN’s reliance on binary string representations and the inefficiencies in structural optimization, our algorithm incorporates a more precise novelty computation method. This is achieved through the integration of SimGNN [Bai et al. \(2020\)](#), which surpasses traditional methods like cosine similarity in exploring the architectural design space, thereby facilitating the discovery of innovative and efficient CNN designs for medical image segmentation.

Chapter 3

Hybridized Neuroevolution for Image Classification

Convolutional Neural Networks (CNNs) have significantly advanced the field of computer vision, proving their effectiveness in a variety of vision-related tasks. However, designing optimal CNN architectures remains a complex task, requiring in-depth domain expertise and often a considerable amount of experimentation. In recent years, Evolutionary Algorithms (EAs) have gained prominence as a means to streamline the CNN design process. While EAs are capable of navigating through extensive solution spaces, they also bring challenges, particularly the high computational demand involved in searching complex and often misleading solution landscapes.

In this chapter, our exploration is centered on the design of CNN architectures, specifically through the use of Cartesian Genetic Programming (CGP), a form of Evolutionary Algorithm. We enhance the basic CGP framework by integrating advanced strategies for detecting and refining potential or existing local optima in the broad solution space. This enhancement involves implementing Novelty Search, which encourages the discovery of diverse solutions, and further refinement using the Simulated Annealing algorithm for more precise optimization.

Our detailed experiments are conducted using the MNIST dataset [LeCun and Cortes \(2010\)](#), a widely recognized benchmark in the field of handwritten digit recognition. The goal of our research is to rigorously test and evaluate our hybrid Evolutionary Algorithm approach. We aim to highlight the significant decrease in computational requirements this method offers compared to the traditional EA method in terms of the fitness evaluations. At the same time, we intend to demonstrate that our method achieves comparable, if not superior, results in terms of recognition accuracy. Additionally, we aspire to establish the CNNs developed through our approach as noteworthy competitors in the realm of neuroevolutionary methods

This chapter offers an in-depth examination of our approach, focusing on the methods we have implemented to refine the process of designing CNN architectures. Our goal is to

validate the efficacy of our methodology through analysis and empirical data, demonstrating its ability to improve both the efficiency and the overall performance of CNN architecture design.

3.1 Methodology

Our proposed algorithm is a three stage hybrid algorithm as illustrated in Figure. 3.1. It includes the rich initialization stage, the evolutionary algorithm stage and finally the local optimization stage, each of which are explained in detail in the sections below.

We develop and monitor two distinct populations of CNNs. The initial population, known as Population A, starts with a set of randomly generated individuals and is evolved and optimized according to the fitness criteria described in Section. 3.1.3. The second population, Population B, is initialized using a **rich initialization phase**. This is a distinct initialization phase, where the individuals are first initialized randomly, and then optimized for their novelty metric as the fitness objective using the Novelty Search algorithm [Lehman and Stanley \(2011\)](#) for 30 generations. Instead of relying solely on random generation, this step guides the initial population towards diversity, creating a foundation that emphasizes novelty right from the start, setting the stage for the subsequent evolutionary process.

The networks we evolve are structured as a synthesis of two distinct sub-networks: one that is fully convolutional and another that comprises fully connected (dense) layers. The genotype representation of our networks merges two key components: a Cartesian Genetic Programming (CGP) grid and a sequence of fully connected layer configurations.

Table 3.1: Node functions present in the CGP function set. F: number of filters (output channels), K: kernel size, N: number of units in a dense layer.

Sub-Network	Node Function	Variation
Fully-Convolutional Network	ConvBlock	F={1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024} K={1,3,5}
	ResBlock	F={1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024} K={1,3,5}
	DeConvBlock	F={1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024} K={1,3,5}
	Pooling	Average and Max Pooling
	MergeBlock	Concatenation and Summation
Fully-Connected Network	DenseBlock	N={8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096}

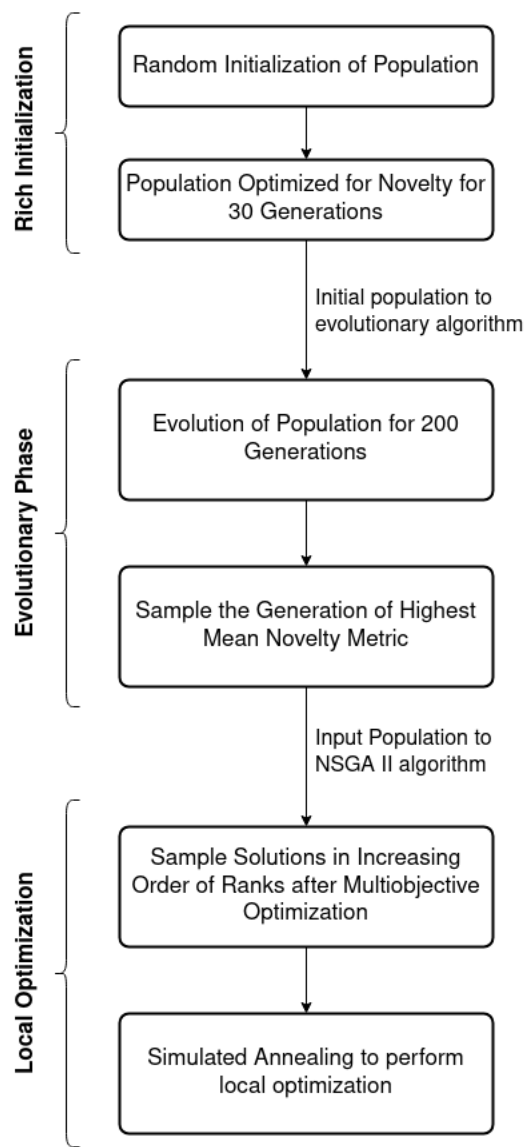


Figure 3.1: Illustration of the three distinct stages of our proposed hybrid algorithm - rich initialization, evolution and local optimization.

ConvBlock	0	0
ResBlock	0	0
AvgPool	1	1
ConvBlock	2	2
Concat	3	4
AvgPool	1	4
Output	5	5

Figure 3.2: Representation of the individual's genotype as a two-dimensional matrix, where each row represents a specific layer within the CNN architecture. These rows comprise three columns each, denoting the type of layer function alongside two node IDs to which the layer is connected to. Notably, the row highlighted in gray, featuring an Average Pooling layer linked to nodes 1 and 4, signifies an inactive node. This particular layer is not expressed in the phenotype, which is the Keras CNN model.

3.1.1 Genetic Representation

To encode the fully convolutional part of the network, we employ a CGP-based method inspired by Suganuma et al. [Suganuma et al. \(2017\)](#), utilizing the functional modules listed in Table 3.1 as node functions. The fully connected segment of the network is modeled using a sequence of node functions selected from a set of dense layers. Therefore, the genotype is a combination of a CGP grid, where each grid element comprises a function gene and a connection gene, and a sequence of dense layer node functions.

CGP, a variant of genetic programming, encodes candidate solutions in a fixed-length format. This encoding is executed using a two-dimensional matrix, where each row contains a function gene and connection genes as in Figure. 3.2. These genes together determine the node’s function and its input node indices, corresponding to a specific node in the network. Seven types of node functions are prepared and included in the function set, namely, Convolutional Blocks (ConvBlocks), Residual Blocks (ResBlocks), Deconvolutional Blocks (DeConvBlock), Max Pooling, Average Pooling, Concatenation, and Summation.

ConvBlocks play a crucial role, featuring adaptive convolutions that align with the data dimensionality. The adaptability is further enhanced by varying strides, usually set at 1 or 2, allowing the network to efficiently capture spatial details. Additionally, ConvBlocks vary in output channels and kernel sizes, which range from 1x1 to 5x5. Smaller kernels are effective for detailed feature extraction, while larger ones capture broader information. The use of "same convolutions" maintains spatial dimensions, relegating dimension reduction to pooling layers. This approach is vital for preserving spatial information, a key aspect of feature extraction and segmentation in our CNN designs. Moreover, ConvBlock layers include batch normalization for training stability and ReLU activation functions to add non-linearity, thereby improving the expressiveness of the features.

To facilitate the construction of deep network structures, we integrate Residual Blocks (ResBlocks) which build on the ConvBlock design. These ResBlocks include an extra convolutional layer, batch normalization, and ReLU activation functions. Their key contribution lies in adding depth to the network and mitigating the vanishing gradient problem via residual connections. Conversely, Deconvolutional Blocks (DeConvBlocks) utilize transposed convolution operations, again paired with batch normalization and ReLU activations, and typically apply an upsampling factor of 2. This element is crucial for reinstating the spatial resolution in feature maps.

Pooling layers are essential due to their ability to condense spatial dimensions while preserving essential feature information, thereby playing a vital role. These layers exhibit versatility with the implementation of 2x2 windows and a stride of 2.

Our method emphasizes feature fusion, which we achieve through concatenation and summation layers. These layers are instrumental in merging outputs from various convolutional layers, thereby enriching the overall feature representation. Concatenation layers enable the combination of multiple feature maps along the channel axis, capturing diverse information from different layers. When dealing with feature maps of varying dimensions, we employ max pooling for down-sampling to align dimensions before concatenation. On the other hand, summation layers allow for the addition of corresponding elements within

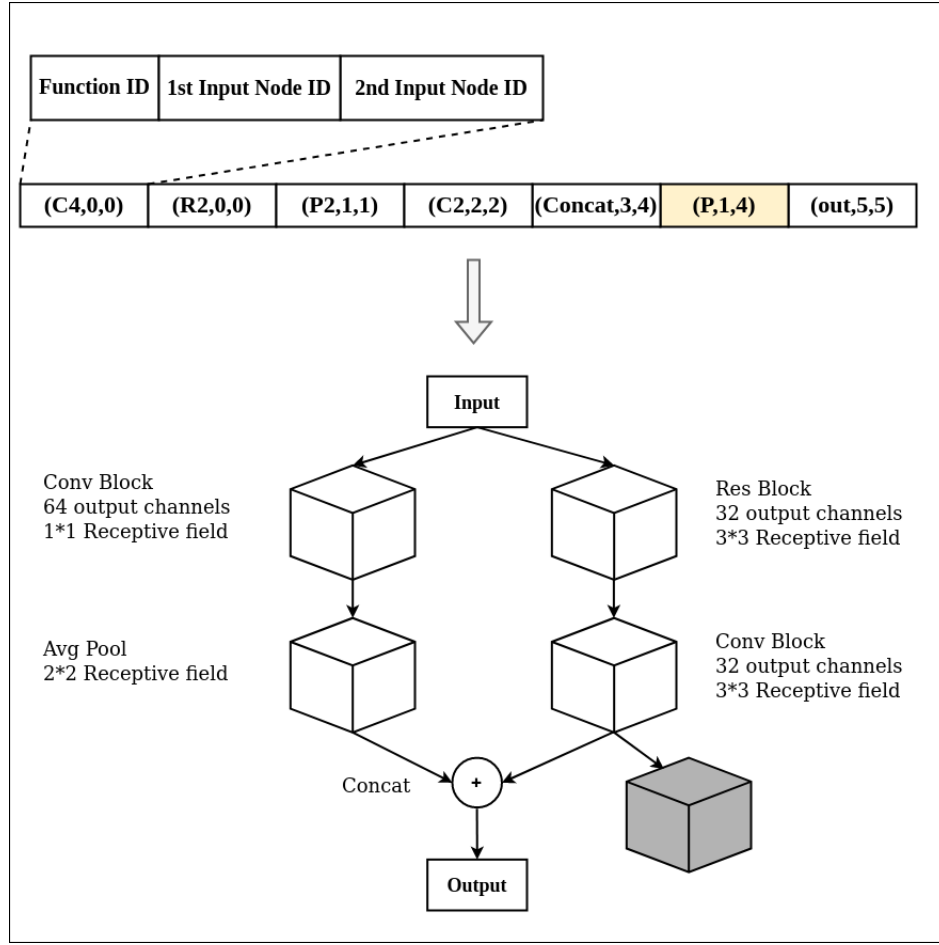


Figure 3.3: An illustration of an example genotype and it's equivalent phenotype. We can see that each cell in the genotype contains a function id and two input connection id's.

feature maps, facilitating the integration of relevant features. To address any discrepancies in dimensions, we strategically use 1x1 convolutions before the summation process to ensure smooth integration of features.

Complementing this, the sequence for the fully connected sub-network is specified through a set of node functions selected from a predefined set of dense layer operations. A softmax output layer with 10 nodes is appended to the end of the fully-connected network. This dual-component structure efficiently encapsulates the complexity and adaptability of our evolved networks.

3.1.2 Phenotypic Representation

The conversion of CGP representation into a Directed Acyclic Graph (DAG) forms a vital step in bridging the genotypic and phenotypic aspects of our CNN architecture. This intermediate phenotypic (DAG) representation enables the effective translation of genetic

code into a functioning neural network.

In the DAG framework, each node represents specific operations or transformations, and edges denote the flow of data through these operations. This arrangement not only aids in visualizing and understanding the network’s topology but also ensures the maintenance of its feed-forward characteristic, which is crucial for its effective operation during both training and inference phases.

The term ‘phenotype’ in this context refers to the actual CNN model, which we implement using the Keras framework [Chollet et al. \(2015\)](#). This model is a direct extrapolation of the DAG structure, where the abstract operations and linkages in the DAG are meticulously transformed into practical layers and connections within a Keras-based model. This transformation is pivotal, as it bridges the gap between the genetic blueprint of the CNN’s architecture and its tangible realization in a deep learning framework. Figure. [3.3](#) illustrates a genotype and its equivalent phenotype.

3.1.3 Evolutionary Algorithm

Evolutionary strategies, draw inspiration from the principles of biological evolution, harnessing the dynamics of mutation, recombination, and selection to refine candidate solutions. In the context of our research, we employ the $(\mu + \lambda)$ evolutionary strategy [T.-Y. Huang and Chen \(2001\)](#) to iteratively evolve CNN architectures tailored for image classification tasks. The overview of our evolutionary algorithm is illustrated in Figure. [3.4](#)

Representation

In the context of evolving neural network architectures using CGP, each individual within the population represents a unique neural network architecture. The individual’s genome comprises a series of genes, where each gene encodes specific attributes of the network nodes, including their type (e.g., convolutional, pooling, fully connected layers) and connections to other nodes. This representation as explained in Sec [3.1.1](#) allows for the flexible design of complex network architectures through simple genetic operations such as mutation and crossover.

The gene of an individual is structured as a two-dimensional array, with each row corresponding to a layer in the model architecture and columns encoding the layer’s function type and its connections. The first column of each row specifies the type of layer function, chosen from a predefined set of network layers or operations as seen in Table [3.1](#), while the subsequent columns indicate the layers to which it connects, thereby defining the model’s architecture. This encoding scheme ensures that the evolutionary algorithm can efficiently navigate the search space of possible network architectures.

Initialization

The initialization of population is a critical first step in the evolutionary process. It involves generating a set of initial individuals, each representing a potential solution to the problem at hand. In our CGP-based approach, this entails the creation of diverse neural

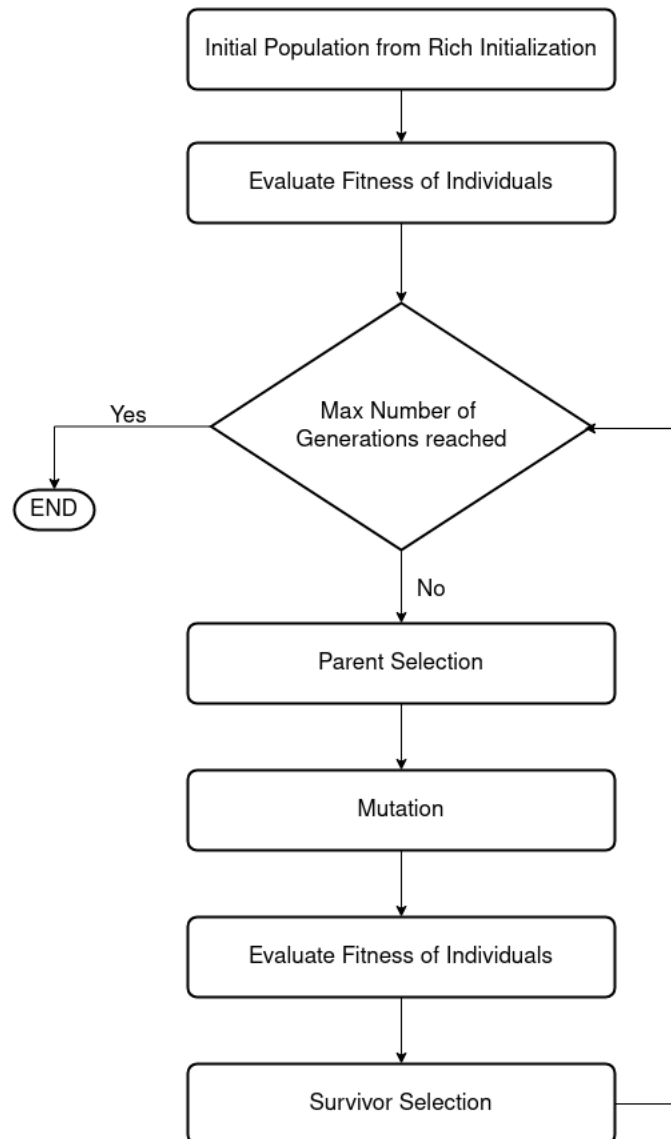


Figure 3.4: Overview of an evolutionary algorithm

network architectures through random initialization methods, where both the type and connectivity of each node are determined by randomly selecting values within the constraints defined by the network configuration. This approach ensures a broad exploration of the architectural search space from the beginning of the evolutionary process. Random initialization is particularly useful for discovering novel network architectures that might not be immediately apparent through human intuition or prior knowledge. The initialization phase also includes the identification of active nodes within each network, ensuring that subsequent genetic operations only affect parts of the architecture that contribute to its output. This focused approach enhances the efficiency of the evolutionary process, allowing for the rapid evolution of high-performing neural network architectures.

For our proposed algorithm, we perform a rich initialization. This involves generating a random initial population and then optimizing it for the novelty metric (computed according to Algorithm 3) according to the Novelty Search algorithm [Lehman and Stanley \(2011\)](#) for 30 generations.

To test the performance of our hybrid algorithm we initialize two distinct populations denoted as Population A and Population B:

Population A: Consists of 20 individuals that are randomly initialized.

Population B: Consists of 20 individuals that are initialized by rich initialization.

Parent Selection

This process involves choosing individuals from the current population to serve as parents for producing offspring in the next generation. In the described evolutionary algorithm, tournament selection [B. L. Miller and Goldberg \(1995\)](#) is employed with a window size of 5, which means that random subsets of the population, consisting of 5 individuals, are repeatedly created. From each subset, the best-performing individual is selected as a parent. We select 20% of the population without replacement. This method helps ensure that individuals with higher fitness have a better chance of becoming parents, thus propagating their genetic information to the next generation. Algorithm. 1 outlines the parent selection method.

Mutation

Mutation is a fundamental operator that introduces genetic variation into the population without the need for recombination (crossover). It is a crucial mechanism for exploring the solution space and preventing premature convergence of the algorithm. The designed mutation operator acts on the genotype of CNNs (illustrated in Figure. 3.2) altering the function type and the connection id's. Through mutation, a layers's function can be randomly altered to another function from the set of functions indicated in Table 3.1. This process allows for the exploration of different architectural configurations. It also involves swapping the connections of a layer with those of its immediate neighbors. This alteration in the connectivity pattern can lead to variations in the information flow within the network, potentially resulting in novel architectures that could be more effective for the given task.

Algorithm 1 Parent Selection using Tournament Selection Algorithm

Require: *population* : population to undergo parent selection
Require: *tour_size* : size of the tournament
Require: *num_tours* : number of tournaments to be conducted
Require: *replacement* : False
 selected $\leftarrow \square$
 while $\text{len}(\textit{selected}) < \textit{num_tours}$ **do**
 tournament $\leftarrow \text{randomChoice}(\textit{population}, \textit{tour_size})$
 max_fitness $\leftarrow 0$
 fittest $\leftarrow \text{None}$
 for *ind* **in** *tournament* **do**
 if $\textit{ind.fitness} - \textit{max_fitness} \geq \epsilon$ **then**
 max_fitness $\leftarrow \textit{ind.fitness}$
 fittest $\leftarrow \textit{ind}$
 end if
 end for
 if *fittest* $\notin \textit{selected}$ **and** *fittest.eval* $\neq 0$ **then**
 selected.append(fittest)
 end if
 end while
 return *selected*

In a separate development, [Real et al. \(2017\)](#) explored an algorithm to optimize the architectures of CNNs. The evolutionary algorithm’s operators, inspired by NEAT, predominantly focused on mutations, which proved highly effective. However, three variations of crossover did not yield any enhancement in results and were therefore not included in further applications. Similar observations have been made by [Silva, Urbano, Correia, and Christensen \(2014\)](#) and [Silva, Correia, and Christensen \(2017\)](#). Thus, our work progresses with the inclusion of a mutation operator only.

Survivor Selection

The algorithm proceeds to selecting individuals for the next generation from the existing population, including both parents and offspring. In this context, we employ tournament selection without replacement. When evaluating two individuals in case of Population A, the algorithm may select the less fit candidate if it lies within a 0.05 fitness tolerance of the fitter individual or possesses fewer trainable parameters. This selection strategy ensures that promising candidates progress to the subsequent generation while considering both their fitness and model complexity. But in case of Population B, the algorithm may select the less fit candidate if it lies within a 0.05 fitness tolerance of the fitter individual or possesses fewer trainable parameters or has a higher novelty score. Algorithm. 2 outlines the survivor selection method.

Algorithm 2 Survivor Selection using Tournament Selection Algorithm

Require: *population* : population from the previous generation
Require: *parents* : list of individuals from parent selection
Require: *children* : list of individuals after mutation of parents and fitness evaluated
Require: *tour_size* : size of the tournament
Require: *replacement* : False

```
next_gen ← []
while len(next_gen) < len(parents) do
    tournament ← randomChoice(population, tour_size, replacement)
    max_fitness ← 0
    fittest ← None
    for ind in tournament do
        if ind.fitness − max_fitness ≥ ε then
            max_fitness ← ind.fitness
            fittest ← ind
        else if |ind.fitness − max_fitness| ≤ 0.05 then
            if fittest = None then
                max_fitness ← ind.fitness
                fittest ← ind
            else if ind.trainable_params < fittest.trainable_params then
                max_fitness ← ind.fitness
                fittest ← ind
            else if ind.novelty > fittest.novelty then
                max_fitness ← ind.fitness
                fittest ← ind
            end if
        end if
    end for
    if fittest ≠ None and fittest ∉ next_gen and fittest.eval ≠ 0 then
        next_gen.append(fittest)
    end if
end while
for p in parents do
    if p ∈ population then
        population.remove(p)
    end if
end for
for c in next_gen do
    population.append(c)
end for
```

Fitness Evaluation

Fitness evaluations are a part of evolution that helps determine how good or effective candidate solutions are within a population. These evaluations serve as a guide for the evolution process to find the best or nearly best solutions. Specifically, in our case, we evaluate each CNN model by training it on the MNIST dataset for 5 epochs. After training, we test the model’s performance on a separate held out validation dataset. It should be noted that the test set is not used for evaluation during evolution and training hyperparameters are kept constant for all individuals.

We use the Adam optimization algorithm [Kingma and Ba \(2017\)](#) with a uniform learning rate of $1e - 4$ to minimize the Categorical Cross-Entropy loss. To measure the performance of the CNN models we compute the F1 Score, a common metric used in image classification tasks. The F1 Score assesses the predictive skill of a model by elaborating on its class-wise performance rather than an overall performance as done by accuracy. It is calculated as shown in Eqn. 1.

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \times TP}{2 \times TP + FP + FN} \quad (1)$$

where TP represents True Positive, FP represents False Positives and FN represents False Negatives.

Termination

The evolutionary algorithm is terminated after a fixed number of generations. We chose to run the evolutionary process upto 200 generations. All the hyperparameters used for the evolutionary algorithm are summarized in Table. [3.2](#)

Table 3.2: Summary of Parameters used for the evolutionary algorithm

Parameter	Value
CGP grid size $N_r \times N_c$	5 x 30
Population size	20
Tournament Size	5
Mutation Probability	0.1
Optimizer	Adam
Learning Rate	Uniform ($1e - 4$)
Number of epochs trained during Fitness Evaluation	5
Loss Function	Cross-Entropy loss
Survivor Selection Fitness Tolerance	0.05
Number of Generations	200

3.1.4 Diversity Measures and Novelty Search

In our methodology, the incorporation of novelty computation, via cosine similarity, is a measure of the diversity within the population of CNN architectures and is of particular significance as it encourages the exploration of distinctive and potentially innovative architectural configurations that may otherwise remain undiscovered.

Vector Representation

We adopt a novel method for representing the structural attributes of CNNs, specifically aimed at evaluating diversity within these architectures. Each layer within a CNN is translated into a 4-dimensional vector, with each dimension thoughtfully designed to capture distinct properties:

c_i : Signifying the number of channels present in the layer.

x_i : Indicating the width of the layer’s activation map.

y_i : Reflecting the height of the layer’s activation map.

d_i : Defining the depth of the layer as the minimum number of layers between the considered layer and the input layer.

$$v_i = \begin{bmatrix} c_i \\ x_i \\ y_i \\ d_i \end{bmatrix} \quad (2)$$

This vector-based representation as in Eqn. 2 serves as a concise yet comprehensive means of encapsulating vital structural information relevant to each layer. The transformation of each CNN layer into this 4-dimensional space not only facilitates the quantification of architectural characteristics but also introduces an innovative approach for comparing and contrasting diverse CNN architectures.

Cosine Similarity and K Nearest Neighbors Computation

Cosine similarity, widely used in fields like text analysis, proves to be a valuable metric for assessing the structural similarity of CNN architectures. It provides a meaningful way to compare their characteristics. This metric focuses on the direction of the vector representations of CNN layers, intentionally disregarding their magnitude. Such an orientation-focused measure is especially relevant in our research. It highlights the differences in architecture between various designs, prioritizing their unique structural features over the size of the layers. This approach is particularly useful in evaluating the novelty of different CNNs, as it places more emphasis on their distinct configurations rather than the scale of their components.

To quantify the novelty of an individual CNN within the population, the algorithm (as shown in Algorithm 3) calculates its similarity with a set of k nearest neighbors [Cover and Hart \(1967\)](#) (with $k=11$ in this particular investigation). This calculation involves averaging the cosine similarities between the individual CNN and its closest neighbors. The resulting

novelty score is obtained by scaling the average similarity and subtracting it from 100. Consequently, the novelty score falls within the range of 0 to 100, where higher scores indicate a greater degree of novelty relative to the nearest neighbors.

Utilizing cosine similarity in this context marks a significant progression in the field of evolutionary algorithms for optimizing CNN architectures. It adds a sophisticated layer to the evaluation process, uncovering distinct characteristics of each architecture and encouraging the exploration of a wider range of solutions. By focusing on this method, we redefine the scope of architectural diversity, thereby enhancing the overall landscape of CNN architecture optimization. This approach allows for a more comprehensive assessment of architectural variations, fostering the discovery of more innovative and diverse network designs.

Algorithm 3 Computation of novelty of an individual as the normalized sum of its similarity with its k nearest neighbours.

Require: V1: vector set representation of individual 1
Require: V2: vector set representation of individual 2
1: similarity: array of length $\max(\text{length}(V1), \text{length}(V2))$
2: norm_sum: normalized element-wise sum of similarity array
3: $V1 = V1[1:\text{length}(V1)-1]$
4: $V2 = V2[1:\text{length}(V2)-1]$
5: **if** $\text{length}(V2) > \text{length}(V1)$ **then**
6: $temp \leftarrow V1$
7: $V1 \leftarrow V2$
8: $V2 \leftarrow temp$
9: **end if**
10: **for** $i \leftarrow 0$ to $\text{length}(V1) - \text{length}(V2)$ **do**
11: $V2.append([0,0,0,0])$
12: **end for**
13: **for** $i \leftarrow 0$ to $\text{length}(V1)$ **do**
14: **if** $\text{norm}(V2) = 0$ **then**
15: $similarity[i] \leftarrow 0$
16: **else**
17: $similarity[i] \leftarrow$
18: $\text{cosine_similarity}(V1[i], V2[i])$
19: **end if**
20: **end for**
21: $norm_sum \leftarrow \text{sum}(similarity) / \text{length}(similarity)$

3.1.5 Local Optimization

The proposed hybrid algorithm combines Evolutionary Algorithms (EAs) for broad exploration and Stochastic Local Search (SLS) algorithms for focused optimization of solutions that are nearly optimal. The local optimization phase of this algorithm, samples a generation of sub optimal solutions from the evolutionary phase and optimizes them for

their novelty metric and fitness score using a multi objective optimization algorithm. The promising sub optimal solutions from this phase are subjected to a rigorous local search within their neighborhood to identify optimal architectures.

Identification of Local Optima

In order to demonstrate this procedure, we select from Populations A and B the generations with highest mean novelty as shown in Figure. 3.5. Every individual in this sampled population consists of a fitness score (as explained in Section. 3.1.3) and a novelty score (as calculated in Section. 3.1.4). Based on these two metrics, we select individuals which are determined to be located near/at local optima. To identify these local optima, we utilize the Non-dominated Sorting Genetic Algorithm II (NSGA-II), a multi-objective optimization algorithm by [Deb, Agrawal, Pratap, and Meyarivan \(2002\)](#) known for its ability to manage multiple objectives efficiently. NSGA-II sorts CNN structures based on their multi-objective fitness values and novelty measures into various non-dominated fronts.

The selection mechanism involves choosing solutions from these fronts, particularly prioritizing those from the highest-ranked front until we have obtained at least 4 individuals. This approach is intended to identify individuals that excel across multiple objectives, such as classification accuracy and diversity. When the new population has limited capacity, a crowding-distance operator is used to select individuals from the less crowded areas of a front. This method ensures the preservation of a diverse group of promising individuals for subsequent evaluation and enhancement.

Figure. 3.6 shows the ranking of the solutions for Population A and B by the NSGA II algorithm. The solutions sampled from Population A consist of rank 1 and rank 2 solutions, while the four solutions sampled from Population B consist of rank 1 solutions only. These constitute the Population C and D, respectively.

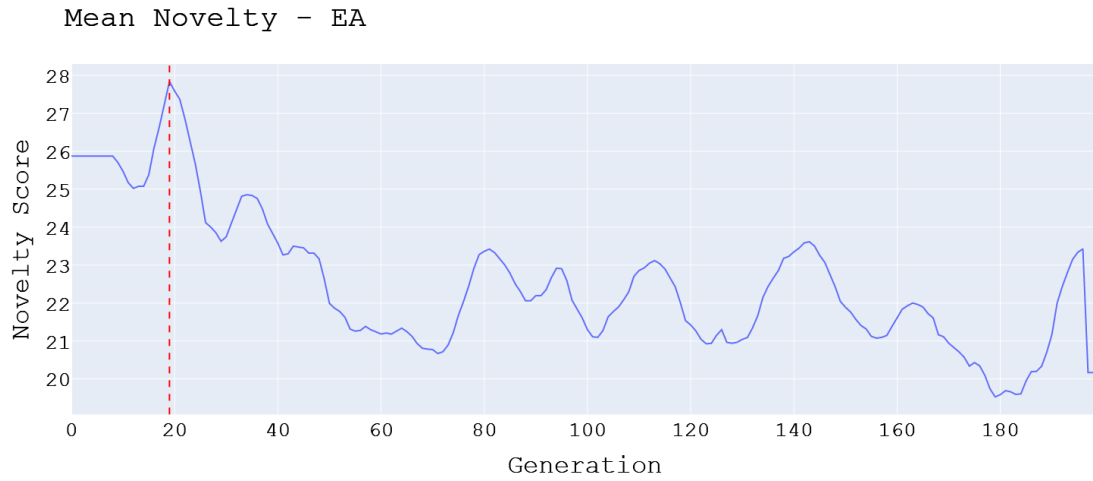
Population C: Comprises of individuals sampled from Population A to undergo local optimization.

Population D: Comprises of individuals sampled from Population B to undergo local optimization.

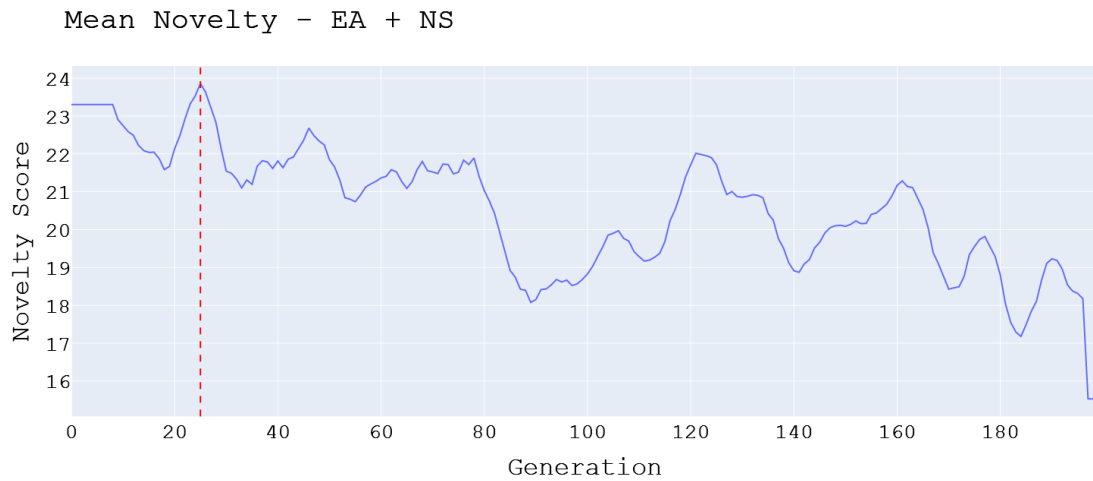
Simulated Annealing

Simulated Annealing (SA) algorithm by [Kirkpatrick et al. \(1983\)](#) is a heuristic optimization method inspired by the annealing process of metals, and in this context, it plays a significant role in refining the CNN architectures selected for local optimization. The SA algorithm as shown in Algorithm 4 starts with an initial "high temperature" phase, where it explores a broad solution space, allowing it to escape potential local optima. As the algorithm progresses, it follows a geometric cooling schedule to reduce the temperature exponentially by a parameter α over iterations as in Eqn. 3. This schedule ensures that the temperature decreases gradually, making the algorithm more focused on exploiting promising solutions as it proceeds.

$$T = \alpha \times T \quad (3)$$



(a) Variation of mean novelty over different generations of evolutionary algorithm for Population A



(b) Variation of mean novelty over different generations of evolutionary algorithm for Population B

Figure 3.5: Mean novelty vs generations: evolutionary algorithm (top) and evolutionary algorithm with Novelty Search (bottom). In each case, the vertical line indicates the generation selected for local optimization.

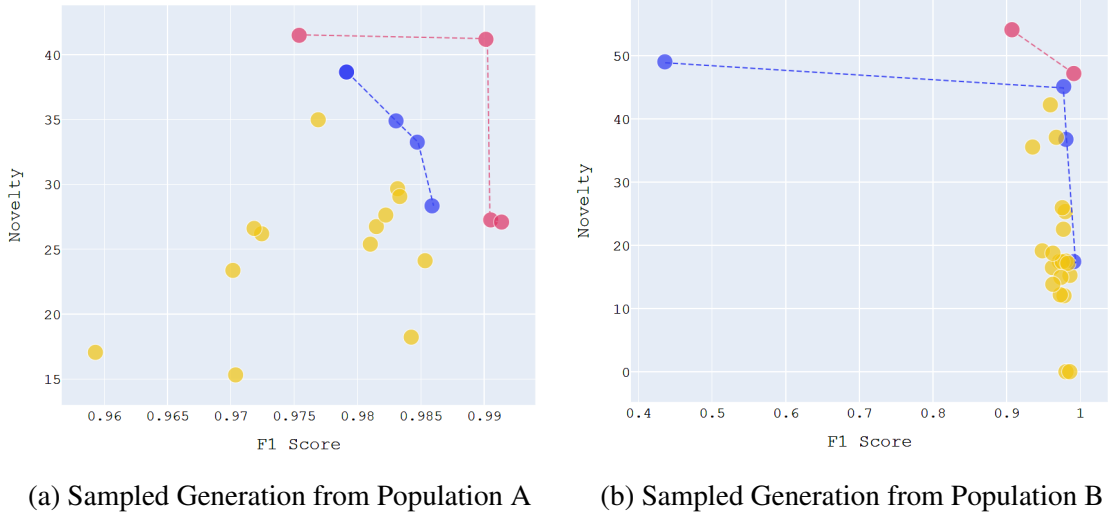


Figure 3.6: Solutions from the sampled generations of highest mean novelty from (a) Population A and (b) Population B are sorted in ranks based on their novelty and fitness score by the NSGA II algorithm. The solutions in pink belong to rank 1, solutions in blue belong to rank 2 and solutions in yellow belong to further lower ranks.

Simulated Annealing relies on a cost function to evaluate the quality of candidate solutions. This cost function quantifies the difference in fitness between the current solution and a neighboring solution. The algorithm generates neighboring solutions by minor mutations to the current solution. The acceptance of a new solution is determined by a probabilistic rule described by Eqn. 4, which considers the cost difference between the current solution and the neighboring solution, as well as the current temperature. If the cost is higher (indicating a better solution) or the temperature is higher (allowing more exploration), the new solution is accepted. The algorithm terminates either after a fixed number of iterations or when the temperature decreases to a predefined threshold. Table. 3.3 summarizes the parameters used in the SA algorithm.

$$P = \exp\left(\frac{cost}{T}\right) \quad (4)$$

Table 3.3: Summary of Parameters used for the Local Optimization Algorithm

Parameter	Value
Initial Temperature	1000
Cooling Schedule	Geometric
α	0.91
Final Temperature	0.000032155
Number of Iterations	183

Algorithm 4 Simulated Annealing Algorithm

Require: *initial_state*: the initial state of the system

Require: *temp*: initial temperature for the algorithm

Require: *alpha*: temperature decay rate

Require: *temp_final*: final temperature threshold

```
1: current_state  $\leftarrow$  initial_state
2: best_state  $\leftarrow$  initial_state
3: current_temp  $\leftarrow$  temp
4: while current_temp > temp_final do
5:   new_state  $\leftarrow$  generate_neighbor(current_state)
6:   cost_diff  $\leftarrow$  cost(new_state) – cost(current_state)
7:   if cost_diff < 0 then
8:     current_state  $\leftarrow$  new_state
9:     if cost(new_state) < cost(best_state) then
10:      best_state  $\leftarrow$  new_state
11:    end if
12:  else
13:    r  $\leftarrow$  randomnumberinrange(0, 1)
14:    if  $\exp(-\text{cost\_diff} / \text{current\_temp}) > r$  then
15:      current_state  $\leftarrow$  new_state
16:    end if
17:  end if
18:  current_temp  $\leftarrow$  current_temp * alpha
19: end while
20: return best_state
```

3.2 Dataset

The MNIST dataset, introduced by [LeCun and Cortes \(2010\)](#), serves as a fundamental cornerstone in the realm of machine learning and computer vision. Comprising a collection of 28x28 pixel gray scale images depicting handwritten digits, it has gained widespread acclaim as a benchmark dataset for classification tasks. Despite its simplicity, it retains its role as an essential dataset in the ever-evolving landscape of machine learning and success on MNIST often translates to viability in more complex tasks.

3.3 Experimental Setting

All experiments are conducted on computational resources characterized by two Intel Xeon Gold 6238 Cascade Lake 2.10GHz CPUs with a total of 44 CPU cores, supported by four NVIDIA T4 Turing GPUs, and a substantial 187 GB of RAM.

For our experimental setup, we partition the training set into two distinct subsets: the primary training set, which encompasses 50,000 images, and a validation set comprising 10,000 images. Prior to training, we conduct preprocessing on the dataset, normalizing pixel values by dividing each by 255. The test dataset encompasses an additional 10,000 separate images.

From each of the population A, B, C and D, we sample the four fittest individuals. These selected individuals are trained on the full MNIST training dataset, for 100 epochs, with an early-stopping patience parameter set at 20 epochs. The trained networks are subsequently evaluated on the previously unseen test images from the dataset, providing a comprehensive assessment of their performance.

The evolutionary process is capped at 200 generations, resulting in 3204 fitness evaluations (20 evaluations in the initial generation and 16 subsequent evaluations in each subsequent generation).

3.4 Results and Discussion

3.4.1 Recognition Error

The recognition error obtained by the four fittest models from each population is summarized in Table 3.4. A comprehensive analysis of the performance of Population A, which underwent evolution alone, reveals a significant divergence in the performance of the fittest and fourth fittest individuals. This discrepancy serves as an indicator of the algorithm’s challenge in simultaneously exploiting different peaks within the fitness landscape, ultimately leading to a reduction in population diversity across generations. Notably, the variance in recognition error among the four top individuals of Population A is approximately 0.0169.

With the incorporation of novelty as a metric in Population B, a striking shift occurs, where the performance of the four best individuals becomes remarkably similar. The

Table 3.4: Summary of the four fittest individuals obtained from each population. Individuals are sorted by test error. The best results are highlighted in bold.

Population	Individual No.	Trainable Parameter Count	Fitness Evaluation Count	Fitness Score	Test Error
Population A	1	5.493 M	2676	0.992	0.57%
	2	0.968 M	3104	0.987	0.69%
	3	9.175 M	136	0.990	0.71%
	4	103.364 M	704	0.989	0.93%
Population B	1	850.491 M	2746	0.993	0.64%
	2	1.018 M	3158	0.987	0.72%
	3	0.232 M	3217	0.989	0.72%
	4	7.358 M	2821	0.985	0.73%
Population C	1	7.400 M	1004	0.994	0.67%
	2	14.488 M	964	0.993	0.52%
	3	32.219 M	716	0.992	0.83%
	4	19.118 M	936	0.992	0.84%
Population D	1	34.011 M	1072	0.994	0.48%
	2	46.063 M	1024	0.993	0.54%
	3	7.550 M	956	0.992	0.57%
	4	8.863 M	1068	0.992	0.62%

substantial novelty metric exhibited by each individual suggests that they originate from distinct regions within the fitness landscape. Notably, the variance in recognition error among the four leading individuals of Population B is substantially reduced to approximately 0.0013.

It is crucial to recall that during the local optimization phase, constraints are imposed to prevent individuals from entering the neighborhoods of solutions occupied by other individuals in the population. An examination of the recognition rates within Populations C and D indicates that in Population C, certain solutions appear to be confined to sub-optimal regions, resulting in varying recognition rates. Notably, Population D boasts the individual with the lowest error rate, standing at an impressive 0.48%. The variance in recognition error among the four top individuals of Population C and D is 0.0172 and 0.0025, respectively.

The above discussion highlights the impact of including novelty as an optimization objective, resulting in a reduction in recognition rate variance by an order of magnitude. The exploratory search behavior within our algorithm, facilitates the discovery of highly fit solutions across diverse regions of the search space while preserving a high degree of diversity.

3.4.2 Number of Fitness Evaluations

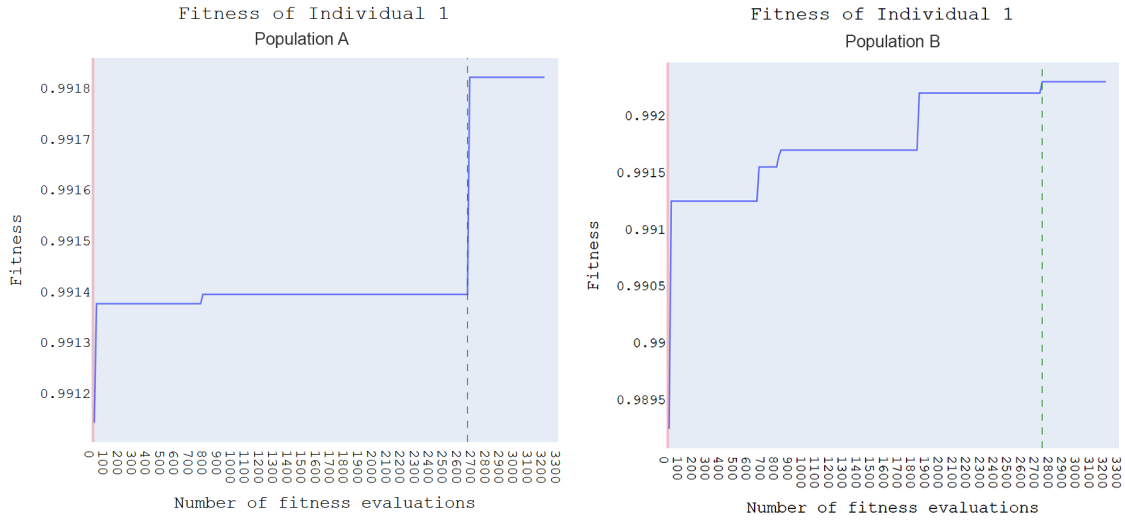
To assess and compare the computational effort required to obtain optimal solutions in each of the four populations, we conducted an analysis of the mean and standard deviation of the number of fitness evaluations. The results for populations A through D were as follows: 1655 ± 1260 , 2986 ± 204 , 905 ± 111 , and 1030 ± 46 , respectively. It is worth noting that, on average, each evaluation cycle consumed approximately 3.12 minutes.

Notably, both Populations C and D exhibited a reduced requirement for fitness evaluations to attain optimal solutions in comparison to Populations A and B ($p < 0.05$). This distinction is visually evident in Figure. 3.7, where the evolution process is characterized by a substantial number of redundant fitness evaluations before the discovery of optimal solutions. Conversely, the Simulated Annealing algorithm, employed in Populations C and D, demonstrates a rapid convergence towards local optima. This phenomenon becomes particularly pronounced after approximately 600 fitness evaluations, at which point the temperature has significantly decreased, rendering the algorithm more deterministic.

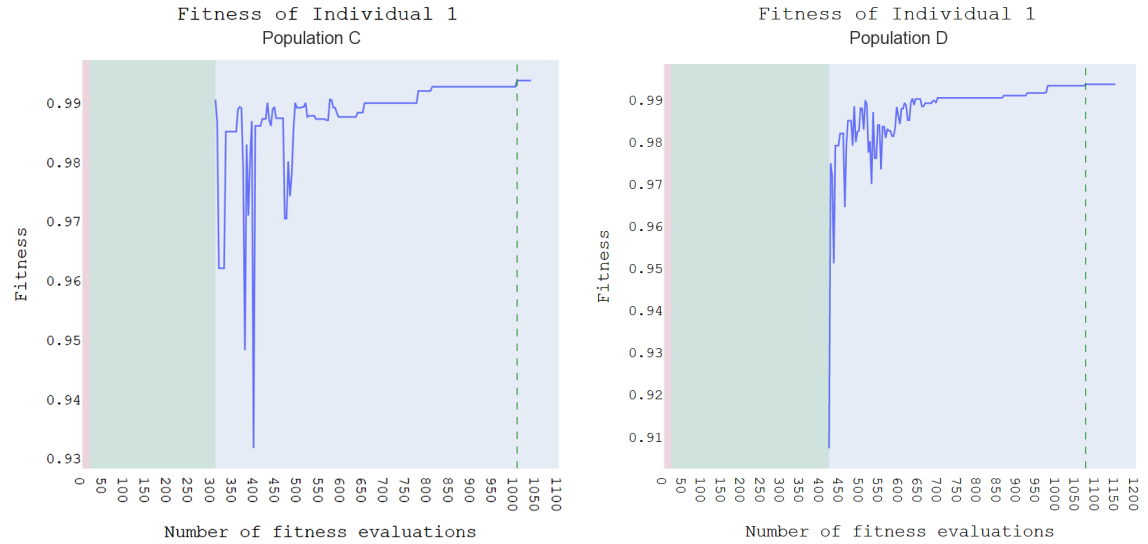
Furthermore, the lower variance observed in Populations C and D underscores the reliability of the local optimization approach in consistently achieving optimal solutions within a narrow range of fitness evaluations.

3.4.3 Comparison to related works

The proposed method demonstrates competitive performance in terms of recognition error when evaluated on the MNIST test dataset. However, the trained networks produced by our method do not attain state-of-the-art performance levels. It was observed during the training process that the early stopping technique consistently terminated training well



(a) The best fitness for Population A is obtained after 2676 evaluations and for Population B after 2746 evaluations



(b) The best fitness for Population C is obtained after 1004 evaluations and for Population D after 1072 evaluations

Figure 3.7: Variation of fitness over number of fitness evaluations during the evolutionary process for Individual 1 from Populations A through D. The shaded red region represents the number of fitness evaluations during the rich initialization of the population and the green region represents the number of fitness evaluations during the evolutionary process until sampling is done.

before reaching 100 epochs.

To potentially enhance the performance of our approach, there is room for further improvement. One of the methods would be the incorporation of additional regularization techniques, such as the inclusion of dropout layers as suggested by [Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov \(2014\)](#). These techniques may help mitigate issues related to overfitting and promote more effective training, ultimately leading to improved recognition performance. Table 3.5 shows the comparison to related works.

Table 3.5: Comparison of the recognition error (%) on the MNIST test dataset to related works.

Model	MNIST (%)
Fernandes Junior and Yen (2019) psoCNN	0.32
Xie and Yuille (2017) GeNet	0.36
Baldominos et al. (2018)	0.37
Our Method	0.48
B. Wang, Sun, Xue, and Zhang (2018) IPPSO	1.13
Sun, Yen, and Yi (2019) evoCNN	1.18
Verbancsics and Harguess (2015) HyperNEAT with CNN	7.9

3.5 Conclusions and Future Work

Our research encompasses the exploration of a hybrid Cartesian Genetic Programming (CGP)-based evolutionary algorithm designed for training Convolutional Neural Networks (CNNs) in handwritten digit recognition, with a specific focus on the renowned MNIST dataset. The initial stage is dedicated to the generation of a diverse population of CNNs. We measure the diversity of this population by employing a structural similarity metric that we have proposed. The second stage, focuses on evolving CNN models that exhibit superior performance while preserving the diversity of the population.

The third and pivotal stage demonstrates the hybrid algorithm’s capability to optimize potentially sub optimal solutions that have been identified in the preceding stage. This optimization process is not only effective but also computationally efficient in reducing the number of fitness evaluations that a traditional evolutionary algorithm would have incurred, marking a significant advancement in our methodology.

We evaluate the computational efficiency of our hybrid algorithm by quantifying the number of fitness evaluations required to obtain optimal solutions. Our findings highlight the computational advantages of our approach, showing it to be less resource-intensive compared to traditional methods. Notably, the inclusion of novelty as an optimization objective in our algorithm significantly reduces the variance in the recognition rate, making it an order of magnitude more stable than conventional evolutionary algorithms.

Our method attains a remarkable recognition error rate of 0.48% on the MNIST test dataset, reflecting its competitive performance in comparison to related works. However, it is essential to acknowledge that some of our peers have achieved recognition error rates as low as 0.32%.

In the realm of future work, we propose several promising directions. One avenue involves the exploration of structural similarity measures that operate on the Directed Acyclic Graph (DAG) representation of the CGP genotype, such as Graph Edit Distance [Sanfeliu and Fu \(1983\)](#). Additionally, we encourage the investigation of regularization techniques, such as dropout layers and the evolution of recurrent structures within CNNs, to further enhance performance. Furthermore, the application of our method to more intricate CNN tasks, such as image segmentation, holds the potential to yield valuable insights and advancements in the field.

Chapter 4

Hybridized Neuroevolution for 2D and 3D Image Segmentation

Since AlexNet’s breakthrough in 2012, Convolutional Neural Networks (CNNs) have become indispensable in image segmentation, revolutionizing the processing of both 2D and 3D images. This evolution has significantly impacted medical imaging, where accurate segmentation is crucial for diagnosis and treatment. Neuroevolution presents a novel approach to refining CNN architectures for segmentation tasks, promising tailored optimization of network parameters, but it also incurs high computational demands, especially for 3D image segmentation, where data complexity substantially increases.

Building on the performance of our hybrid evolutionary algorithm in image classification tasks detailed in Chapter 3, we plan to rigorously evaluate it on more complex challenges, such as medical image segmentation. This investigation will utilize the DRIVE dataset for retinal vessel segmentation by [Staal, Abramoff, Niemeijer, Viergever, and van Ginneken \(2004\)](#) and the Medical Segmentation Decathlon (MSD) challenge dataset by [Simpson et al. \(2019\)](#) which contains three-dimensional images collected across multiple anatomies of interest, multiple modalities and multiple sources. Our objective is to compare and contrast the CNN models generated by our hybrid algorithm against established benchmarked models in this field, focusing on segmentation accuracy, the Dice Score, and the number of trainable parameters. Enhancing the novelty metric computation beyond the cosine similarity method used in Chapter 3, we strive to validate our algorithm’s effectiveness in producing competitive and diverse solutions in fewer computation effort (indicated by number of fitness evaluations) compared to traditional neuroevolutionary methods.

4.1 Methodology

The overview of our proposed methodology is a three stage hybrid algorithm, including a rich initialization, evolutionary and the local optimization phase as illustrated in Figure. [4.1](#).

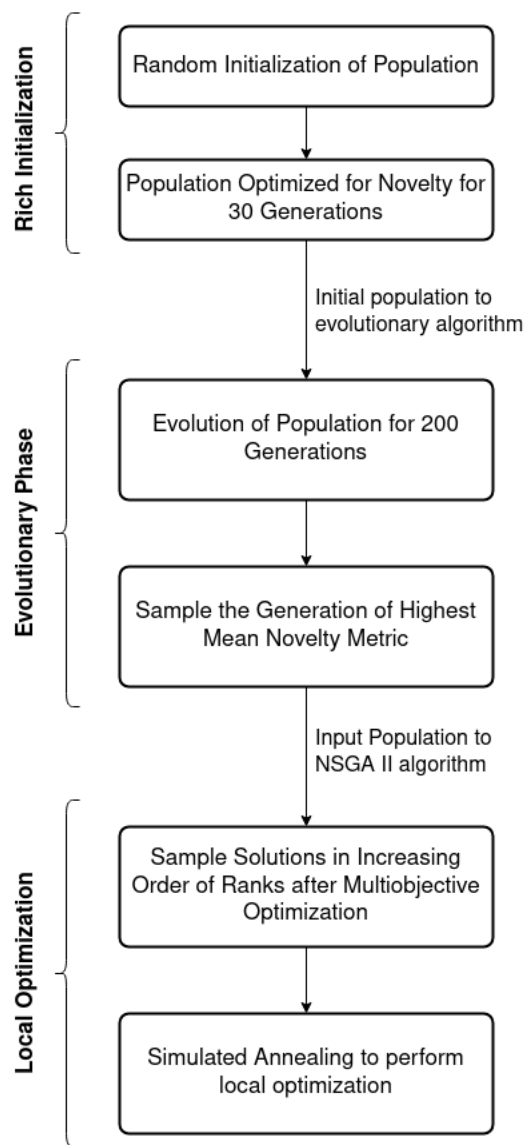


Figure 4.1: Three distinct stages of our proposed hybrid algorithm - rich initialization, evolutionary phase and local optimization.

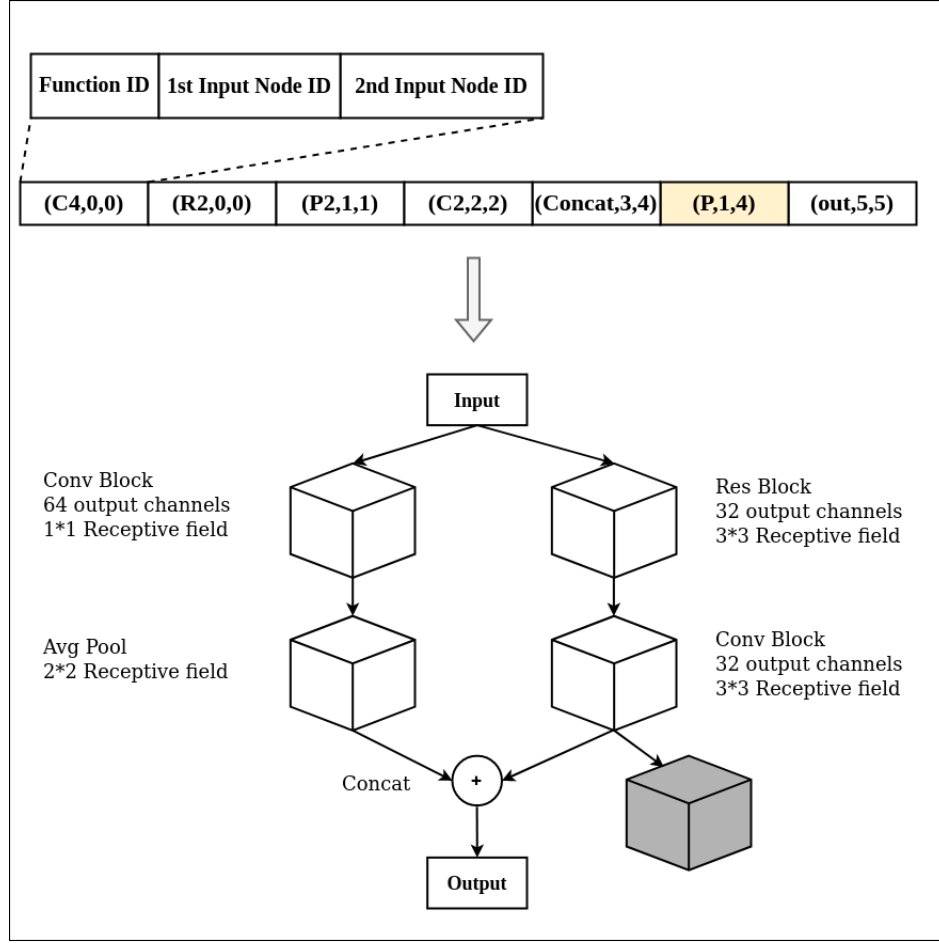


Figure 4.2: An illustration of an example genotype and it’s equivalent phenotype. We can see that each cell in the genotype contains a function id and two input connection id’s.

4.1.1 Genotypic and Phenotypic Representation

Th evolved networks for image segmentation only consist of a fully convolutional network without the fully connected sub network. This involves utilizing Cartesian Genetic Programming (CGP) to encode the architecture of neural networks, a technique inspired by [Suganuma et al. \(2017\)](#). Therefore, the genotype is a two dimensional CGP grid as in Fig. 3.2, where each grid element comprises a function gene and two connection genes, maintaining a consistent genotypic representation of neural networks.

Our selection of operations remains as diverse as introduced previously, covering Convolutional Blocks (ConvBlocks), Residual Blocks (ResBlocks), Deconvolutional Blocks (DeConvBlocks), and various pooling and fusion techniques. This selection, presented in Table. 4.1 ensures our network architectures are adaptable, capable of capturing spatial details efficiently through varying kernel sizes and strides, and promoting feature expressiveness with layers of batch normalization and ReLU activation functions. It is noteworthy

to observe that the function set does not contain any dense block functions constituting the fully connected layers.

Furthermore, the architecture’s depth and complexity are enhanced by incorporating ResBlocks and DeConvBlocks, addressing the challenge of the vanishing gradient problem and improving spatial resolution in feature maps, respectively. Pooling layers continue to play a crucial role in spatial dimensionality reduction, preserving essential feature information. In feature fusion, we leverage concatenation and summation layers to integrate diverse feature maps, enhancing the richness of the network’s feature representation.

The CGP representation is converted into a functional CNN architecture, utilizing the Directed Acyclic Graph (DAG) as an intermediary phenotypic representation. Within the DAG structure, each node signifies distinct layer operations or functions, while edges represent the data flow across these operations. This configuration gives us a clear understanding of the network’s topology while preserving the essential feed-forward nature of the CNN, vital for its efficiency during training and inference.

The ‘phenotype’, referring to the actual CNN model, is materialized using the Keras framework, as established by [Chollet et al. \(2015\)](#). This step involves translating the functions and connections depicted in the DAG into executable layers and linkages within a Keras-based CNN model. A final node with a softmax activation is appended to the network to obtain the output. This transformation is crucial for actualizing the CNN’s architecture from its genetic representation, enabling its application and evaluation in a deep learning environment.

The transition from genotype to phenotype is illustrated in Fig. 4.2, showcasing the direct correlation between the genetic encoding and the resultant CNN model. This methodology, consistent with our approach in Chapter 3, serves as the foundation for our continued exploration and optimization of neural network architectures in this chapter.

Table 4.1: CGP function set and range of different parameters to construct the genotype.

Node Function	Variation
ConvBlock	Output Channels = {1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024} Kernel Size = {1,3,5}
ResBlock	Output Channels = {1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024} Kernel Size = {1,3,5}
DeConvBlock	Output Channels = {1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024} Kernel Size = {1,3,5}
Pooling	Average and Max Pooling
MergeBlock	Concatenation and Summation

4.1.2 Evolutionary Algorithm

We revisit the evolutionary strategies that form the cornerstone of our methodology for evolving CNN architectures, initially detailed in Section 3.1.3. These strategies, rooted

in biological evolution principles, utilize mutation, recombination, and selection to progressively refine solutions. Specifically, we employ the $(\mu + \lambda)$ strategy [T.-Y. Huang and Chen \(2001\)](#), (with $\mu = 20$ and $\lambda = 16$), focusing on iteratively enhancing CNN designs for medical image segmentation. The overview of our algorithm is illustrated in Fig. 4.3

Representation

Each individual of the population corresponds to a distinct neural network architecture. The genetic structure of these individuals is laid out in a two-dimensional array, where each array row represents a network node. The first column in each row specifies the node's type (e.g., convolutional, pooling, fully connected layers), selected from a predefined set of operations defined in Table. 4.1. The remaining columns detail the node's connections, establishing the network's topology. This genetic encoding, as detailed in Section 3.1.3, enables the efficient exploration and design of intricate network architectures through genetic operations like mutation and crossover.

Initialization

We initialize a population of 20 individuals by random initialization and then further optimize this population by evolving it for 30 generations, considering the novelty metric of each individual as it's fitness objective. This method is referred to as the **rich initialization phase**, ensuring randomness in the initialization to uncover unique architectures that might elude conventional design strategies and promoting a diverse set of initial population. The population from the rich initialization phase serves as the initial population for the evolutionary algorithm. Additionally, identifying active nodes from the start focuses the evolution on effective architectural components, enhancing the process efficiency.

Parent Selection

We adopt tournament selection algorithm for parent selection with a window size of 5, meaning we form random subsets of five individuals from the population without replacement. From each subset, the individual demonstrating the highest performance is chosen as a parent, selecting 20% of the population for this role. This approach prioritizes individuals with superior fitness for reproduction, ensuring the propagation of advantageous genetic traits. Algorithm. 1 outlines the parent selection method.

Mutation

Mutation, our primary genetic operator, introduces variability into the population by altering node functions and connections. This process permits the exploration of diverse architectural possibilities without relying on crossover, enhancing the search for effective network configurations. Based on findings from [Real et al. \(2017\)](#), our algorithm emphasizes mutation, foregoing crossover due to its limited impact on performance enhancement.

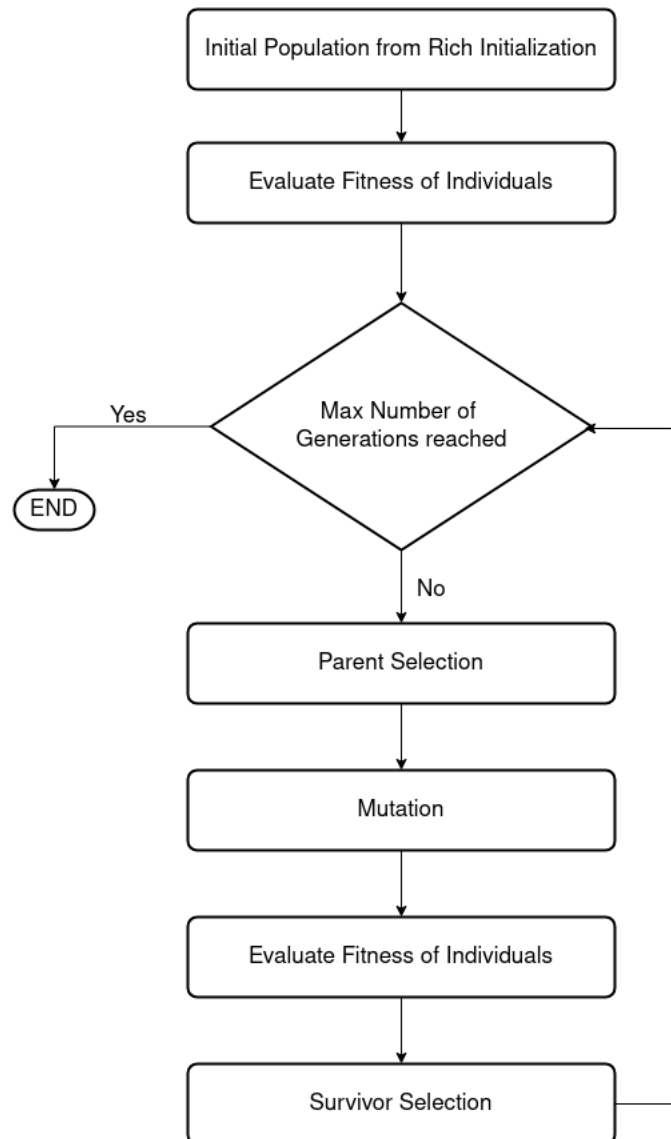


Figure 4.3: Overview of the evolution process

This strategy is important for uncovering novel and potentially more efficient network architectures. Through mutation, a layer's function can be randomly altered to another function from the set of functions indicated in Table 4.1. It also involves swapping the connections of a layer with those of its immediate neighbors, leading to variations in the information flow within the network, potentially resulting in novel architectures that could be more effective for the given task.

Survivor Selection

After offspring production, we select individuals for the next generation using the tournament selection algorithm without replacement with a window size of 5, incorporating both parents and offsprings. This method (outlined in Algorithm. 2) may favor a slightly less fit individual over a fitter one if the former has fewer trainable parameters and falls within a 0.05 fitness tolerance or has a higher novelty metric. This selection criteria balances fitness with model simplicity and population diversity, ensuring the advancement of promising and diverse architectures.

Fitness Evaluation

Fitness evaluations are important in identifying the effectiveness of candidate CNN models within our population. These evaluations guide the selection of optimal or near-optimal solutions by assessing each model's performance. We conduct these evaluations by training each CNN model on a designated training dataset for a predetermined number of epochs, 20 in our experiments. Subsequently, we assess the trained model's performance on a distinct validation dataset to ensure generalizability.

Selecting appropriate metrics like Dice Similarity Coefficient (DSC), Intersection over Union (IoU), Accuracy, F1 Measure, Sensitivity, and Specificity is very important to accurately assess the model performance in segmentation tasks. These metrics are chosen based on their ability to provide a comprehensive overview of a model's ability to segment medical images effectively. By understanding the strengths and limitations of each metric, we can better interpret the model's performance, ensuring the development of more accurate and reliable segmentation methods for medical imaging.

The Dice Similarity Coefficient (DSC), also known as the Sørensen–Dice index, calculated as in Eqn. 5, measures the similarity between two sets of data. Intersection over Union (IoU), on the other hand, calculates the area of overlap between the predicted segmentation and the ground truth, divided by the area of union between the predicted and ground truth segmentation as in Eqn. 6. The key difference is that DSC focuses on the overlap twice as much, potentially giving more weight to true positives, while IoU directly compares the overlap to the combined area, providing a more straightforward ratio of overlap to total area. DSC and IoU are recommended for medical image segmentation as they provide a more accurate representation of model performance by focusing on the true positive rate without being significantly biased by the large number of true negatives (background pixels).

For all the equations to compute the different metrics, TP represents True Positive (the number of pixels that have been properly classified), FP represents False Positives (the number of pixels that have been misclassified as a masked region) and FN represents False Negatives (the number of pixels that have been misclassified as the background).

$$DSC = \frac{2TP}{(2TP + FP + FN)} \quad (5)$$

$$IoU = \frac{TP}{TP + FP + FN} \quad (6)$$

Accuracy, calculated as in Eqn. 7, measures the proportion of true results (both true positives and true negatives) in the dataset. However, due to the prevalence of class imbalance in medical images, where the background class often dominates, accuracy can be misleading. It tends to show high values even when the model performs poorly in identifying the regions of interest (ROIs), such as vessels or tumors.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (7)$$

Sensitivity (Recall) and Specificity are metrics derived from the confusion matrix that assess the model's ability to correctly identify positives and negatives, and are calculated as in Eqn. 8 and Eqn. 9 respectively. Sensitivity is crucial for tasks like Tumor Segmentation where missing a positive case (a tumor) could have serious implications. Specificity, while less critical in the direct evaluation of segmentation performance, ensures that the model accurately identifies negative cases (non-tumor areas).

$$Sensitivity = \frac{TP}{TP + FN} \quad (8)$$

$$Specificity = \frac{TN}{TN + FP} \quad (9)$$

In summary, while all these metrics provide valuable insights into model performance, DSC and IoU are emphasized as particularly crucial for evaluating segmentation tasks in medical imaging due to their effectiveness in dealing with class imbalance and focusing on the segmentation accuracy of ROIs. We use the Adam optimizer, with a uniform learning rate of $1e - 4$ applied across all models to minimize the loss function. The fitness score of the model, reflective of its performance is calculated as the DSC. The Dice Score varies from 0 to 1, with higher scores indicating a more accurate overlap between the predicted and actual masks. The loss function for training and validation is calculated as a combination of the weighted Cross-Entropy loss and the Dice loss. The Dice Loss is calculated as in Eqn. 10.

$$DiceLoss = 1 - DSC = 1 - \frac{2TP}{(2TP + FP + FN)} \quad (10)$$

Termination

The evolutionary algorithm is terminated after a fixed number of generations. We chose to run the evolutionary process upto 200 generations. All the parameters used for the evolutionary algorithm are summarized in Table. 4.2

Table 4.2: Summary of Parameters used for the evolutionary algorithm

Parameter	Value
CGP grid size $N_r \times N_c$	5 x 30
Population size	20
Tournament Size	5
Mutation Probability	0.1
Optimizer	Adam
Learning Rate	Uniform ($1e - 4$)
Number of epochs trained during Fitness Evaluation	20
Loss Function	Weighted Cross-Entropy+Dice loss
Survivor Selection Fitness Tolerance	0.05
Number of Generations	200

4.1.3 Novelty Computation using GED

The novelty score of every individual is calculated by computing the graph similarity using Graph Neural Networks (SimGNN) [Bai et al. \(2020\)](#). The core idea is to use the structural and feature similarities between the graphs of neural network architectures, as computed by SimGNN, to quantify how novel or unique an architecture is relative to others in the population. SimGNN stands out for its ability to efficiently manage complex graph structures. It embeds graphs into a vector space, which allows the neural network to estimate the Graph Edit Distance (GED) between two graphs, a fundamental NP-hard problem in graph theory. This method offers a more efficient alternative to traditional GED calculations, which are resource-intensive in terms of the time taken to complete the calculations. The novelty for each individual is calculated as the average distance (GED) to its k nearest neighbors within the combined set of the population and scaling it by 100.

Our decision to use SimGNN for graph similarity computation, instead of the previously utilized cosine similarity metric, is because of its superior accuracy and reduced runtime. SimGNN’s neural network model is adept at navigating complex graph structures, giving it a notable edge. The effectiveness of cosine similarity, is based on graph-to-vector transformation, where finer details about the graphs are omitted and matrix operations are time intensive upon scaling. However, implementing SimGNN requires initial efforts in training the neural network, which can be a demanding task in terms of the time taken in training. But, once the model is trained, it can swiftly compute similarities. Fig. 4.4 shows the runtime comparison of SimGNN with other GED computation algorithms for three datasets (AIDS, Linux, IMDB). The figure source is from [Bai et al. \(2020\)](#).

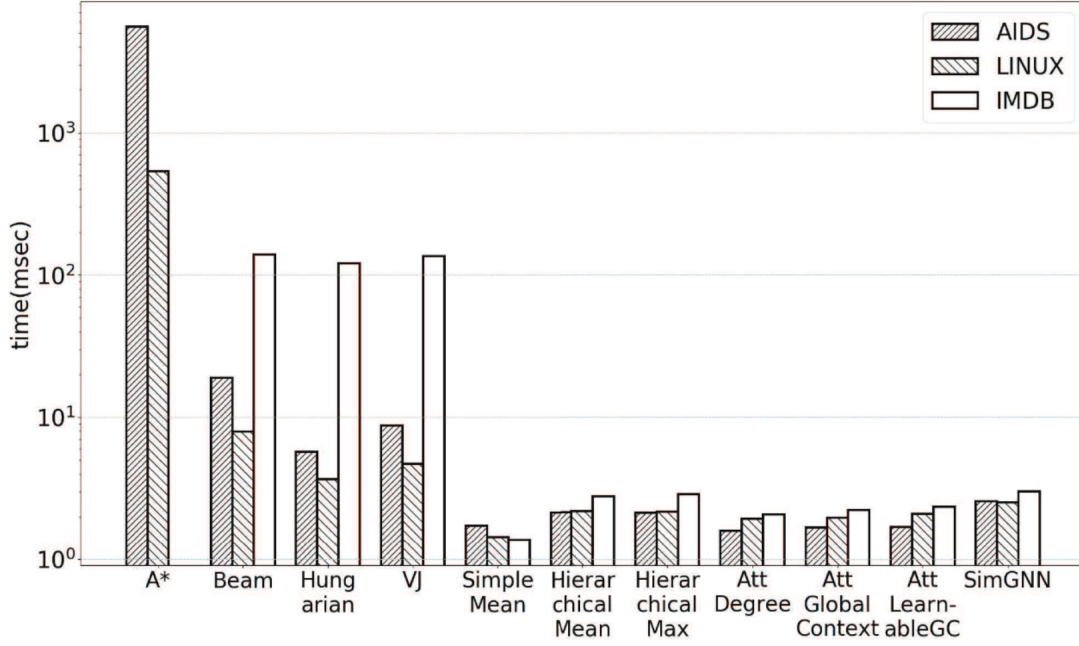


Figure 4.4: Runtime comparison of SimGNN with other benchmarked methods of GED computation over three datasets.

4.1.4 Vector Representation

We adopt a vector-based methodology to estimate the GPU memory usage for each layer of the CNN, incorporating key attributes such as layer depth, spatial dimensions, and channel count. This approach allows for an in-depth evaluation of the network’s architectural footprint.

The computation of the CNN’s memory demand involved assessing both activation and parameter memory. Activation memory for each layer is based on its spatial dimensions and channel count, reflecting the storage needs for the layer’s outputs. On the other hand, parameter memory calculation takes into account the inter-layer connections, factoring in the channels and spatial dimensions of each layer and those preceding it.

This strategy provides a viable means for approximating a CNN’s memory footprint, a critical consideration for deployment in GPU-constrained environments. It’s essential to recognize, however, that this estimation mainly addresses the memory used during operations and does not encompass the total memory usage during training phases, which includes gradients, optimizer states, and mini-batch data. Ultimately, the overall memory requirements are also dependent on variables like batch size, network depth, and architectural complexity.

A single layer, v_i , in a CNN architecture for 2D context can be represented as in Eqn. 11, where c_i represents the number of channels, x_i and y_i represent the adjusted spatial dimensions and d_i indicates the depth of the layer in the architecture from the input node.

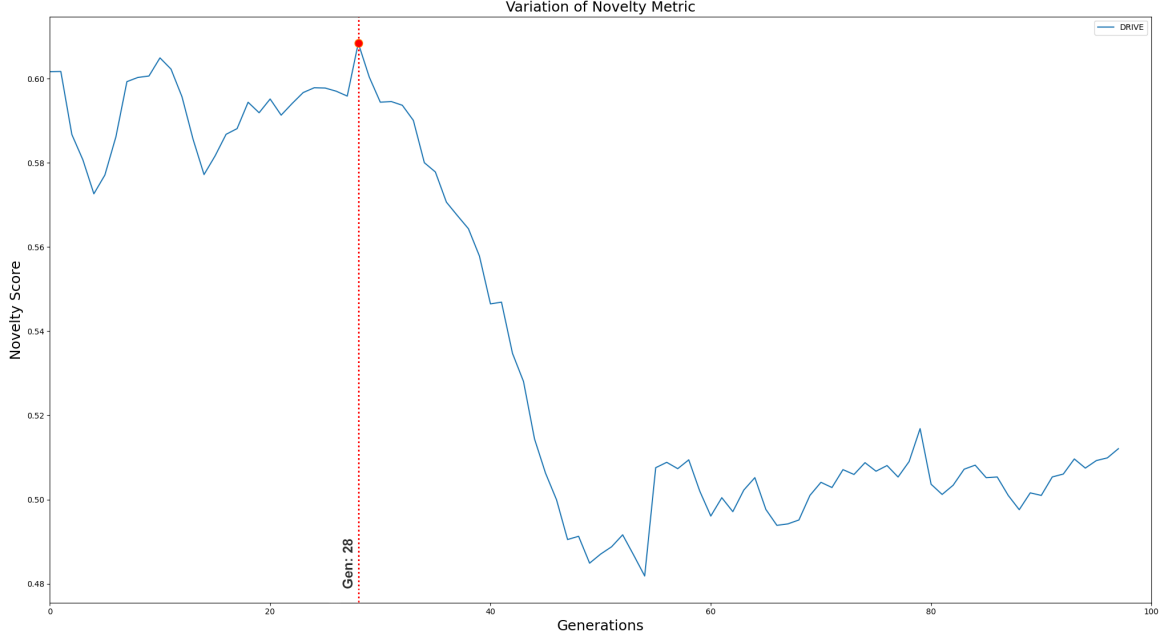


Figure 4.5: Mean novelty variation across generations during evolution of solutions for DRIVE dataset segmentation, with novelty scores on the y-axis and number of generations on the x-axis. The population with the highest mean novelty was sampled at the **28th** generation, as indicated by the dotted line.

$$v_i = \begin{bmatrix} c_i \\ x_i \\ y_i \\ d_i \end{bmatrix} \quad (11)$$

Identification of Local Optima

The population from the evolutionary algorithm with the highest mean novelty is sampled for the local optimization phase as indicated in Figures. 4.5 and 4.6. This choice of switching from exploration to exploitation by choosing the generation of highest mean novelty ensures that optimal diversity is maintained in the sampled population and the sub-optimal solutions belong to different regions of the fitness landscape. We optimize this sampled population to select potential solutions that are predominantly diverse or have a better performance in terms of their fitness score. This is performed by using the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [Deb et al. \(2002\)](#), prior to local search. This renowned algorithm excels in sorting solutions based on their novelty score and their fitness score, organizing them into different ranks. Our selection prioritizes solutions that demonstrate excellence in both these metrics, and we sample 4 solutions from the set of individuals. In scenarios where the new population space is constrained, a crowding-distance

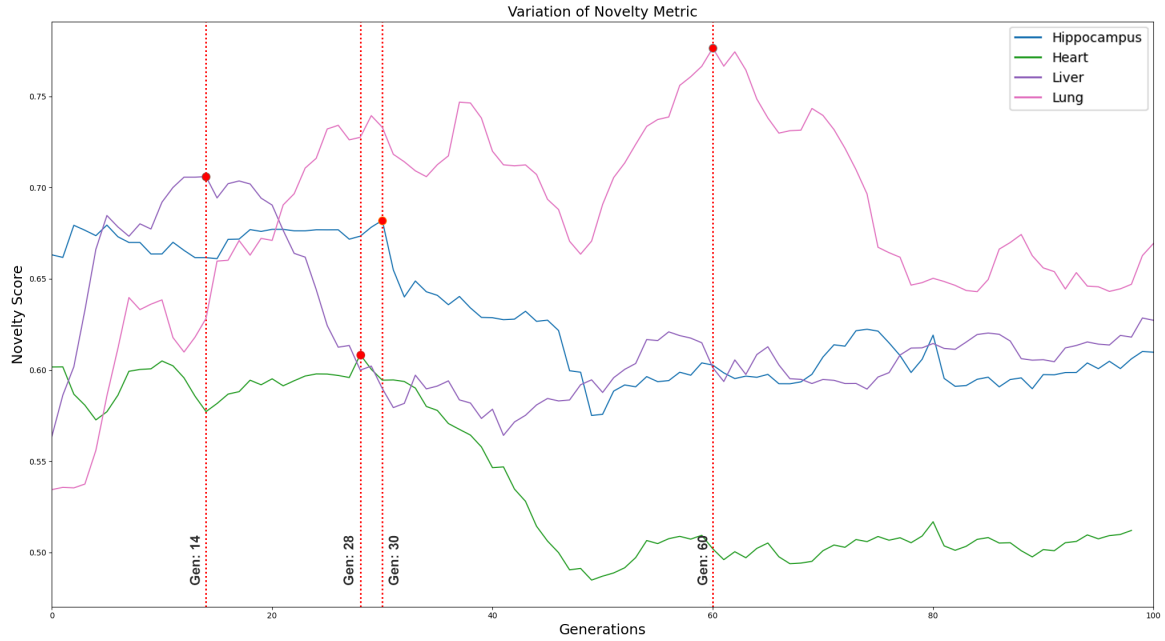


Figure 4.6: Mean novelty variation across generations during evolution of solutions for MSD Dataset Tasks, with novelty scores on the y-axis and number of generations on the x-axis. The blue curve represents the Hippocampus Model, and the population with highest mean novelty was sampled at the **30th** generation. The green curve represents the heart model, with the population of highest mean novelty found at the **28th** generation, while the purple curve represents the liver model with the highest mean novelty population sampled at the **14th** generation. The pink curve represents the lung model with the population sampled from the **60th** generation. Dotted Lines mark the discovery of generations with highest mean novelty.

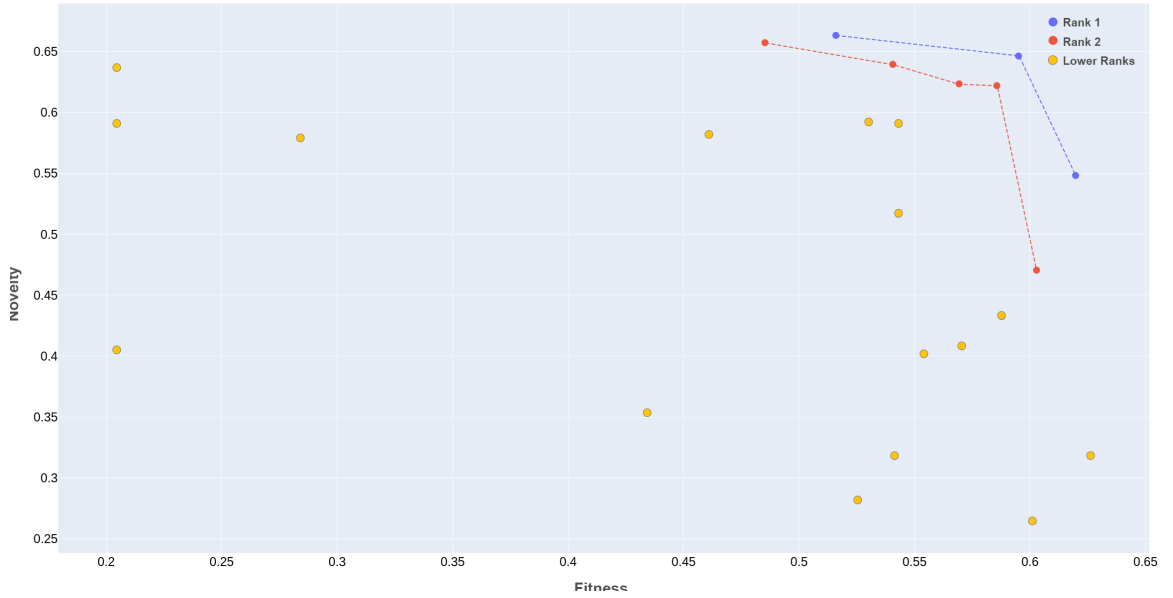


Figure 4.7: Population for the DRIVE segmentation task as ranked by the NSGA II algorithm, based on their fitness and novelty score. The novelty score is presented on the y-axis while the fitness is presented on the x-axis. Solutions in blue represent Rank 1, red represent Rank 2 and yellow represents further lower ranks.

strategy is employed to maintain diversity by favoring individuals from less dense regions of the solution space.

Figure. 4.7 shows the solutions ranked by the NSGA II algorithm for the DRIVE dataset segmentation task. Solutions with varying novelty metric indicate effective exploration and preserving and promoting diversity in the populations.

Figures. 4.8, 4.9, 4.10 and 4.11 shows the solutions ranked by the NSGA II algorithm for the MSD dataset hippocampus, heart, lung and liver segmentation tasks. The solutions again exhibit varying novelty scores, indicating the diversity of the population. The fitness values are however lower because each of these individuals are only trained for 20 epochs on the MRI and CT datasets.

4.1.5 Local Optimization

The four solutions sampled from the results of the NSGA II algorithm, for each of the segmentation task, is then fed into the the Simulated Annealing (SA) [Kirkpatrick et al. \(1983\)](#) algorithm. SA plays a major role in this hybrid approach by exploiting the local neighborhood of these sub optimal solutions to find a a better performing local optima. It starts with a wide-ranging search under a high initial temperature of 1000, gradually narrowing its focus as the temperature decreases according to a geometric cooling schedule with $\alpha = 0.9$. The change in temperature in every iteration is computed using Eqn. 12.

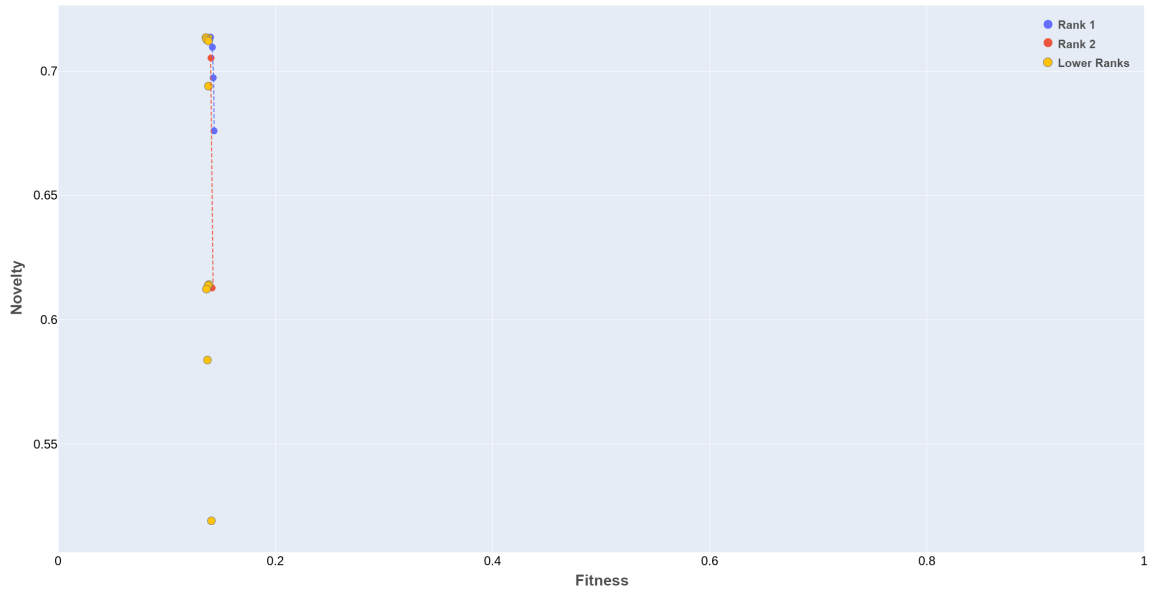


Figure 4.8: Population for the MSD Hippocampus segmentation task as ranked by the NSGA II algorithm, based on their fitness and novelty score. The novelty score is presented on the y-axis while the fitness is presented on the x-axis. Solutions in blue represent Rank 1, red represent Rank 2 and yellow represents further lower ranks.

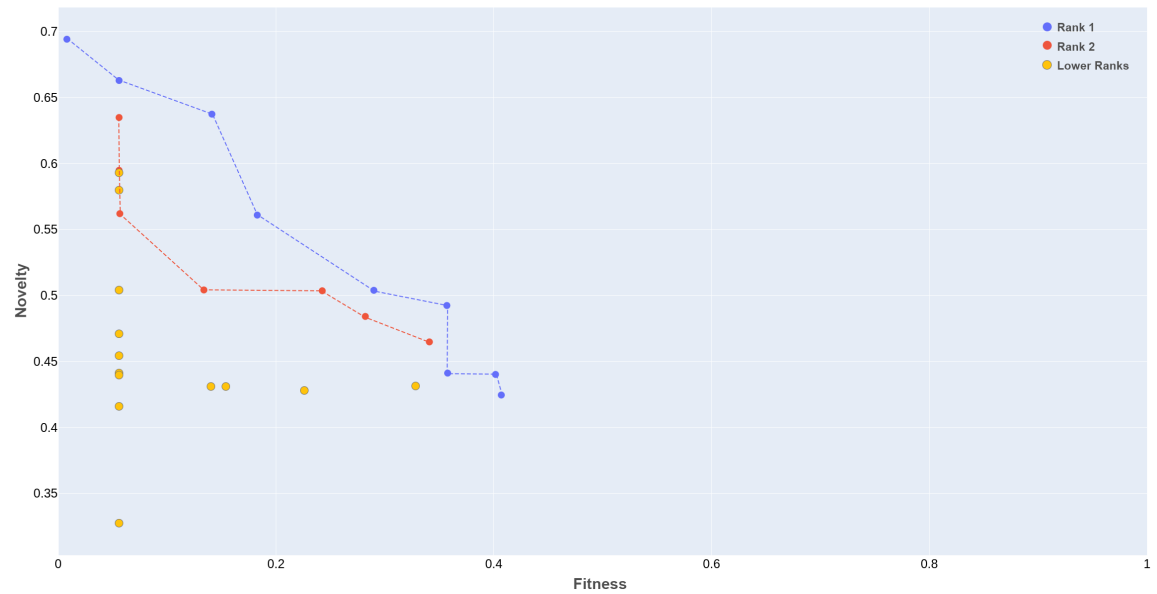


Figure 4.9: Population for the MSD Heart segmentation task as ranked by the NSGA II algorithm, based on their fitness and novelty score. The novelty score is presented on the y-axis while the fitness is presented on the x-axis. Solutions in blue represent Rank 1, red represent Rank 2 and yellow represents further lower ranks.

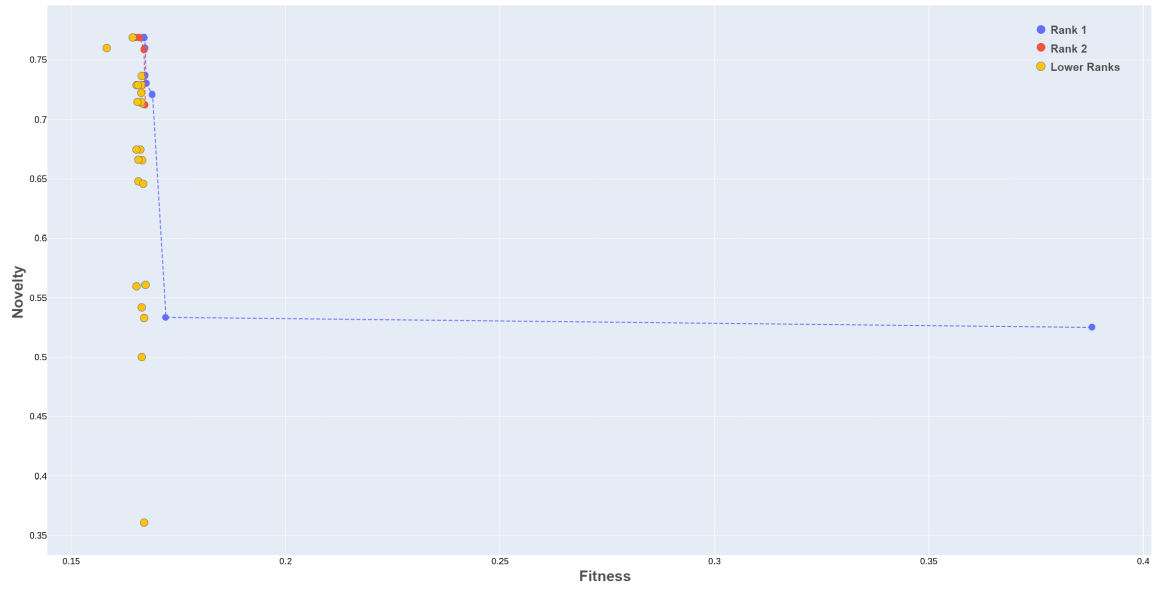


Figure 4.10: Population for the MSD Lung segmentation task as ranked by the NSGA II algorithm, based on their fitness and novelty score. The novelty score is presented on the y-axis while the fitness is presented on the x-axis. Solutions in blue represent Rank 1, red represent Rank 2 and yellow represents further lower ranks.

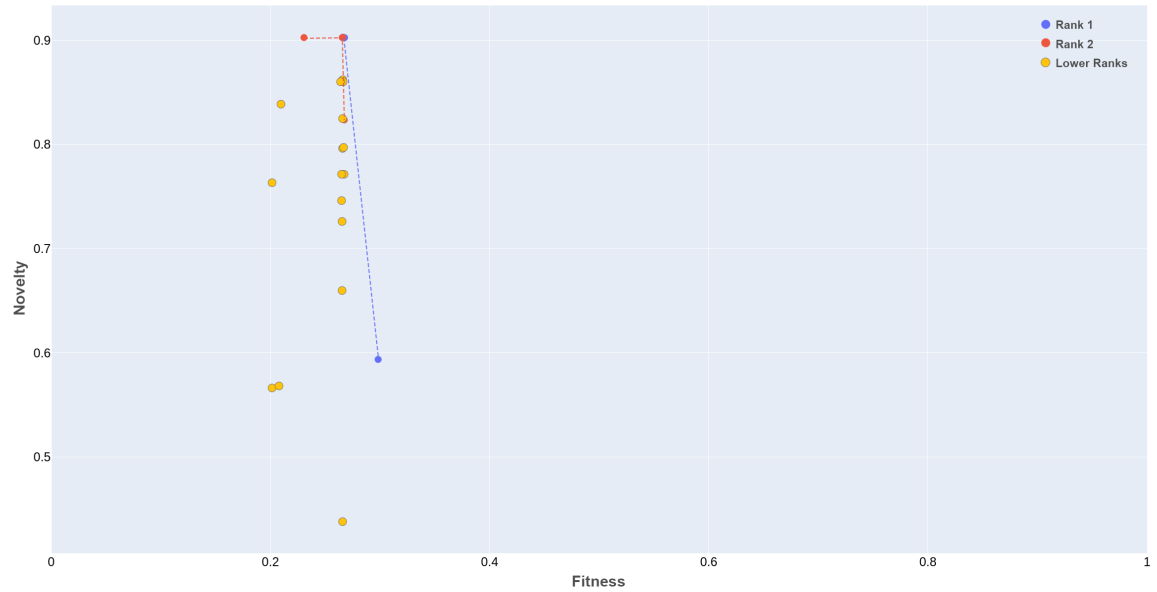


Figure 4.11: Population for the MSD Liver segmentation task as ranked by the NSGA II algorithm, based on their fitness and novelty score. The novelty score is presented on the y-axis while the fitness is presented on the x-axis. Solutions in blue represent Rank 1, red represent Rank 2 and yellow represents further lower ranks.

This method balances broad exploratory searches with the exploitation of promising solutions, guided by a cost function that assesses the potential of neighboring solutions through mutations. The cost function is calculated by the difference in fitness of the current solution and its neighbour. The algorithm’s progression is determined by a probabilistic acceptance of new solutions, as in Eqn. 13, considering both the solution quality and the exploration allowance provided by the current temperature. The process concludes once the temperature falls below 0.0000001, ensuring a thorough yet efficient optimization journey.

$$T = \alpha \times T \quad (12)$$

$$P = \exp\left(\frac{cost}{T}\right) \quad (13)$$

4.2 Dataset

The DRIVE and MSD datasets are benchmarked datasets in the realm of medical image segmentation. Their significance lies in their diversity, comprehensive annotations, and direct relevance to critical medical conditions. These datasets are indispensable tools for researchers and practitioners dedicated to advancing segmentation algorithms in the domain of medical imaging.

4.2.1 DRIVE Dataset (Digital Retinal Images for Vessel Extraction)

The DRIVE dataset [Staal et al. \(2004\)](#) holds high significance in the realm of medical image analysis, specifically tailored for retinal vessel segmentation. Comprising of 40 high-resolution color fundus images of $584 * 565$ pixels stored in JPEG format it is further accompanied by manual segmentation performed by ophthalmologists, adding a layer of ground truth for algorithm validation.

The dataset comprises a total of 40 images, divided into two equal subsets: 20 images for training and 20 for testing. To enrich the dataset and enhance its versatility, various data augmentation techniques were applied. This augmentation process generated both overlapping and non-overlapping patches, each with dimensions of $128 * 128$ pixels. As a result, the training dataset was expanded to encompass a total of 7,650 distinct training images, while an additional 850 images were specifically allocated for validation purposes. Pixel intensity normalization was applied as part of the data loading process to ensure consistency and facilitate effective training and evaluation.

4.2.2 Medical Segmentation Decathlon (MSD) Dataset

The Medical Segmentation Decathlon (MSD) dataset emerges as a versatile resource for evaluating 3D medical image segmentation algorithms. Comprising multiple segmentation tasks across various medical imaging modalities, including MRI and CT scans, the MSD dataset offers an opportunity to test and benchmark segmentation methodologies. It

covers a wide range of anatomical structures, including but not limited to the lung, liver, hippocampus, and heart. Within the MSD dataset, our study specifically focuses on MRI volumes of the heart and hippocampus, as well as CT volumes of the lungs and liver.

The MRI dataset pertaining to the heart comprises 30 MRI volumes, meticulously covering the entire heart during a single cardiac phase. The liver dataset consists of 201 contrast-enhanced CT images of patients. The hippocampus dataset comprised of MRI volumes acquired from a total of 90 healthy adult subjects and 105 adults diagnosed with a non-affective psychotic disorder. Furthermore, the lung dataset encompassed 96 CT scans of patients diagnosed with non-small cell lung cancer.

MRI scans often exhibit intensity non-uniformity, known as "bias" or "bias field," affecting intensity values across the image volume. To address this, we apply N4 Bias Field Correction which addresses the spatial intensity variations, improving tissue contrast. We further perform Z-score normalization in order to standardize the intensity values across different MRI scans. This normalization is crucial for algorithms that assume data normality. Subsequently, the normalized 3D volumes are cropped as specified in Table 4.3, converted to 2D slices, and augmented through flipping, rotation, and Contrast Limited Adaptive Histogram Equalization (CLAHE) for contrast enhancement.

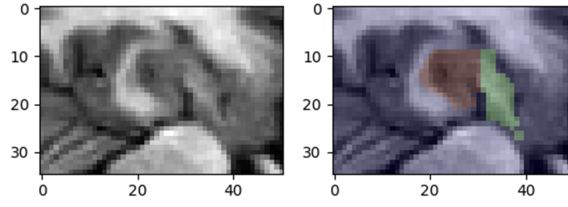
In the case of CT volumes, a similar preprocessing regimen is employed, but with slight variations due to the distinct characteristics of CT imaging. CT images are conventionally quantified in Hounsfield Units (HU), and different HU ranges correspond to various tissue types within the body. To tailor the CT volumes for our specific analysis requirements, clipping is performed to isolate the relevant HU range of -500 to 1500 for lungs and 40 to 75 for liver based on the reports in [Murphy \(2023\)](#). The HU thresholded volumes are Z score normalized, cropped as specified in Table 4.3 and converted to 2D slices and further augmented to increase the number of samples.

Table 4.3: Original and Augmented Dimensions for 2D slices of different tasks in the MSD dataset. The Hippocampus task has varying dimensions for every sample and hence the largest dimension is reported here.

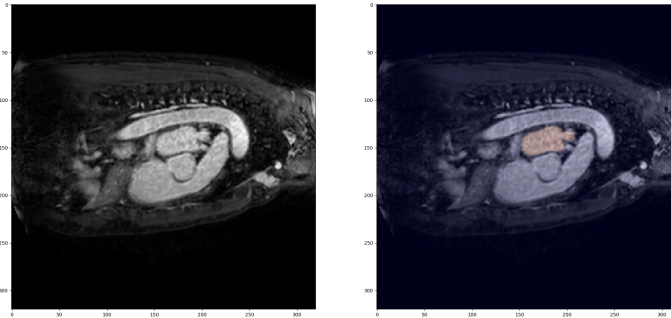
Task	Original Slice Dimension	Cropped Slice Dimension
Hippocampus	36 x 51	24 x 24
Heart	320 x 320	256 x 256
Lung	512 x 512	256 x 256
Liver	512 x 512	256 x 256

4.3 Experimental Setting

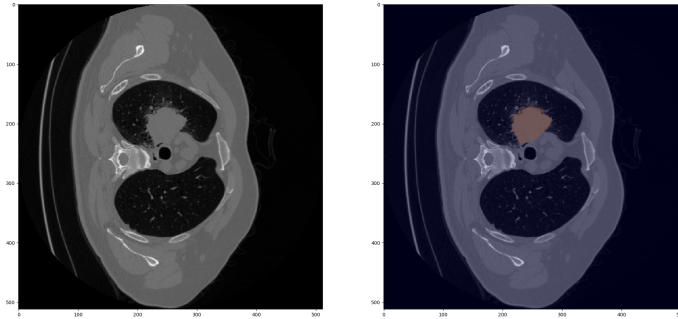
All our experiments are conducted on resources characterized by two Intel Xeon Gold 6238 Cascade Lake 2.10GHz CPUs with a total of 44 CPU cores, supported by four NVIDIA T4 Turing GPUs, and a substantial 187 GB of RAM.



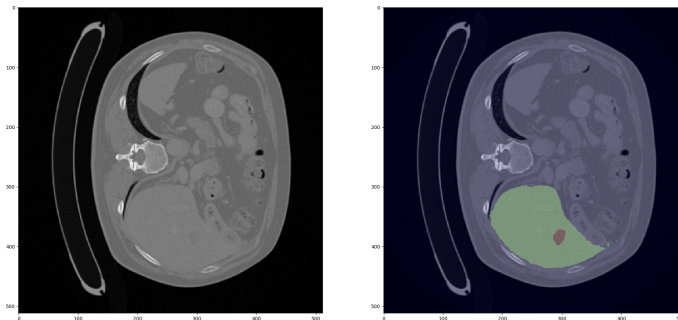
(a) MRI slice from Hippocampus task of dimension 35*51



(b) MRI slice from Heart task of dimension 320*320



(c) CT slice from Lung task of dimension 512*512



(d) CT slice from Liver task of dimension 512*512

Figure 4.12: Visualization of the (i) original slice of a input sample on the left with it's (ii) respective segmentation mask overlay on the right for the MSD dataset prior to preprocessing.

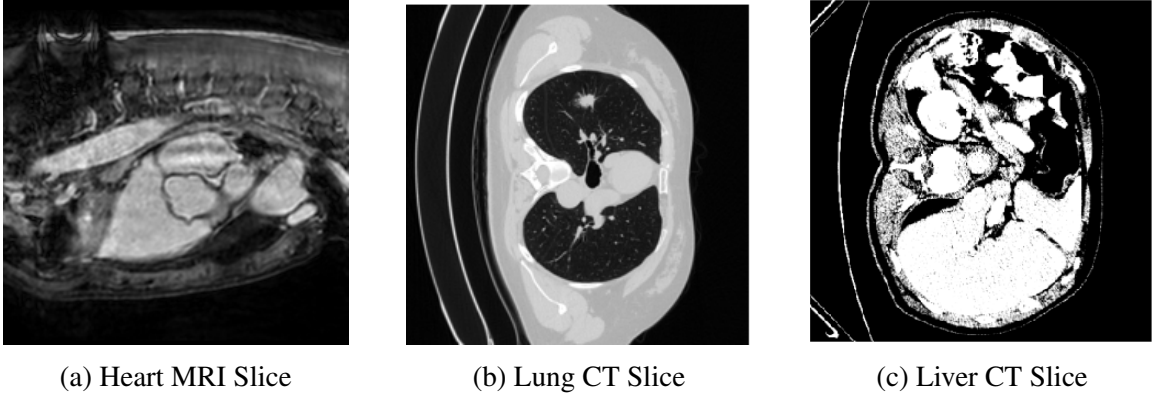


Figure 4.13: Slices from the (a) Heart, (b) Lung, (c) Liver task of the MSD dataset visualized after the preprocessing pipeline

The experimental process starts by creating a random population of 20 individuals with rich initialization. This population is enhanced through evolution, monitoring the average novelty per generation, ending after 200 generations. The generation with the highest novelty is selected for further analysis, optimizing them based on novelty and fitness. Four solutions are selected from this in increasing order of their ranks sorted by the NSGA II algorithm to undergo local optimization using the Simulated Annealing algorithm to find locally optimal solutions. The top two models from each segmentation task are then fully trained, and their segmentation results on an unseen test set are detailed in Section 4.4.

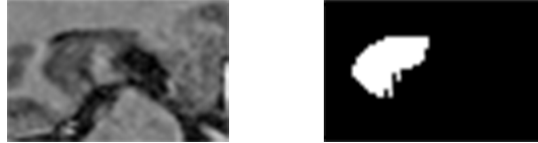
4.4 Results and Discussion

4.4.1 Segmentation Results

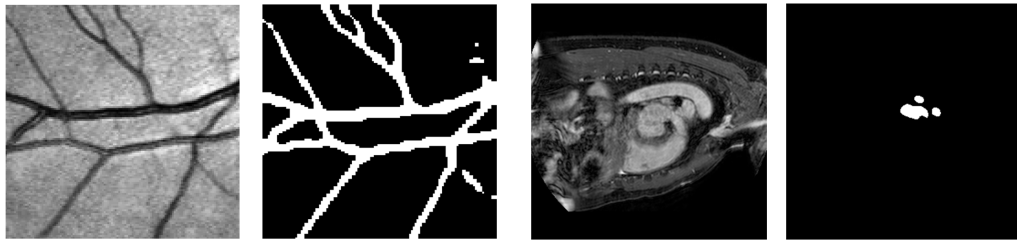
The two best performing models for each of the five segmentation tasks were trained on the complete training dataset for 1000 epochs with an early stopping parameter of 30 epochs, to mitigate overfitting. The performance of these models are summarised in Table 4.4 and the results of their segmentation are visualized in Figure 4.14.

The CNN models for all the five tasks achieve a good accuracy score, but our main focus here is the DSC and the IoU because the region of interest in these segmentation tasks is very small. Hence the data samples are highly imbalanced due to the dominance of the background pixels. Thus all the models learn the background pixels accurately. The IoU is a more strict measure of the model segmentation performance when compared to the DSC.

For the retinal vessel segmentation, our model achieves a DSC of 82.8% and is reported to have only 9 million trainable parameters. Of all the MSD segmentation tasks, the performance for the lung dataset is the most lacking in its DSC. The reason being that the lung nodules of interest are very small in dimension in most of the data samples. Thus this huge imbalance in the foreground and background pixels is only worsened by augmenting

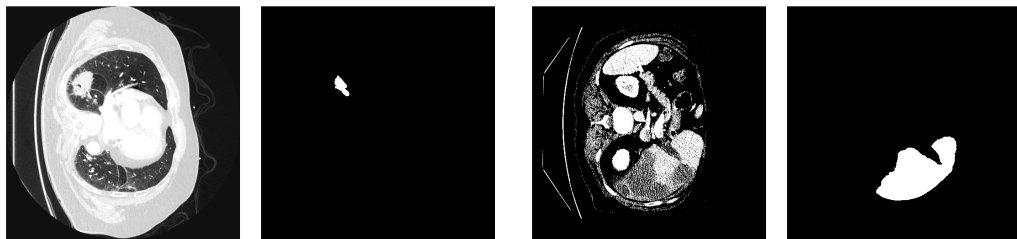


(a) Hippocampus Task Result



(b) DRIVE Result

(c) Heart Task Result



(d) Lung Task Result

(e) Liver Task Result

Figure 4.14: Visualization of the Input slice(left) and predicted mask(right) for the different segmentation tasks

Table 4.4: Performance of the top two models for each of the segmentation task. The best results for each task is shown in bold formatting.

Model	Dataset	Segmentation Task	DSC	IoU	Accuracy
M1	DRIVE	Retinal Vessel	0.828	0.716	0.981
M2	DRIVE	Retinal Vessel	0.814	0.736	0.978
M1	MSD	Hippocampus	0.857	0.748	0.986
M2	MSD	Hippocampus	0.849	0.719	0.983
M1	MSD	Heart	0.881	0.772	0.956
M2	MSD	Heart	0.924	0.823	0.962
M1	MSD	Lung	0.706	0.596	0.975
M2	MSD	Lung	0.713	0.628	0.978
M1	MSD	Liver	0.811	0.684	0.973
M2	MSD	Liver	0.798	0.592	0.976

the same slices. We might need to work on effective cropping and contrast enhancement strategies to improvise the accuracy of segmentation here. But even with a DSC of 71.3% our model is still comparable to the other benchmarked models.

4.4.2 Novelty Optimization of the Proposed Algorithm

The novelty metric computation of every individual within our evolutionary algorithm is based on the edit distances between their graph representations and those of others in the population, as determined by the SimGNN model. A traditional evolutionary algorithm tends to lose diversity over its progression due to the inherent mechanisms of selection, crossover, and mutation, which are designed to optimize the population towards the best solutions. This convergence can restrict the algorithm’s capacity to navigate and unveil superior solutions that lie beyond the currently explored areas.

But the proposed algorithm, is designed to favor the evolution of more diverse individuals in the population. We observe a significant enhancement in the algorithm’s exploratory capability through the implementation of the rich initialization phase. This approach not only seeds the initial population with a broad array of potential solutions but also aids the evolutionary process for a more dynamic exploration of the solution space, thereby preserving diversity. Throughout the survivor selection algorithm, a less fit individual can be chosen if it’s novelty metric is higher, provided the difference in fitness is within the specified tolerance range.

The switching from exploration to the exploitation samples the population with the

highest mean novelty in the evolutionary phase. This ensures that the algorithm chooses sub optimal solutions in different regions of the fitness landscape and exploit their local neighborhood to reach better solutions. The diversity in the population is visualized while projecting the population ranked by the NSGA II algorithm, in the novelty-fitness space. For each of the different segmentation tasks it can be seen from Figures. 4.7, 4.8, 4.9, 4.10 and 4.11 that the solutions have varying novelty scores, implying that they edit distance between them is large and that they belong to different regions of the solution space.

The algorithm proposed here significantly advances the design of CNNs by generating diverse solutions that exhibit competitive performance. This diversity is crucial, as manually designing CNNs for specific tasks may overlook potentially superior architectures. The human designer might not consider some high-performing architectures, highlighting the algorithm’s ability to explore and identify effective solutions beyond human intuition. This capability ensures a broader exploration of the solution space, likely leading to innovations and improvements in CNN design and application.

4.4.3 Number of Fitness Evaluations

Table 4.5: Number of fitness computations incurred in obtaining the best performing model for each of the segmentation tasks. F1 indicates the number of computations for the evolutionary phase and F2 indicates the number of computations for the local optimization phase.

Model	Dataset	F1	F2	Total Fitness Computations = F1+F2
M1	DRIVE	452	344	796
M1	MSD (Hippocampus)	484	220	704
M2	MSD (Heart)	452	580	1032
M2	MSD (Lung)	964	636	1600
M1	MSD (Liver)	228	436	664

Given that fitness evaluations are the most significant contributors to GPU time and memory usage, we measure our algorithm’s computational overhead in terms of the number of fitness evaluations conducted during the evolution and optimization phases for various segmentation tasks. Each task begins with a rich initialization phase, enhancing the novelty of the initial population for the evolutionary algorithm for 30 epochs. This enhanced population then enters the evolution phase, during which we track the mean novelty of each generation and select the one with the highest mean novelty for further processing.

In the case of the DRIVE dataset, the optimal model was found at the 86th iteration as shown in Fig. 4.15, culminating in a total of 796 fitness evaluations.

For the MSD segmentation tasks, Fig. 4.16 shows the progress of the SA algorithm and the specific iterations at which the solutions for the four tasks were sampled from. For the

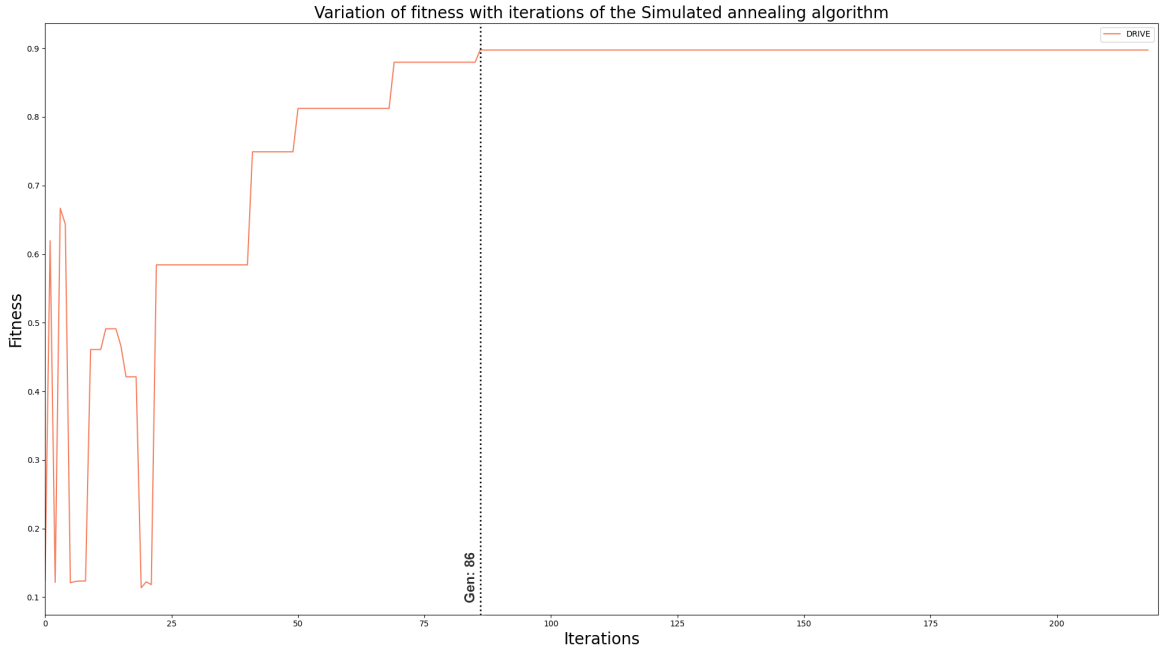


Figure 4.15: Fitness variation across iterations in the simulated annealing algorithm during optimization of the model for DRIVE dataset segmentation task, with dotted line marking the discovery of the optimal model at the **86th** iteration.

Hippocampus task, the best solution was found at the 30th iteration, resulting in 704 fitness evaluations. For the heart task, the best solution was found at the 28th iteration, resulting in 1032 fitness evaluations. For the lung and liver tasks, the best solutions were found at the 60th and 14th iterations, resulting in 1600 and 664 fitness evaluations.

Four individuals were selected for each task from the multi objective optimization algorithm to be input to the SA algorithm. Upon computing the variance in the number of fitness evaluations required to reach the best performing solution, it was found to be less than 250 in all the cases. Thus the algorithm efficiently identifies optimal solutions using fewer fitness evaluations, reflecting its computational efficiency compared to traditional evolutionary methods. By leveraging local optimization techniques such as Simulated Annealing, it achieves notable computational savings, making it highly effective for practical applications.

4.4.4 Comparison with related works

A further detailed comparison of the evaluation metrics between our best performing model on the DRIVE dataset and the state of art models is presented in Table. 4.6.

Our best performing model on the DRIVE dataset, while not outperforming the RV-GAN in segmentation metrics, is still comparable in performance to the top benchmarked models. The proposed algorithm designs and optimizes CNN models with minimal human intervention, primarily during the initial setup phase. This approach results in a capable

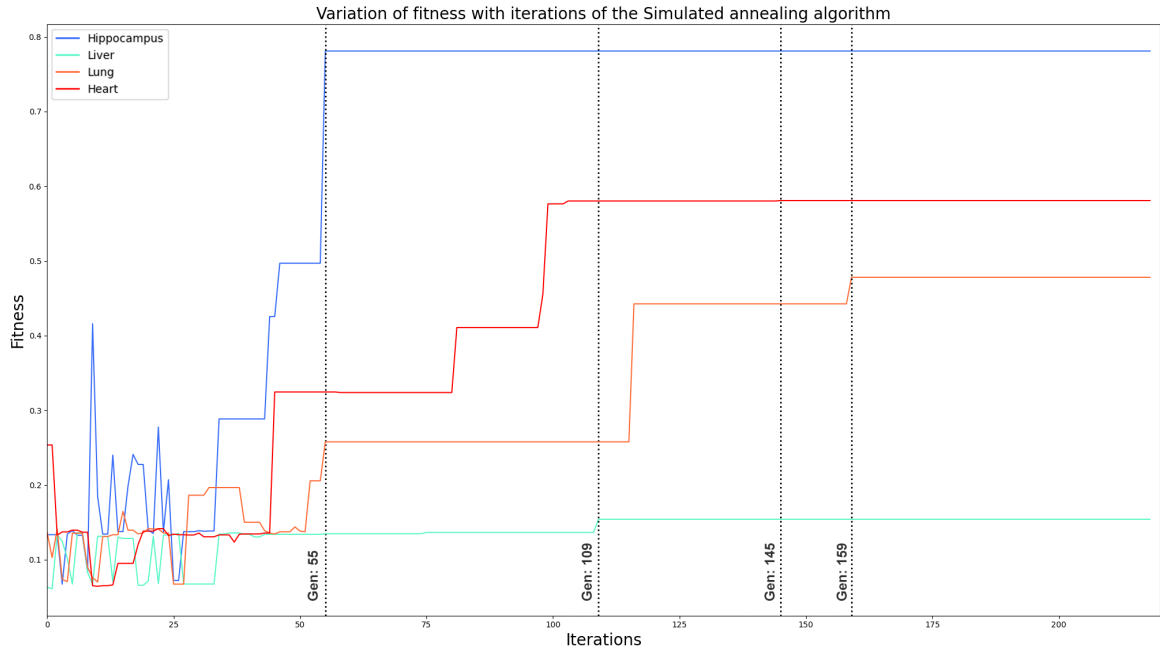


Figure 4.16: Fitness variation across iterations in the simulated annealing algorithm during optimization of the models for MSD Dataset Tasks, with Dotted Lines Marking the Discovery of Optimal Models. The blue curve represents the Hippocampus Model, the red curve the heart model, the cyan curve the liver model and the orange curve the lung model.

Table 4.6: Results (%) of Retinal Vessel Segmentation on DRIVE dataset. The accuracy, dice score coefficient and the intersection of union metrics for all the benchmarked models are tabulated in decreasing order of DSC. The best results are in bold.

Model	Acc	DSC	IoU
RV-GAN Kamran et al. (2021)	97.90	86.90	97.62
FR-UNet W. Liu et al. (2022)	97.05	83.16	71.20
Fan, Wei, Zhu, Mo, and Li (2020)	97.02	82.97	-
MERIT-GCASCAD Rahman and Marculescu (2023a)	97.07	82.90	70.81
Our Model - Drive Model 1	98.10	82.80	71.60
MERIT-CASCAD Rahman and Marculescu (2023b)	96.89	82.21	69.08
PVT-GCASCAD Rahman and Marculescu (2023a)	96.89	82.10	69.70
PVT-CASCAD Rahman and Marculescu (2023c)	96.79	81.73	69.10
UNet Ronneberger et al. (2015)	96.78	81.41	68.64
UNet++ Zoph et al. (2018)	96.79	81.14	68.27
Attention UNet Oktay et al. (2018)	96.62	80.39	67.21
PVTv2-b2 (only) W. Wang et al. (2022)	96.24	79.14	65.48

model that not only performs competitively but also significantly reduces training time. Our model, with its 9 million parameters, achieves an impressive 98.10% accuracy, outperforming the current models and completes its training on the DRIVE dataset in just an average of 10-12 hours using a NVIDIA T4 GPU. This highlights adaptability of our method in generating effective CNN architectures with performance comparable to other state of the art models for complex tasks like retinal vessel segmentation.

In contrast, RV-GAN exhibits outstanding performance on the DRIVE dataset for retinal vessel segmentation, boasting a high DSC (86.90%) and mean IoU (97.62%) . This model benefits from a specialized design that incorporates a multi-scale GAN approach, tailored specifically for microvessel detection. However, RV-GAN’s performance comes with considerable computational demands, requiring extensive training time (reported to be 24-48 hours in [Kamran et al. \(2021\)](#)) and significant human input for design and hyperparameter optimization. Despite these requirements, RV-GAN sets a benchmark in segmentation accuracy, presenting a trade-off between performance and the model training time compared to our evolutionary algorithm-derived model.

In their research, [Fan et al. \(2020\)](#) introduced a modified evolutionary strategy for refining an encoder-decoder architecture aimed at segmenting retinal vessels in the DRIVE dataset. The methodology in [Fan et al. \(2020\)](#) surpasses the performance of Drive Model 1 in the DSC metric. This evolutionary algorithm trains each individual in the population for 100 epochs with a batch size of 1 to calculate their fitness scores, concluding after 500 generations. This extensive number of fitness evaluations highlights the evolutionary phase’s computational demand. The diversity within the population, critical for the evolutionary algorithm’s search capability, tends to diminish over time due to genetic operators. This reduction in diversity can lead to decreased exploration potential, restricting the algorithm’s ability to generate novel solutions across the fitness landscape’s various regions.

Our proposed hybrid algorithm, is designed to overcome these limitations. Instead of merely pursuing highly fit solutions, our strategy emphasizes exploratory behavior and then thorough optimization of sub-optimal solutions, producing models with competitive performance. This objective is supported by a rich initialization phase and the implementation of the crowding operator in the NSGA II algorithm. By focusing on sampling solutions from less crowded areas and employing multi-objective optimization, our methodology ensures a broader exploration of the solution space, fostering the generation of diverse and competitive solutions.

The current benchmarked models for the MSD dataset and their respective DSC scores are compared in Table. 4.7.

For the four segmentation tasks in the MSD dataset, our best performing models have a DSC metric of 0.81(Liver Model 1), 0.713(Lung model 2), 0.857(Hippocampus Model 1) and 0.924(Heart Model 2). Although our models do not rival the performance of the state of art models, they do achieve comparable performances.

The standout model across the board for all ten tasks within the Medical Segmentation Decathlon (MSD) dataset is the Swin UNETR [Tang et al. \(2022\)](#), which boasts an impressive average Dice Score of 0.7668. This model integrates the cutting-edge Vision

Table 4.7: Comparison of the DSC metric different tasks of the MSD dataset of the current state of the art models. Some of the models do not report the results for all the four tasks and are hence left blank

Model	Liver	Lung	Hippocampus	Heart
Swin UNETR Tang et al. (2022)	0.855	0.774	-	-
DiNTS He, Yang, Roth, Zhao, and Xu (2021)	0.8499	0.772	0.8916	0.9635
nnUNet Isensee et al. (2018)	0.8586	0.7602	0.8946	0.9674
Trans VW Haghighi, Hosseinzadeh Taher, Zhou, Gotway, and Liang (2021)	0.864	0.745		
C2FNAS Yu et al. (2020)	0.839	0.704	.8796	0.9249
Models Gen. Zhou, Sodha, Pang, Gotway, and Liang (2021)	0.866	0.745	-	-
Kim et al Kim et al. (2019)	0.8361	0.6251	0.8942	0.9644
Our Model	0.810	0.713	0.857	0.924

Transformer (ViT) technology, utilizing self-attention mechanisms to adeptly capture long-range dependencies within the data. It's important to highlight that the complexity of a vision transformer comes with its array of self-attention layers and extensive parameters, necessitating a significant volume of data to ensure its effective generalization capabilities. The Vision Transformer's intricate function set and architecture present a stark contrast to the more streamlined UNet models. UNet architectures are celebrated for their memory efficiency and their flexibility in achieving commendable results even with considerably smaller datasets. Additionally, it's critical to acknowledge that the Swin UNETR's reported results were achieved following a pre-training phase involving various proxy tasks. Consequently, it is fair to assert that our algorithm, which does not rely on pre-training, still manages to design models of segmentation performance comparable with the Swin UNETR, while maintaining a simpler architecture and function set. This aspect highlights the effectiveness of our algorithm, emphasizing its capability to achieve competitive results without the need for complex pre-training processes, thereby offering a more straightforward approach to designing CNN models for segmentation tasks.

nnU-Net is renowned for its high performance across various medical image segmentation challenges, attributed to its dynamic architecture and extensive data augmentation strategies. With its adaptive framework, nnU-Net is designed to maximize performance across a wide range of segmentation tasks without explicit pre-training. However, its performance can significantly benefit from large datasets, given its capability to tailor its architecture and processing pipeline based on the data characteristics. Again, a significant advantage of our proposed algorithm is its ability to perform competitively without the necessity for extensive pre-training on multiple proxy tasks. This characteristic is particularly valuable in scenarios where data availability is limited or where computational resources for extensive pre-training are scarce. While nnU-Net's adaptability and performance are benchmarks in the field, our algorithm's comparable performance, despite its streamlined

architecture and lack of pre-training, showcases its potential as an effective tool for medical image segmentation.

4.5 Conclusions and Future Work

In this chapter, we have expanded the scope of our hybrid evolutionary algorithm, applying it to the domain of both 2D and 3D image segmentation. Our rigorous assessments across benchmark datasets have illuminated the algorithm’s computational overhead and its capacity to evolve competitive solutions, although not surpassing state-of-the-art models. Notably, on the DRIVE dataset, our models achieved Dice Similarity Coefficients (DSC) of 0.828 and 0.814, Intersection over Union (IoU) scores of 0.716 and 0.736, and accuracy levels of 98.1% and 97.8% for Drive Model 1 and Drive Model 2, respectively, with the models having trainable parameters counts of 9 million and 27 million. These results highlight the algorithm’s efficiency in evolving competitive solutions that, while not outperforming the state-of-the-art, are capable of near state-of-the-art performance.

For the Medical Segmentation Decathlon (MSD) dataset, covering a range of segmentation tasks including the hippocampus, heart, lung, and liver, the models produced DSC scores up to 0.857 for the hippocampus and 0.924 for the heart, demonstrating the algorithm’s adaptability to both 2D and 3D segmentation challenges. This adaptability is further evidenced by achieving accuracy levels as high as 98.6% for the hippocampus task and 96.2% for the heart task, indicating the algorithm’s robustness across varied medical imaging modalities.

Our algorithm’s design promotes diversity and efficient exploration of the fitness landscape. By efficiently managing the number of fitness evaluations, our approach demonstrates a marked improvement in computational efficiency, making it a viable and innovative alternative for designing CNN architectures for complex image segmentation tasks. However, in this work we quantify the computational effort in terms of the number of fitness evaluations. But we would further like to extend this and solidify our claim by computing and comparing the Floating Point Operations (FLOPs) for each of our model and the benchmarked models. There is also potential for further optimization and exploration within the domains of data preprocessing, augmentation techniques, and the integration of more complex layers within the CGP function set. Integrating transfer learning with local optimization offers a promising pathway to overcoming dataset scarcity challenges, particularly in 3D segmentation, by leveraging knowledge transfer across tasks to streamline the evolutionary process and achieve more efficient and robust solutions.

Chapter 5

Conclusions and Future Work

Our research has led to the development of a hybrid evolutionary algorithm aimed at designing Convolutional Neural Networks (CNNs) for image classification and medical image segmentation tasks. This algorithm combines Cartesian Genetic Programming (CGP) with innovative optimization techniques like Novelty Search and Simulated Annealing (SA) to efficiently search through the design space of neural networks. By introducing a novel approach to measure the novelty of network architectures, we've made it possible to explore unique and potentially more effective designs without the need for exhaustive computational resources.

Our experiments, detailed across Chapters 3 and 4, revealed the algorithm's effectiveness in generating competitive models that closely approach the performance of current state-of-the-art methods. Specifically, Chapter 3 focused on the optimization of CNN architectures for image classification, highlighting the algorithm's proficiency in achieving commendable accuracy for the MNIST dataset. Extending this approach to image segmentation tasks in Chapter 4, the algorithm was subjected to evaluations on benchmark datasets such as DRIVE and the Medical Segmentation Decathlon (MSD). We achieved notable performance on the DRIVE dataset, with Dice Similarity Coefficient (DSC) scores of up to 0.828 and Intersection over Union (IoU) scores of up to 0.716. For the MSD dataset, encompassing various medical imaging modalities, our top models reported DSC scores as high as 0.924 for heart segmentation.

The proposed algorithm maintains a balance between sufficient exploration and efficient exploitation and maintains the diversity of the population throughout. We demonstrate this by projecting the ranked solutions from the NSGA II algorithm onto the fitness-novelty space. The variation of the novelty metric of the solutions demonstrates that they are spread out on different regions of the fitness landscape. But in the direction of future work, we'd like to quantify this by computing the actual GED between the solutions for the different tasks and reporting them.

A key aspect of our algorithm is its minimal reliance on manual effort and its ability to uncover promising solutions that traditional search methods might overlook. Our algorithm's capacity to generate diverse, competitive solutions with minimal manual intervention marks a significant advancement in the automation of CNN design for image segmentation and beyond.

A majority of the state-of-art models in the segmentation task are either data hungry and parameter heavy transformers or a fine tuned UNet structure that requires extensive pre-training on other datasets. We claim that our proposed method reduces the computational effort in discovering a diverse set of solutions by quantifying it in terms of the number of fitness evaluations performed. There is a clear reduction in the number of fitness evaluations when compared to traditional evolutionary algorithms. But to make a fair comparison with other models in terms of the computational effort involved, we'd like to compute and report the Floating Point Operations (FLOPs) for each of these models as a future extension. FLOPs provides a quantitative measure of a model's computational complexity, which is essential for comparing efficiency, planning deployment, and optimizing performance.

Other significant improvements that can be pursued as future work are, refining data preprocessing and augmentation techniques. By customizing these processes to better suit the intricacies of various image types and segmentation tasks, we can significantly improve the model's adaptability and performance across a broader range of applications.

Integrating more complex layers within the Cartesian Genetic Programming (CGP) function set represents another promising area for development. This approach could facilitate the evolution of more sophisticated CNN architectures, potentially leading to breakthroughs in tasks that current models find challenging.

Addressing the scarcity of extensive datasets, particularly for 3D segmentation, is a hurdle that remains. Here, combining transfer learning with local optimization emerges as a compelling strategy. By effectively leveraging the knowledge gained from one dataset to inform the evolutionary process on subsequent datasets, we could significantly streamline the algorithm's efficiency and enhance its performance. This "knowledge transfer" could mitigate the challenges posed by dataset scarcity and the high dimensionality of 3D data.

A novel aspect we plan to explore involves devising an automatic switching mechanism that transitions the algorithm from the evolutionary phase to the optimization phase. This mechanism would monitor the mean novelty of the population, employing a patience parameter to determine the optimal moment for transition. Such a feature could optimize the algorithm's computational resources, focusing on local optimization efforts when the evolutionary process ceases to yield substantial novelty improvements. This dynamic adjustment could significantly refine the algorithm's efficiency, reducing computational overhead and focusing efforts where they are most productive.

These proposed enhancements not only aim to fortify the algorithm's foundational strengths but also expand its versatility and effectiveness. As we move forward, our goal is to not only refine the existing framework but also to explore new paradigms in the evolutionary design of CNNs, pushing the boundaries of what is achievable in automated model generation and optimization.

References

- Al-Sahaf, H., Bi, Y., Chen, Q., Lensen, A., Mei, Y., Sun, Y., ... Zhang, M. (2019). A survey on evolutionary machine learning. *Journal of the Royal Society of New Zealand*, 49(2), 205-228. Retrieved from <https://doi.org/10.1080/03036758.2019.1609052> doi: 10.1080/03036758.2019.1609052
- Bai, Y., Ding, H., Bian, S., Chen, T., Sun, Y., & Wang, W. (2020). *Simgnn: A neural network approach to fast graph similarity computation*.
- Baker, B., Gupta, O., Naik, N., & Raskar, R. (2017). Designing neural network architectures using reinforcement learning. In *International conference on learning representations*. Retrieved from <https://openreview.net/forum?id=S1c2cvqee>
- Baldeon Calisto, M., & Lai-Yuen, S. K. (2020). Adaen-net: An ensemble of adaptive 2d-3d fully convolutional networks for medical image segmentation. *Neural Networks*, 126, 76-94. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0893608020300848> doi: <https://doi.org/10.1016/j.neunet.2020.03.007>
- Baldeon-Calisto, M., & Lai-Yuen, S. (2019, 04). Adaresu-net: Multiobjective adaptive convolutional neural network for medical image segmentation. *Neurocomputing*, 392. doi: 10.1016/j.neucom.2019.01.110
- Baldominos, A., Saez, Y., & Isasi, P. (2018). Evolutionary convolutional neural networks: An application to handwriting recognition. *Neurocomputing*, 283, 38-52. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0925231217319112> doi: <https://doi.org/10.1016/j.neucom.2017.12.049>
- Bergstra, J., Yamins, D., & Cox, D. D. (2012). *Making a science of model search*.
- Cai, H., Chen, T., Zhang, W., Yu, Y., & Wang, J. (2018, Apr.). Efficient architecture search by network transformation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1). Retrieved from <https://ojs.aaai.org/index.php/AAAI/article/view/11709> doi: 10.1609/aaai.v32i1.11709
- Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020, 11). End-to-end object detection with transformers. In (p. 213-229). doi: 10.1007/978-3-030-58452-8_13
- Chollet, F., et al. (2015). *Keras*. GitHub. Retrieved from <https://github.com/fchollet/keras>
- Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions*

- on *Information Theory*, 13(1), 21-27. doi: 10.1109/TIT.1967.1053964
- Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Trans. Evol. Comput.*, 6, 182-197. Retrieved from <https://api.semanticscholar.org/CorpusID:9914171>
- De Jong, K. (2006). *Evolutionary computation – a unified approach*.
- Domhan, T., Springenberg, J. T., & Hutter, F. (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *International joint conference on artificial intelligence*. Retrieved from <https://api.semanticscholar.org/CorpusID:369457>
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... Houlsby, N. (2021). *An image is worth 16x16 words: Transformers for image recognition at scale*.
- Eiben, A., & Smith, J. (2003). *Introduction to evolutionary computing* (Vol. 45). doi: 10.1007/978-3-662-05094-1
- Fan, Z., Wei, J., Zhu, G., Mo, J., & Li, W. (2020). *Evolutionary neural architecture search for retinal vessel segmentation*.
- Fernandes Junior, F. E., & Yen, G. (2019, 06). Particle swarm optimization of deep neural networks architectures for image classification. *Swarm and Evolutionary Computation*, 49, 62-74. doi: 10.1016/j.swevo.2019.05.010
- Floreano, D., Dürr, P., & Mattiussi, C. (2008). Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1), 47-62. Retrieved from <http://infoscience.epfl.ch/record/112676> doi: <https://doi.org/10.1007/s12065-007-0002-4>
- Ghillani, D. (2022, 09). *Deep learning and artificial intelligence framework to improve the cyber security*. doi: 10.22541/au.166379475.54266021/v1
- Haghighi, F., Hosseinzadeh Taher, M. R., Zhou, Z., Gotway, M., & Liang, J. (2021, 02). *Transferable visual words: Exploiting the semantics of anatomical patterns for self-supervised learning*.
- Hassanzadeh, T., Essam, D., & Sarker, R. (2020, 11). 2d to 3d evolutionary deep convolutional neural networks for medical image segmentation. *IEEE transactions on medical imaging, PP*. doi: 10.1109/TMI.2020.3035555
- He, Y., Yang, D., Roth, H., Zhao, C., & Xu, D. (2021). *Dints: Differentiable neural network topology search for 3d medical image segmentation*.
- Huang, H., Lin, L., Tong, R., Hu, H., Zhang, Q., Iwamoto, Y., ... Wu, J. (2020). Unet 3+: A full-scale connected unet for medical image segmentation. In *Icassp 2020 - 2020 IEEE international conference on acoustics, speech and signal processing (icassp)* (p. 1055-1059). doi: 10.1109/ICASSP40776.2020.9053405
- Huang, T.-Y., & Chen, Y.-Y. (2001). Parental population sizing in evolutionary strategies. In *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. no.01th8546)* (Vol. 2, p. 1351-1358 vol. 2). doi: 10.1109/CEC.2001.934348
- Isensee, F., Petersen, J., Klein, A., Zimmerer, D., Jaeger, P. F., Kohl, S., ... Maier-Hein, K. H. (2018). *nnu-net: Self-adapting framework for u-net-based medical image segmentation*.

- Joshi, D., Mishra, V., Srivastav, H., & Goel, D. (2021, oct). Progressive transfer learning approach for identifying the leaf type by optimizing network parameters. *Neural Process. Lett.*, 53(5), 3653–3676. Retrieved from <https://doi.org/10.1007/s11063-021-10521-x> doi: 10.1007/s11063-021-10521-x
- Joshi, D., & Singh, T. P. (2020, aug). A survey of fracture detection techniques in bone x-ray images. *Artificial Intelligence Review*, 53(6), 4475–4517. Retrieved from <https://doi.org/10.1007/s10462-019-09799-0> doi: 10.1007/s10462-019-09799-0
- Kamran, S. A., Hossain, K. F., Tavakkoli, A., Zuckerbrod, S. L., Sanders, K. M., & Baker, S. A. (2021). Rv-gan: Segmenting retinal vascular structure in fundus photographs using a novel multi-scale generative adversarial network. In *Lecture notes in computer science* (p. 34–44). Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-030-87237-3_4 doi: 10.1007/978-3-030-87237-3_4
- Kim, S., Kim, I., Lim, S., Baek, W., Kim, C., Cho, H., ... Kim, T. (2019). Scalable neural architecture search for 3d medical image segmentation. In *Medical image computing and computer assisted intervention – miccai 2019* (p. 220–228). Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-030-32248-9_25 doi: 10.1007/978-3-030-32248-9_25
- Kingma, D. P., & Ba, J. (2017). *Adam: A method for stochastic optimization*.
- Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983, 06). Optimization by simulated annealing. *Science (New York, N.Y.)*, 220, 671–80. doi: 10.1126/science.220.4598.671
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 25). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
- LeCun, Y., Bengio, Y., & Hinton, G. (2015, 05). Deep learning. *Nature*, 521, 436–44. doi: 10.1038/nature14539
- Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324. doi: 10.1109/5.726791
- LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database. <http://yann.lecun.com/exdb/mnist/>. Retrieved 2016-01-14 14:24:11, from <http://yann.lecun.com/exdb/mnist/>
- Lehman, J., & Stanley, K. (2011, 11). Novelty search and the problem with objectives. In (p. 37–56). doi: 10.1007/978-1-4614-1770-5_3
- Liu, W., Yang, H., Tian, T., Cao, Z., Pan, X., Xu, W., ... Gao, F. (2022, 07). Full-resolution network and dual-threshold iteration for retinal vessel and coronary angiograph segmentation. *IEEE journal of biomedical and health informatics*, PP. doi: 10.1109/JBHI.2022.3188710
- Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... Guo, B. (2021). Swin transformer:

- Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (p. 9992-10002). doi: 10.1109/ICCV48922.2021.00986
- Lou, A., Guan, S., & Loew, M. (2021, 02). Dc-unet: rethinking the u-net architecture with dual channel efficient cnn for medical image segmentation. In (p. 98). doi: 10.1117/12.2582338
- Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., ... Hodjat, B. (2017). *Evolving deep neural networks*.
- Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., ... Hodjat, B. (2019a). Evolving deep neural networks. In *Artificial intelligence in the age of neural networks and brain computing* (p. 293-312). Elsevier.
- Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., ... Hodjat, B. (2019b). Evolving deep neural networks. In *Artificial intelligence in the age of neural networks and brain computing* (p. 293-312). Elsevier.
- Miller, B. L., & Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Syst.*, 9. Retrieved from <https://api.semanticscholar.org/CorpusID:6491320>
- Miller, J. (2003). *Cartesian genetic programming* (Vol. 43). doi: 10.1007/978-3-642-17310-3
- Murphy, A. (2023). *radiopaedia.org*. Retrieved 13 Feb 2024, from <https://radiopaedia.org/articles/windowing-ct?lang=us>
- Oktay, O., Schlemper, J., Folgoc, L. L., Lee, M., Heinrich, M., Misawa, K., ... Rueckert, D. (2018). *Attention u-net: Learning where to look for the pancreas*.
- Pinto, N., Doukhan, D., Dicarlo, J., & Cox, D. (2009, 11). A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS computational biology*, 5, e1000579. doi: 10.1371/journal.pcbi.1000579
- Rahman, M. M., & Marculescu, R. (2023a). *G-cascade: Efficient cascaded graph convolutional decoding for 2d medical image segmentation*.
- Rahman, M. M., & Marculescu, R. (2023b, January). Medical image segmentation via cascaded attention decoding. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (p. 6222-6231).
- Rahman, M. M., & Marculescu, R. (2023c). *Multi-scale hierarchical vision transformer with cascaded attention decoding for medical image segmentation*.
- Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2019, Jul.). Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01), 4780-4789. Retrieved from <https://ojs.aaai.org/index.php/AAAI/article/view/4405> doi: 10.1609/aaai.v33i01.33014780
- Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., ... Kurakin, A. (2017). *Large-scale evolution of image classifiers*.
- Ren, S., He, K., Girshick, R., & Sun, J. (2016). *Faster r-cnn: Towards real-time object detection with region proposal networks*.
- Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-net: Convolutional networks for*

- biomedical image segmentation.
- Rosenblatt, F. (1957, January). *The perceptron - a perceiving and recognizing automaton* (Tech. Rep. No. 85-460-1). Ithaca, New York: Cornell Aeronautical Laboratory.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986a). Learning representations by back-propagating errors. *Nature*, 323, 533-536. Retrieved from <https://api.semanticscholar.org/CorpusID:205001834>
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986b). Learning representations by back-propagating errors. *Nature*, 323, 533-536. Retrieved from <https://api.semanticscholar.org/CorpusID:205001834>
- Sanfeliu, A., & Fu, K.-S. (1983). A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-13*, 353-362. Retrieved from <https://api.semanticscholar.org/CorpusID:1087693>
- Silva, F., Correia, L., & Christensen, A. (2017, 07). Evolutionary online behaviour learning and adaptation in real robots. *Royal Society Open Science*, 4. doi: 10.1098/rsos.160938
- Silva, F., Urbano, P., Correia, L., & Christensen, A. (2014, 12). odneat: An algorithm for decentralised online evolution of robotic controllers. *Evolutionary computation*, 23. doi: 10.1162/EVCO_a.00141
- Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., ... Hassabis, D. (2016, 01). Mastering the game of go with deep neural networks and tree search. *Nature*, 529, 484-489. doi: 10.1038/nature16961
- Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556. Retrieved from <https://api.semanticscholar.org/CorpusID:14124313>
- Simpson, A. L., Antonelli, M., Bakas, S., Bilello, M., Farahani, K., van Ginneken, B., ... Cardoso, M. J. (2019). A large annotated medical image dataset for the development and evaluation of segmentation algorithms. *CoRR*, abs/1902.09063. Retrieved from <http://arxiv.org/abs/1902.09063>
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014, 06). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15, 1929-1958.
- Staal, J., Abramoff, M., Niemeijer, M., Viergever, M., & van Ginneken, B. (2004). Ridge-based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 23(4), 501-509. doi: 10.1109/TMI.2004.825627
- Stanley, K., Clune, J., Lehman, J., & Miikkulainen, R. (2019, 01). Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1. doi: 10.1038/s42256-018-0006-z
- Stanley, K., & Miikkulainen, R. (2002, 02). Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10, 99-127. doi: 10.1162/106365602320169811
- Stanley, K. O., D'Ambrosio, D. B., & Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2), 185-212. doi: 10.1162/

artl.2009.15.2.15202

- Suganuma, M., Shirakawa, S., & Nagao, T. (2017). *A genetic programming approach to designing convolutional neural network architectures*.
- Sun, Y., Xue, B., Zhang, M., & Yen, G. (2018, 08). Automatically designing cnn architectures using genetic algorithm for image classification..
- Sun, Y., Yen, G. G., & Yi, Z. (2019). Evolving unsupervised deep neural networks for learning meaningful representations. *IEEE Transactions on Evolutionary Computation*, 23(1), 89-103. doi: 10.1109/TEVC.2018.2808689
- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). The MIT Press. Retrieved from <https://mitpress.mit.edu/books/reinforcement-learning-second-edition>
- Tang, Y., Yang, D., Li, W., Roth, H. R., Landman, B., Xu, D., ... Hatamizadeh, A. (2022). Self-supervised pre-training of swin transformers for 3d medical image analysis. In *2022 IEEE/CVF conference on computer vision and pattern recognition (cvpr)* (p. 20698-20708). doi: 10.1109/CVPR52688.2022.02007
- Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2021). *Training data-efficient image transformers and distillation through attention*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2023). *Attention is all you need*.
- Verbancsics, P., & Harguess, J. (2015). Image classification using generative neuro evolution for deep learning. *2015 IEEE Winter Conference on Applications of Computer Vision*, 488-493. Retrieved from <https://api.semanticscholar.org/CorpusID:12346405>
- Wang, B., Sun, Y., Xue, B., & Zhang, M. (2018). *Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification*.
- Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., ... Shao, L. (2022, March). Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3), 415–424. Retrieved from <http://dx.doi.org/10.1007/s41095-022-0274-8> doi: 10.1007/s41095-022-0274-8
- Xie, L., & Yuille, A. (2017). *Genetic cnn*.
- Yu, Q., Yang, D., Roth, H., Bai, Y., Zhang, Y., Yuille, A. L., & Xu, D. (2020). C2fnas: Coarse-to-fine neural architecture search for 3d medical image segmentation. In *2020 IEEE/CVF conference on computer vision and pattern recognition (cvpr)* (p. 4125-4134). doi: 10.1109/CVPR42600.2020.00418
- Zhou, Z., Rahman Siddiquee, M., Tajbakhsh, N., & Liang, J. (2018). Unet++: A nested unet architecture for medical image segmentation. In L. Maier-Hein et al. (Eds.), *Deep learning in medical image analysis and multimodal learning for clinical decision support - 4th international workshop, dlmia 2018 and 8th international workshop, ml-cds 2018 held in conjunction with miccai 2018* (pp. 3–11). Springer Verlag. doi: 10.1007/978-3-030-00889-5_1
- Zhou, Z., Sodha, V., Pang, J., Gotway, M., & Liang, J. (2021, 01). Models genesis. *Medical Image Analysis*, 67, 101840. doi: 10.1016/j.media.2020.101840
- Zoph, B., & Le, Q. (2017). Neural architecture search with reinforcement learning. In

International conference on learning representations. Retrieved from <https://openreview.net/forum?id=r1Ue8Hcxg>

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). *Learning transferable architectures for scalable image recognition.*