Fabio Henrique Cardoso

Quantum-inspired Neural Architecture Search applied to Semantic Segmentation using Symmetric Networks

Dissertação de Mestrado

Thesis presented to the Programa de Pós–graduação em Engenharia Elétrica, do Departamento de Engenharia Elétrica da PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Elétrica.

Advisor : Profa. Marley Maria Bernardes Rebuzzi Vellasco
Co-advisor: Profa. Karla Tereza Figueiredo Leite

Rio de Janeiro
April 2025

Fabio Henrique Cardoso

Quantum-inspired Neural Architecture Search applied to Semantic Segmentation using Symmetric Networks

Thesis presented to the Programa de Pós–graduação em Engenharia Elétrica da PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Elétrica. Approved by the Examination Committee:

Rio de Janeiro, April the 14th, 2025

Fabio Henrique Cardoso

Graduated in Computer Science at the State University of Rio de Janeiro (UERJ) in 2022.

Bibliographic data

Cardoso, Fabio Henrique

Quantum-inspired Neural Architecture Search applied to Semantic Segmentation using Symmetric Networks / Fabio Henrique Cardoso; advisor: Marley Maria Bernardes Rebuzzi Vellasco; co-advisor: Karla Tereza Figueiredo Leite. – 2025.

94 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2025.

Inclui bibliografia

1. Engenharia Elétrica – Teses. 2. Engenharia de Materiais – Teses. 3. Busca por Arquitetura Neural. 4. Algoritmos Evolucionários. 5. Segmentação Semântica. 6. Computação com Inspiração Quântica. I. Rebuzzi Vellasco, Marley Maria Bernardes. II. Leite, Karla Tereza Figueiredo. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. IV. Título.

CDD: 621.3

Acknowledgments

I will never forget my family kindness during my pursue the master's degree, in special my mother Carla and my grandmother Gilda that supported my daily during these period.

I also would like to express my deepest gratitude to my two advisors Marley and Karla for her guidance, support, and availability. Karla has encouraged me to start my master degree and I'm really glad for that, and Marley has been very supportive and provided valuable feedback and encouragement.

I would also like to thank M.Sc. Diego Paez and M.Sc. Laura Parra for all their companionship and conversations, which were very valuable during my master's degree. In addition, they were friendships that I gained during this time and that I intend to maintain.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Cardoso, Fabio Henrique; Rebuzzi Vellasco, Marley Maria Bernardes (Advisor); Leite, Karla Tereza Figueiredo (Co-Advisor). **Quantum-inspired Neural Architecture Search applied to Semantic Segmentation using Symmetric Networks**. Rio de Janeiro, 2025. 94p. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Deep learning has revolutionized various domains, showing great performance for several perceptual tasks, in the fields of computer vision, speech recognition, and natural language processing. However, designing optimal deep neural network (DNN) architectures often rely on expert knowledge and time-consuming trial and error approaches. Neural Architecture Search (NAS) has emerged as a promising solution, automating the design process to discover architectures that enhance performance and efficiency. This work introduces SegQSNAS, an extension of SegQNAS, designed for Q-NAS (Quantum-inspired Neural Architecture Search) in semantic segmentation, particularly within the medical imaging domain. SegQSNAS is designed to search for symmetrical U-Net-like architectures, thereby reducing the search space and eliminating the need for feasibility checks that were needed in SegQNAS. Furthermore, it enhances the search process by incorporating a two-point crossover operation in order to improve the exploitation during evolutionary process, alongside the addition of self-attention, MobileNet, and EfficientNet functions to the search space that enables the discovery of efficient, customized networks. To address inaccuracies in multi-class scenarios, the SegQNAS implementation of Dice-Sørensen Coefficient (DSC) and loss functions were corrected because it provides an overestimated DSC score in those scenarios. Experiments on medical segmentation datasets from the Medical Segmentation Decathlon challenge demonstrate that on the prostate dataset experiment, SegQSNAS performed well with the best results in DSC score, parameter count and GPU days, archiving 0.7924 in average DSC score with half million parameters. It also archive of good results on the liver dataset experiment due to the best trade-off considering that it is a limited computing resource scenario. Although, in some experiments it show some limitation of the search strategy or computational resource available. In addition, in any experiment the self-attention block was not selected in the best architecture found, and this is indicative that the predefined maximum number of nodes or the complexity of the tested problems might not have been high enough for this operation to be selected during evolution process.

Keywords

Neural Architecture Search; Evolutionary Algorithms; Semantic Segmentation; Quantum-inspired Computin.

Resumo

Cardoso, Fabio Henrique; Rebuzzi Vellasco, Marley Maria Bernardes; Leite, Karla Tereza Figueiredo. **Busca por arquitetura neural com inspiração quântica aplicada a segmentação semântica utilizando redes neurais simétricas**. Rio de Janeiro, 2025. 94p. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

A aprendizagem profunda revolucionou vários domínios, demonstrando excelente desempenho em várias tarefas perceptivas nos campos de visão computacional, reconhecimento de fala e processamento de linguagem natural. No entanto, o projeto de arquiteturas ideais de redes neurais profundas geralmente depende do conhecimento de especialistas e de abordagens demoradas de tentativa e erro. O campo da busca por arquiteturas neurais (NAS - *Neural Architecture Search*) surgiu como uma solução promissora, automatizando o processo de design para descobrir arquiteturas que melhoram o desempenho e a eficiência. Este trabalho apresenta o SegQSNAS, uma extensão do SegQNAS, projetado para Q-NAS (*Quantum-inspired Neural Architecture Search*) em segmentação semântica, especialmente no domínio de imagens médicas. O SegQSNAS foi projetado para pesquisar arquiteturas simétricas do tipo U-Net, reduzindo assim o espaço de busca e eliminando a necessidade de verificações de viabilidade de solução que eram necessárias no SegQNAS. Além disso, ele aprimora o processo de busca incorporando uma operação de cruzamento de dois pontos para melhorar a exploração durante o processo evolutivo, juntamente com a adição de funções de self-attention, MobileNet e EfficientNet ao espaço de busca que permite a descoberta de redes eficientes e personalizadas. Para tratar de imprecisões em cenários com vários rótulos, a implementação do SegQNAS do Coeficiente de Dice-Sørensen (DSC) e as funções de perda foram corrigidas porque fornecem uma avaliação de DSC superestimada nesses cenários. Os experimentos com conjuntos de dados de segmentação médica do desafio *Medical Segmentation Decathlon* demonstram que, no experimento do conjunto de dados de próstata, o SegQSNAS teve um bom desempenho com os melhores resultados em DSC, contagem de parâmetros e dias de GPU, alcançando 0.7924 no DSC médio com meio milhão de parâmetros. SegQSNAS também apresentou bons resultados no experimento com o conjunto de dados do fígado devido ao melhor *trade-off*, considerando que se trata de um cenário de recursos de computação limitados. No entanto, em alguns experimentos, ele mostra alguma limitação da estratégia de pesquisa ou do recurso computacional disponível. Além disso, em todos os experimentos, a função de self-attention não foi selecionada na melhor arquitetura encontrada, o que indica que o número máximo predefinido de nós ou a complexidade dos problemas testados podem não ter sido suficientemente altos para que essa operação fosse selecionada durante o processo de evolução.

Palavras-chave

Busca por Arquitetura Neural; Algoritmos Evolucionários; Segmentação Semântica; Computação com Inspiração Quântica.

Table of contents

List of figures

List of tables

List of Abreviations

ANN – Artificial Neural Network
ASPP – Atrous Spatial Pyramid Pooling
CNN – Convolutional Neural Network
CRF – Conditional Random Field
DAG – Direct Acyclical Graph
DNN – Deep Neural Network
DPC – Dense Prediction Cell
EA – Evolutionary Algorithm
FCN – Fully Convolutional Network
MLP – Multilayer Perceptron
NAS – Neural Architecture Search
QIEA – Quantum-Inspired Evolutionary Algorithm
Q-NAS – Quantum-Inspired Neural Architecture Search
RL – Reinforcement Learning
SegQNAS – Quantum-inspired Neural Architecture Search applied to Semantic Segmentation
SegQSNAS – Quantum-inspired Neural Architecture Search applied to Semantic Segmentation using Symmetric Networks

1
Introduction

Deep learning techniques have achieved excellent results across various domains, ranging from image classification to natural language processing. Its architecture, characterized by multiple layers of processing units, enables the extraction of hierarchical representations from complex data [1, 2, 3]. The ability of deep learning models to learn from vast amounts of unstructured data has positioned them as superior alternatives to traditional machine learning techniques, such as Support Vector Machines (SVM) and K-Nearest Neighbors (K-NN) [4, 5].

Before the use of Deep Learning models, the feature extraction process was performed by traditional pattern recognition methods, which were time-consuming and lacked efficacy due to the inherent human bias [6]. On the other hand, deep learning techniques can automatically learn to extract features that are important for the task, and that may be non-obvious even to an expert in the field. Additionally, deep learning techniques can extract multiple features on various scales in a hierarchical form and define the importance of each feature for the task.

In the healthcare sector, Deep Learning models[7] are rapidly transforming the landscape of medical diagnosis, empowering medical experts to provide more accurate and faster diagnoses. For example, deep learning techniques have been successfully applied in tumor detection and segmentation, achieving state-of-the-art results in various cancer imaging tasks [8]. The deep learning capabilities to process and analyze large quantities of data are beneficial in the healthcare sector, where it can detect patterns previously elusive to human experts [5, 9].

Despite deep learning's ability to automatically extract task-specific features, tuning several hyperparameters is still required for optimal neural network performance. Each layer of the neural network can perform a different operation, such as convolution, fully connected layers, pooling, transposed convolution, etc. Each layer has its own set of hyperparameters to be defined. For instance, configuring a convolutional layer involves specifying parameters such as the number of kernels (filters), kernel size, stride, and padding.

Historically, defining DNN architectures has relied on expert knowledge and a trial-and-error approach. However, manually designing novel DNN architectures, aside from being time-consuming, is prone to human bias and may leave non-obvious architectures unexplored [10, 11, 12].

Given the challenges of manually configuring DNNs [13], an obvious next step is to automate their architecture design. This research field is known as Neural Architecture Search (NAS) [14]. NAS algorithms aim to find an optimal neural network architecture that improves performance and computational efficiency [15].

Two prominent NAS methodologies are Evolutionary Algorithms (EAs) and Reinforcement Learning (RL). EAs are generally more sample efficient, allowing for a broader exploration of the architecture space with fewer evaluations [16] and can be advantageous in scenarios where computational resources are limited. However, RL methods, while potentially more powerful in discovering complex architectures, often suffer from longer training times and higher resource consumption [17].

Quantum-Inspired Evolutionary Algorithm (QIEA) incorporates concepts from quantum mechanics, such as superposition and entanglement, into the evolutionary process. QIEAs utilize quantum bits (qubits) to represent potential solutions, allowing for a more refined exploration of the search space.

The use of QIEAs in NAS presents advantages over traditional EA and RL methods. QIEAs can effectively balance exploration and exploitation, leading to faster convergence towards optimal architectures [18]. In addition, QIEAs can represent multiple states simultaneously, which allows a more comprehensive search, identifying architectures that might be underestimated by conventional methods [19]. Q-NAS (Quantum-inspired Neural Architecture Search [20] is one of the proposed methods, which introduces a Quantum-inspired Evolutionary Algorithm (QIEA) to perform the search for a deep neural architecture on the image classification task.

The SegQNAS (Quantum-inspired Neural Architecture Search applied to Semantic Segmentation) is an extension of Q-NAS that performs NAS on semantic segmentation DNNs, generating U-Net-like architectures. The algorithm was evaluated on two datasets from the Medical Segmentation Decathlon challenge [21].

1.1 Objectives

In this work, we introduce SegQSNAS (Quantum-inspired Symmetrical Neural Architecture Search applied to Semantic Segmentation), an extension of SegQNAS. Our research pursues two main objectives.

First, we aim to develop a NAS algorithm that builds upon SegQNAS and creates a symmetrical neural network to reduce search space. This strategy eliminates the need for individual feasibility checks during the search process.

Second, the goal is to enhance the SegQNAS algorithm by incorporating new types of layers and adding crossover to the Quantum-inspired Evolutionary Algorithm (QIEA) applied to SegQNAS.

Finally, we conduct a comparative analysis to evaluate SegQSNAS's effectiveness. This comparison evaluates its performance against the original SegQNAS algorithm and other state-of-the-art NAS approaches in the literature.

1.2 Contributions

In this section, we list the main contributions of this work:

– **Search space reduction by using Symmetrical Neural Network**: SegQSNAS was designed to search the space of symmetrical U-net-like networks. This symmetry reduce the search space in half compared to SegQNAS because the other half of the network mirrors the first half. Furthermore, this symmetrical design inherently eliminates the need for the feasibility check required by SegQNAS, as every downsampling operation in one half of the network is guaranteed to have a corresponding upsampling operation of the same proportion in the other half.

– **Addition of the crossover to the evolutionary process**: A two-point crossover operation was added to the evolutionary process presented by the SegQNAS algorithm, providing an improved and less noisy evolution compared to the original one.

– **Addition of self-attention block**: A self-attention block was added to the available blocks in the search space. This kind of layer allows for a better composition of filters found during the convolutional process.

– **Addition of lightweight convolutional block**: Lightweight blocks, such as MobileNet and EfficientNet, were also added, providing the possibility to achieve an individual solution with an even smaller number of parameters when compared to the blocks implemented on SegQNAS.

– **Metrics and Loss functions correction**: The Dice-Sørensen coefficient (DSC) function used in SegQNAS calculate the correct values for binary problems. However, for multi-class problems, the calculated value was overestimated because the numerator in the DSC formula does not fully account for the true overlap between multiple classes, while the denominator is inflated due to the sum over all labels. In order to solve this, the DSC function was corrected for both binary and multi-class problems.

1.3 Work outline

This work comprises five additional chapters, which we describe below.

Chapter 2 presents the semantic segmentation task and describes how current deep neural network architectures are designed to tackle this computer vision problem.

Next, Chapter 3 provides the theoretical background for understanding NAS and SegQNAS, their components (search space, search strategy, performance estimation strategy), and their challenges.

Chapter 4 introduces SegQSNAS and discusses its changes, including the search space used and enhancements to the algorithm.

Chapter 5 describes and discusses the experiments proposed to validate SegQSNAS. The results are presented and compared to baselines and other models in the literature.

Finally, Chapter 6 concludes our work and discusses the next steps of our research.

2
Semantic Segmentation

In this chapter, we present the concept of semantic segmentation and how convolutional neural networks are used to solve this task. First, we provide a brief review of how convolutional neural networks perform automatic feature extraction. Then, we focus on the encoder-decoder-based methods.

2.1
Semantic Segmentation

Image segmentation is a key research area in computer vision that involves assigning a meaningful label to each pixel inside an image, enabling a more detailed understanding of the scene [22]. This task can be divided into three main categories.

- *Semantic segmentation*: This focuses on understanding the overall scene by assigning a class label to every pixel, thereby segmenting the image into regions that correspond to different object categories, providing a comprehensive understanding of the scene;
- *Instance segmentation*: In contrast, instance segmentation involves not only classifying each pixel but also distinguishing between different instances of the same category;
- *Panoptic segmentation*: Panoptic segmentation deals with performing both semantic and instance segmentation. It aims to assign a semantic label to each pixel while also providing unique identities for each instance of objects in the scene.

Semantic segmentation is a critical area of research within computer vision and is essential for various applications, including autonomous driving, medical imaging, and remote sensing, where precise object localization and classification are crucial [24, 25, 26, 27, 28, 29, 30, 31]. Regarding autonomous driving applications, the labels could be road, pedestrian, vehicle, etc. On the other hand, when it comes to medical image analysis, the semantic labels could be organs, vessels, and tissues, to name a few possibilities.

The evolution of semantic segmentation has been significantly influenced by advances in deep learning, especially by the use of convolutional neural networks. These networks have allowed the extraction of high-level features from images, which enhances the accuracy and efficiency of segmentation tasks [32, 33, 34].

Traditional semantic segmentation methods often relied on hand-crafted features and shallow learning techniques, which limited their performance. Several semantic segmentation methods have been developed in the literature, such as thresholding [35], region-growing [36], clustering of k-means [37], watershed methods [38], active contours [39], graph cuts [40], conditional random fields (CRFs) [41], Markov random fields [42] and sparsity-based methods [43, 44]

In contrast, deep learning approaches have demonstrated superior handling of complex image data, leading to state-of-the-art results on various benchmarks [46, 47]. Deep learning methods were able to address the issue of automated feature learning, achieving remarkable performance in several computer vision tasks, such as classification, object detection, and semantic segmentation [45].

The following sections will first introduce the fundamentals of convolutional neural networks (CNNs) and their application in semantic segmentation. We will then explore the encoder-decoder architecture, a widely used framework for segmentation tasks.

2.2 Convolutional Neural Networks Applied to Segmentation Task

Convolutional Neural Networks (CNNs) are deep neural networks employed in image processing, video analysis, and other tasks that require feature extraction from structured data. The fundamental operation of CNNs revolves around the convolutional layer, which utilizes a set of learnable filters (or kernels) to extract local features from input data. Typically, CNNs are composed of stacked blocks of convolutional layers, activation layers, and pooling layers (Figure 2.1).

The convolutional layers apply filters to the input data, generating feature maps that highlight key patterns [48, 49]. The convolutional process is inherently hierarchical, where the initial layers capture low-level features such as edges and textures, while the deeper layers specialize in abstract representations, allowing the network to understand complex patterns and structures [50, 48, 51].

A convolutional layer receives an input tensor of shape $(H \times W \times C)$, where H is the height, W is the width, and C is the number of channels. The number of channels typically corresponds to the color components in an

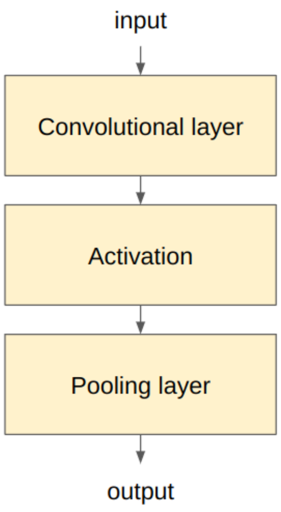


Figure 2.1: A typical CNN block comprises a convolutional layer, activation layer, and pooling layer.

image (e.g., 3 for RGB). Then, it convolves k kernels of size $(h \times w \times C)$ along the x and y axes of the input tensor, producing an output tensor with shape $(\frac{H-h+2p}{s} + 1 \times \frac{W-w+2p}{s} + 1 \times k)$ where p is the padding and s is the stride.

Padding refers to the process of adding extra layers of pixels around the input data. This technique is often used to control the spatial dimensions of the output feature map, allowing for the preservation of the input size or allowing the kernel to be applied to the edges of the input. The stride defines the step size at which the kernel moves across the input tensor. A larger stride reduces the spatial dimensions of the output feature map more significantly. The kernel is a tensor of trainable weights that are adjusted through the training process to extract task-specific features from the input tensor.

Figure 2.2 represents the convolution operation applied to a $4 \times 4 \times 1$ input tensor with a $2 \times 2 \times 1$ kernel with $s = 1$ and $p = 0$ producing an output tensor of $3 \times 3 \times 1$.

The activation layer performs an element-wise non-linear operation on an input tensor, enabling the CNN to learn non-linear features. There are several possibilities for activation functions, ReLU [52] and Softmax [53] are the most common.

The ReLu function for every input x, if x is positive, the output is x; otherwise, the output is zero (Equation 2-2). In the Softmax function z represents the input logits, z_i is the score for class i, and the denominator sums the exponentials of all logits to ensure that the output sum is one (Equation 2-1).

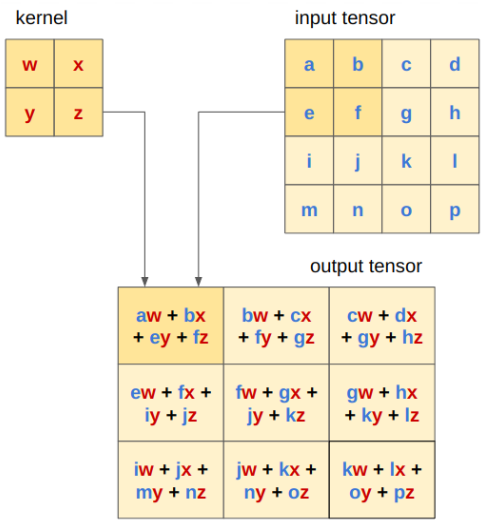


Figure 2.2: Example of a convolution with $s = 1$ and $p = 0$.

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{2-1}$$

$$\text{ReLU}(x) = \max(0, x) \tag{2-2}$$

Finally, the pooling layers reduce the dimensionality of the feature maps to retain significant features while discarding less important information [54, 55]. Reducing the feature maps resolution can reduce the resource requirement, which improves efficiency, and, besides that, it also allows for small kernels to have a larger receptive field, which means that a larger region of the input image is taken into account in a kernel patch during convolution.

The polling layers that are the most commonly used are the average polling layer and the maximum polling layer [56]. Pooling layers replaces the values by an aggregation method and slides through the input with stride s as shown in Figure 2.3 with average as the aggregation method.

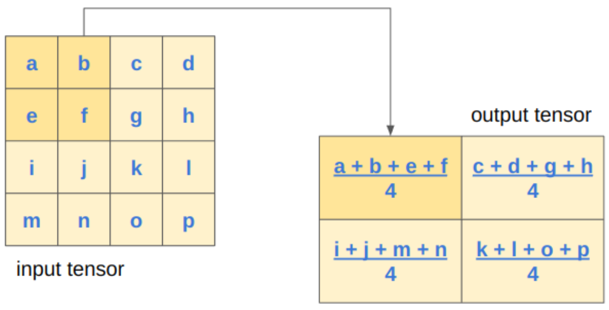


Figure 2.3: Example of an 2x2 average pooling operation with $s = 2$.

The Convolutional Neural Networks are designed to allow them to understand the spatial context and relationships between different parts of an image. This capability gives them significant leverage in tasks like classification and segmentation. The primary distinction between segmentation and classification tasks is the output format. In the segmentation task, it produces a pixel-wise classification, while classification assigns a single label to the entire image.

In the next section, we will present encoder-decoder-based models, a class of CNN that has become increasingly popular in image segmentation, especially in medical imaging.

2.3 Encoder-Decoder-based models

Encoder-decoder architectures (Figure 2.4) have become fundamental in the field of image segmentation, particularly in medical imaging. The main characteristics of these architectures are that they are separated into three main parts: the encoder, which captures the context and features of the input image; the latent space, which is the most compressed information that represents all the image; and the decoder, which reconstructs the image from latent space to produce a segmentation map.

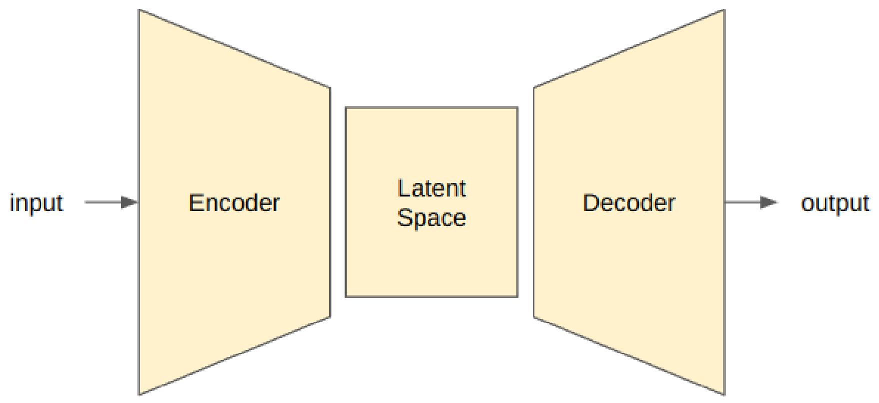


Figure 2.4: Default autoencoder representation.

The U-Net model (Figure 2.5) is one of the encoder-decoder architectures widely used in the semantic segmentation task. Originally developed for biomedical image segmentation, U-Net employs a symmetric structure with skip connections. A skip connection is added after every encoder convolutional block (before the pooling operation) to the decoder, and its purpose is to facilitate the integration of high-resolution features from the encoder, improving the preservation of fine details during the segmentation process [57, 58].

Variants of U-Net [59], such as U-Net++ [57] and Double U-Net [60], have been proposed to further improve performance by incorporating nested skip pathways and stacking multiple U-Net architectures. These modifications were proposed to refine the segmentation output, especially in complex scenarios where precise labels are necessary [61, 62].

Other contributions of the U-Net paper [59] include the training strategy that is highly dependent on data augmentation and the Dice loss function (Equation 2-4) based on the Dice coefficient (Equation 2-3).

The dice function receive every pixel real label $y_{true,i}$ and predicted label $y_{pred,i}$ and use the representation of intersection as $\sum_i y_{true,i} \cdot y_{pred,i}$ and the union as $\sum_i (y_{true,i} + y_{pred,i})$ to calculate the score, being 1 the representation of a perfect overlap of the real and predicted label and 0 the representation of no overlap between real and predicted labels.

$$\text{Dice Coefficient} = \frac{2 \cdot (\sum_i y_{true,i} \cdot y_{pred,i})}{(\sum_i (y_{true,i} + y_{pred,i})) + \text{smooth}} \tag{2-3}$$

$$\text{Dice Loss Function} = 1 - \text{Dice Coefficient} \tag{2-4}$$

Another encoder-decoder architecture is SegNet [63], which also follows the encoder-decoder paradigm. SegNet (Figure 2.6) proposes a similar approach to U-Net, but the key difference lies in how they transfer information from the encoder to the decoder. SegNet sends only pooling indices to the decoder, whereas U-Net transmits the entire feature map via its skip connections. Both approaches are used to create an accurate and detailed reconstructed image.

SegNet architecture has been adopted for high-resolution image segmentation tasks due to its ability to maintain spatial hierarchies while processing the image [61, 64]. Additionally, incorporating attention mechanisms in models like Attention U-Net has further improved these networks' ability to focus on relevant features, enhancing segmentation accuracy [65].

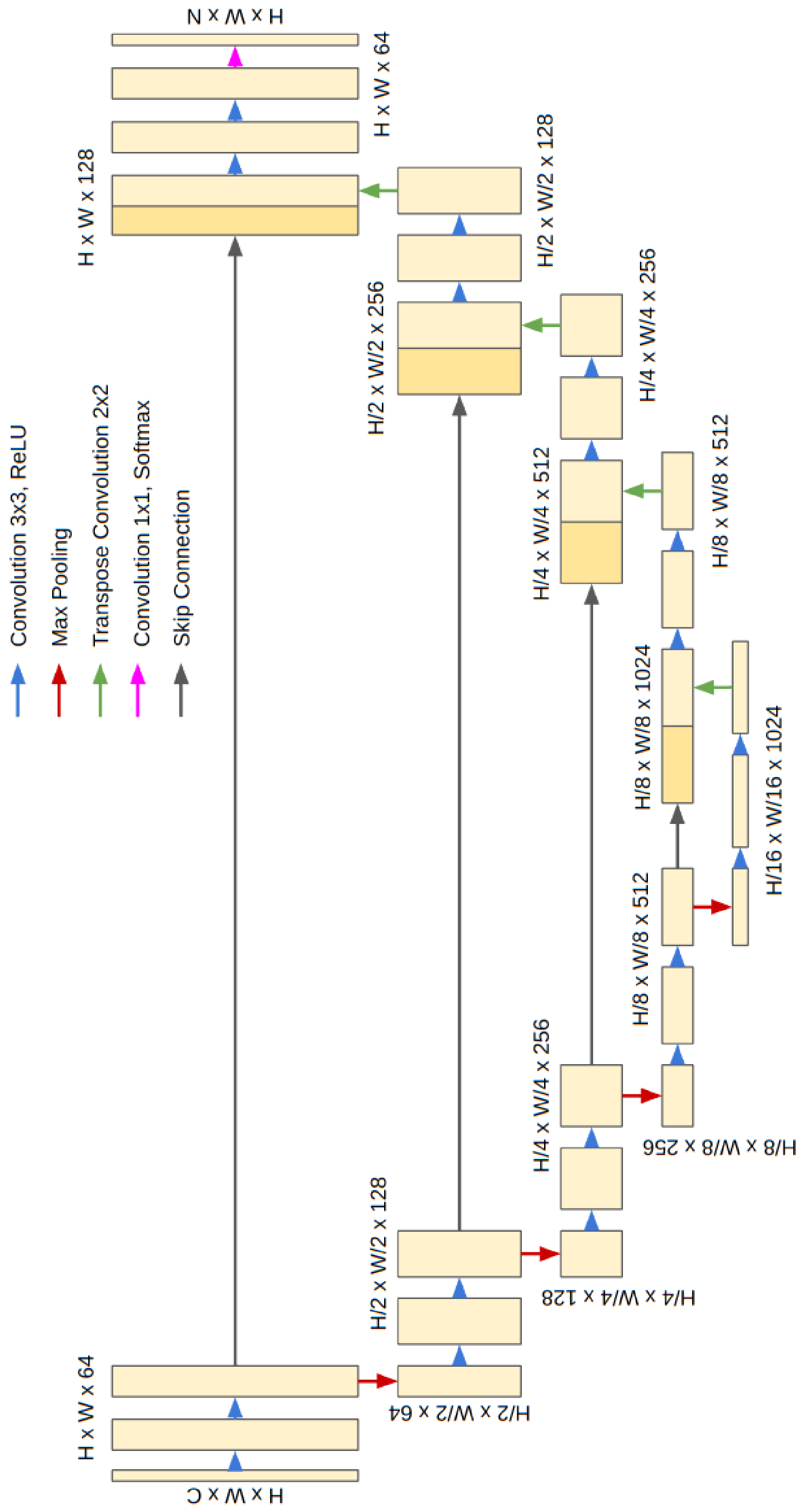


Figure 2.5: U-Net architecture.

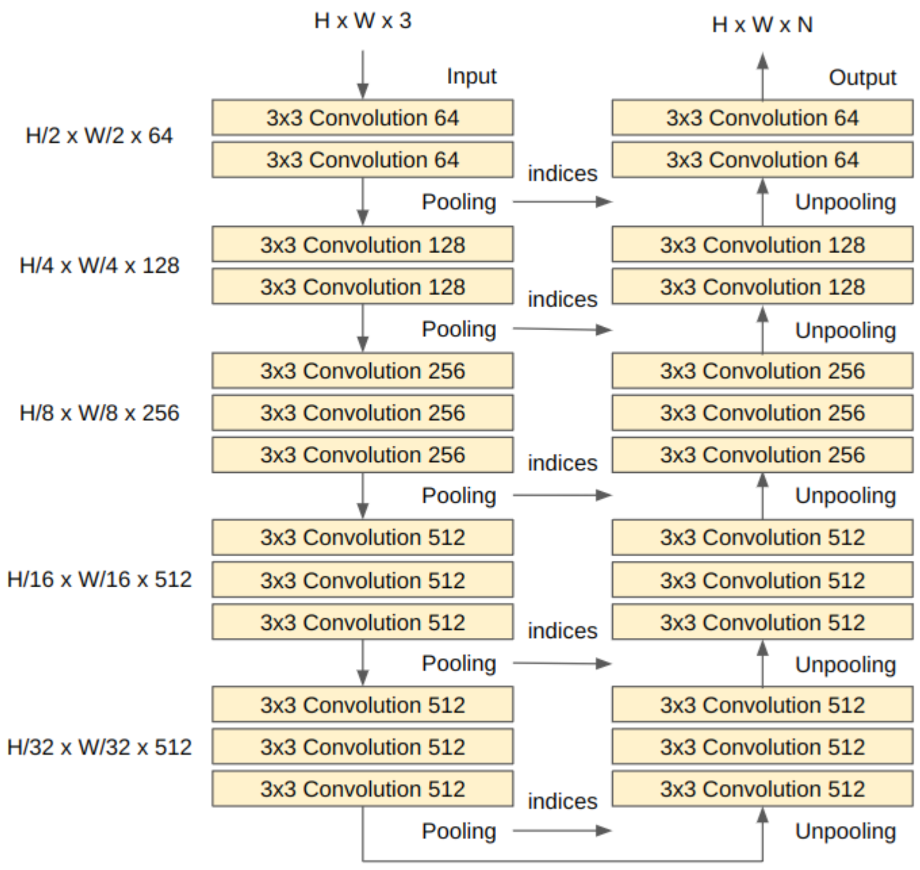


Figure 2.6: SegNet layer representation.

3
Neural Architecture Search

In this chapter, we present the Neural Architecture Search (NAS) field. First, we review some of the NAS methods. Then, we detail NAS using a Quantum Evolutionary Algorithm (Q-NAS), including its adaptation for the segmentation task (SegQNAS).

3.1
Neural Architecture Search

The NAS process can be divided into three main components: Search Space, Search Strategy, and Performance Estimation Strategy (Figure 3.1) [66]. The Search Strategy module samples candidate architectures a from the search space A and evaluates them according to a performance estimation strategy. The evaluation is used to determine which candidates are better than the others and to assist in the search strategy sampling of the search space, as the problem deals with the exploration-exploitation trade-off.

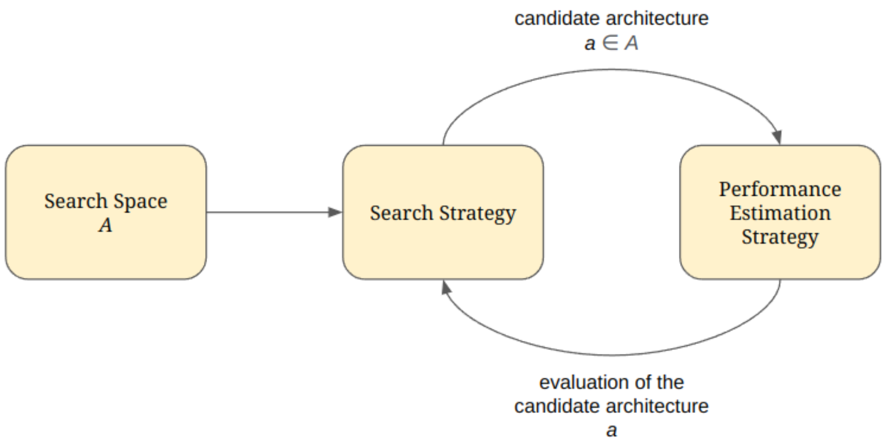

candidate architecture
$a \in A$
Search Space
A
Search Strategy
Performance Estimation Strategy
evaluation of the candidate architecture
a

Figure 3.1: NAS process

3.1.1
Search Space

The network is represented as a direct acyclic graph (DAG) [67] composed of k nodes $\in Z$ connected sequentially, where each node $z^{(k)}$ of the graph represents a tensor and its associated operation $o^{(k)} \in O$ that is applied to its set of parent nodes $z^{(k-1)}$ (Equation 3-1). All possible architectures represented by this DAG comprise the search space, combining all possible node connections and operations presented in the set O. The operations o such as convolutions, polling operations, and fully connected layers, belong to the set O.

$$z^{(k)} = o^{(k)}(z^{(k-1)}) \tag{3-1}$$

The choice of search space affects the complexity of the search problem, as the search space can be non-continuous and exponentially large [66]. Usually, two main groups of search spaces are used to represent CNNs:

– Global search space: Refers to the comprehensive set of all possible neural network architectures that can be generated based on a defined set of operations and connections. This space is often vast and complex, since it includes every conceivable arrangement of layers, nodes, and connections throughout the entire network;
– Cell search space: Refers to the layers' operations and hyperparameters, such as kernel size, number of kernels, stride, padding, and activation function for a convolutional layer.

These two types of search spaces are critical for an effective NAS. The global search space provides the overarching framework for exploring various architectures. In contrast, the cell search space allows for detailed optimization of the components that compose these architectures.

3.1.2 Search Strategy

The objective of the search strategy is to find an architecture a from the search space A that maximizes a certain performance metric in unseen data (validation set). Furthermore, the search strategy defines how the search space will be sampled to perform better in the validation set when trained in the training set (Figure 3.1).

The search space is potentially exponentially large, so the choice of search strategy must address the exploitation-exploration trade-off. Although quick convergence is desirable, it may lead to poor-performing DNNs. Two main strategies are used in NAS: Reinforcement Learning (RL) and Evolutionary Algorithms (EA).

RL methods are used to model sequential decision-making processes. The approach involves defining a policy that selects architectures based on their expected performance, which is evaluated through a reward mechanism. The policy defines how the agent should act based on the environmental state, and the environment state is obtained by observation (Figure 3.2).

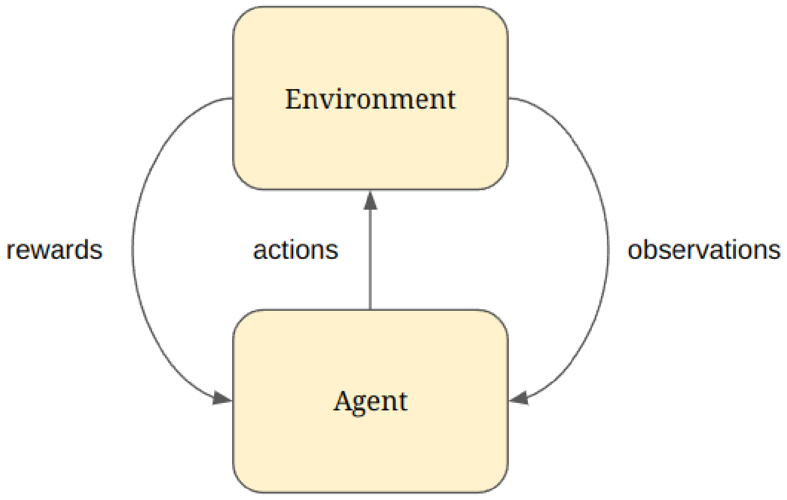


Figure 3.2: A general framework for RL methods.

EAs, on the other hand, are optimization algorithms inspired by the natural process of evolution. When applying EAs to NAS, the population is a set of candidate architectures sampled from the search space, and its fitness is the validation set's evaluation. In addition, the combination of mutation and crossover operations in EA facilitates the discovery of architectures that might not be found using traditional gradient-based methods [68].

The algorithm comprises some key steps: initialization, parent selection, recombination and mutation, and selection of the fittest individuals [69]. The population of architectures evolves over generations. Each architecture is evaluated based on its performance, and the best-performing architectures are selected to produce offspring for the next generation, as shown in Figure 3.3.

EA approach allows for a more diverse exploration of the search space through mechanisms such as mutation and crossover, which can lead to the discovery of architectures that might not be found through the RL approach [70, 71]. In contrast, RL methods rely on a single agent's experience, which can limit exploration and lead to suboptimal solutions [72, 73].

The computational cost associated with training multiple architectures in RL can be high due to often requiring extensive training cycles to evaluate the performance of each candidate architecture [72, 74]. In contrast, EAs can evaluate multiple architectures simultaneously, reducing the computational burden [75, 76].

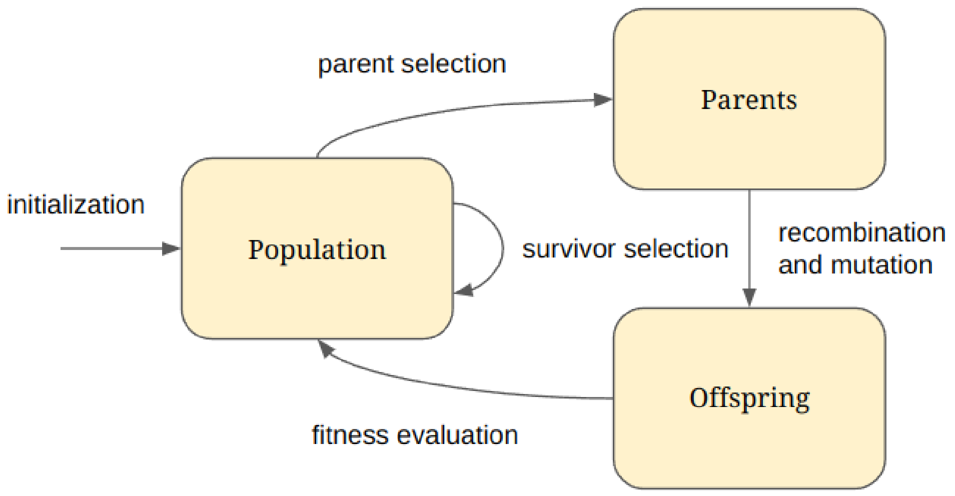


Figure 3.3: A general framework for EA methods.

Another advantage of EAs compared to RL is their robustness to hyperparameter tuning, since RL methods require tuning of various hyperparameters, such as learning rates and exploration strategies, which can be time-consuming and may not generalize well across different tasks [77, 78].

In summary, EA offers advantages over RL, including exploration capabilities, improved sample efficiency, and robustness in hyperparameter tuning.

3.1.3 Performance Estimation Strategy

As mentioned, NAS aims to find a neural architecture a that maximizes performance on unseen data. Performance estimation strategies in NAS can be categorized into full evaluations, multi-fidelity evaluations, and one-shot models.

Full evaluations involve training each candidate architecture from scratch and evaluating its performance, which, while accurate, is computationally expensive and time-consuming [74, 79]. To mitigate these costs, multi-fidelity evaluations have been proposed, where architectures are evaluated using fewer resources, such as limited training epochs or smaller datasets, to provide faster feedback on their potential performance [80, 81]. One-shot models are a new approach in NAS where a single model is trained to evaluate multiple architectures. This approach can reduce the training time since the weights are shared across different architectures [82]. However, one-shot models can lead to misleading performance estimates [74].

Although the above mentioned strategies focus mainly on reducing the time and resource consumption of the performance estimation process, noise in the estimation is also important. Estimating the performance of a DNN is a stochastic process due to several factors such as weights initialization, training algorithm, and dataset split. To reduce noise in the process, performing k-fold cross-validation, where the dataset is divided into k subsets, where each part is then used once as validation set while the remaining $k - 1$ parts form the training set, leads to a more reliable performance estimate than the straightforward single-initialization and single evaluation [84].

3.2 Q-NAS

Quantum-inspired evolutionary algorithms (QIEAs) represent solutions using the basic unit of quantum computing, q-bit, which allows a probabilistic approach to the search space, allowing for enhanced exploration capability [85, 86]. In the context of NAS, QIEAs offer advantages over conventional evolutionary algorithms due to efficient exploration of diverse architectures and facilitate better convergence, reducing the likelihood of premature convergence, which is a common issue in classical evolutionary algorithms [87, 88, 89].

This section presents Q-NAS, a QIEA-based neural architecture search proposed by [83]. The proposed work searches for the best DNN architecture to perform image classification. However, it was designed to be task agnostic, meaning it could be applied to any task.

3.2.1 Search Space

Quantum individual representation is a key concept for quantum-inspired evolutionary algorithms as these quantum individuals encode a superposition of all potential solutions, allowing for a more efficient exploration of the search space. Upon observation, the quantum individual collapses into a classical representation, which is then decoded into a feasible solution for the problem. In Q-NAS, a quantum individual consists of two components: the numerical part and the categorical part.

The numerical part of the quantum chromosome represents the probability distribution of all possible hyperparameter values. However, the categorical part represents the probability distribution over the possible functions assigned to each node in the architecture.

The numerical part is represented by Equation 3-2 where G represents the sets of hyperparameters. The function $p_{i1}(x)$ represents the probability density function (PDF) that governs the probability of observing a specific value (in the specified range) for the hyperparameter j when the quantum individual

i is observed. During initialization, the user provides lower l_{ij} and upper u_{ij} bounds for each hyperparameter in the numerical part of the chromosome, constraining the search space within feasible limits. Figure 3.4 shows the visual representation of the numerical chromosome.

$$q_i = [p_{i1}(x), ..., p_{iG}(x)] \tag{3-2}$$

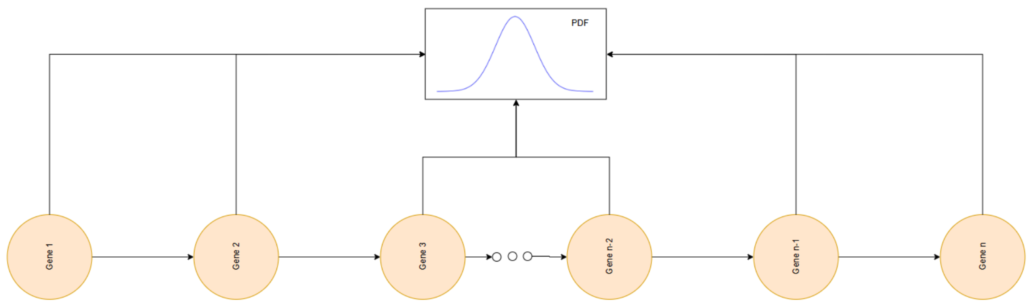


Figure 3.4: The image shows the composition of the numerical chromosome, where each gene has a PDF function

The categorical part is represented by Equation 3-3 where q_i represents the quantum individual and L the number of nodes of an individual. In addition, $p_{i1}(x)$ represents the probability mass function (PMF) that defines the probability of observing a function of all F user-defined sets of functions. Figure 3.5 shows the visual representation of the categorical chromosome.

$$q_i = [p_{i1}(x), .., p_{iL}(x)], p_{ij} \in [0, F-1] \tag{3-3}$$

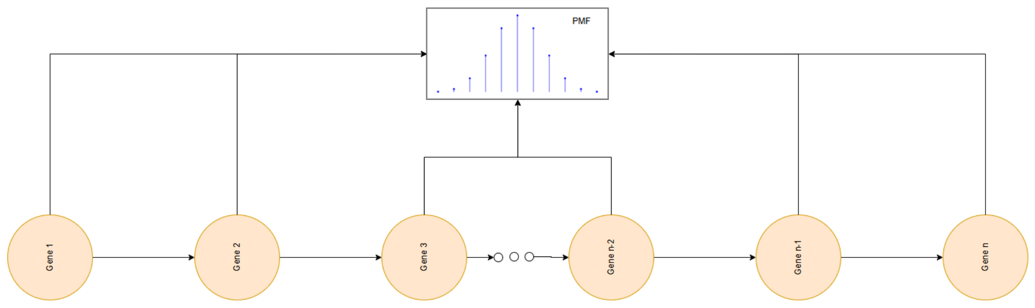


Figure 3.5: The image shows the composition of the categorical chromosome, where each gene has a PMF function

The network template proposed in [83] is a chain-like structure composed of L nodes, and every node executes a function F from the predefined function set. A fully connected layer is fixed at the end of the network to ensure structured output. Consequently, when a classical individual is processed, a decoding process is needed to transform the chained nodes into the network with its functions and hyperparameters.

The classical individual can sometimes generate infeasible network structures due to constraints in the architectural design. For example, for a given input image resolution, a maximum number of pooling operations is needed to maintain structural integrity. In Q-NAS, the parameter *penalize_number* sets an upper limit on the number of pooling layers that are decoded. Any additional pooling layers beyond this threshold are ignored during the decoding process to ensure feasibility.

3.2.2 Search Strategy

This subsection presents the steps of the Q-NAS algorithm (Algorithm 1) used in [83] and provides a brief explanation of how the search strategy works.

Algorithm 1 Q-NAS

```
$t \leftarrow 0$
Initialize $Q(t)$
while $t \leq T$ do
    Generate classical population $C(t)$ observing $Q(t)$
    if t = 0 then
        Evaluate $C(t)$
        $P(t) \leftarrow C(t)$
    else
        $C(t) \leftarrow$ recombination between $C(t)$ and $P(t)$
        Evaluate $C(t)$
        $P(t) \leftarrow$ best individuals from $[C(t) \cup P(t)]$
    end if
    $Q(t+1) \leftarrow$ update $Q(t)$ based on $P(t)$ values
    $t \leftarrow t + 1$
end while
```

The quantum population $Q(t)$ is initialized. $Q(t)$ is composed of N quantum individuals $q_i^t, i = 1, 2, .., N$. In the initialization step, every gene starts with the same probability for each value. The user can also specify custom probability values, so an initial bias towards some functions can be inserted. Subsequently, the observation process simply uses probabilities to sample the function to be observed.

Then, the classical population is generated by observing both parts of the quantum individuals' chromosomes. The classical individuals are then evaluated, and the best individuals are used to update the probabilities of the quantum individuals.

The best classical individuals guide the update of the quantum chromosome. The update of the categorical quantum chromosome aims to increase the probability that functions observed in a good-performing classical individual should be increased, while reducing the probability of the remaining functions. Q-NAS uses a simple heuristic for that. First, a random mask, based on the *update_quantum_rate* parameter, is generated to update the quantum individual. This mask defines which gene g_i will be updated, then for each node i in the mask, the following steps are executed:

1. get the function for node i in the best classical individual
2. calculate $update\_value = r \times 0.05$ where r is a random number in the interval $[0, 1]$
3. increase the probability of observing that function by *update_value*
4. decrease the probability of observing the other functions by $\frac{update\_value}{F-1}$

The update of the numerical quantum chromosome also aims to change the probability distribution of the hyperparameters based on high-performing classical individuals. To start the update, a random mask is generated based on the parameter *update_quantum_rate* to define, which $p_{ij}(x)$ are to be updated. The idea is that $p_{ij}(x)$ is changed in order to increase the probability of observing values that lead to better-evaluated individuals. This is done by changing the mean μ_{ij} and the width σ_{ij} of $p_{ij}(x)$ using the following equation (Eq. 3-4).

$$\begin{aligned} h^t &= \max_{i=1..K} c_{ij}^t - \min_{i=1..K} c_{ij}^t \\ \mu_{ij}^{t+1} &= \mu_{ij}^t + r \times (c_{ij}^t - \mu_{ij}^t) \\ \sigma_{ij}^{t+1} &= \sigma_{ij}^t + r \times (h^t - \sigma_{ij}^t) \end{aligned} \tag{3-4}$$

Where c_{ij}^t is the jth current value of a classical individual i and r is a random number in the range $[0, 1]$. In addition, h represents the difference between the maximum and minimum value of all K classical individuals. Rewriting Equation 3-4 in terms of l_{ij} and u_{ij} we have:

$$
\begin{aligned}
l_{ij}^{t+1} &= l_{ij}^{t} + r \times \left(c_{ij}^{t} - l_{ij}^{t} - \frac{h^t}{2} \right) \\
u_{ij}^{t+1} &= u_{ij}^{t} + r \times \left(c_{ij}^{t} - u_{ij}^{t} + \frac{h^t}{2} \right)
\end{aligned} \tag{3-5}
$$

The Equation 3-5 can be interpreted as the PDF defined by $p_{ij}(x)$, being shifted and narrowed in the direction of the value of the best classical individual hyperparameter.

Once the initial generation is completed, a previous population $P(t)$ will be available and the crossover can then be applied to the numerical chromosome component. In addition, Q-NAS uses the steady-state approach for selecting $P(t)$. That means that if the classical population has the size K, $P(t)$ will save the K best individuals from the set $P(t) \cup Q(t)$.

An important aspect of using a quantum individual is that each can generate one or more classical individuals. Because the observation process is stochastic, each observation of the same quantum individual may lead to different classical individuals.

3.2.3 Performance Estimation Strategy

In this subsection, the evaluation process is presented. After the observation process, each classical individual is decoded, into a neural architecture, and evaluated. The evaluation process involves the following steps:

1. train the decoded network for 50 epochs on a subset of the training set;
2. evaluate the accuracy of the network on the validation set for the five last epochs;
3. the higher validation accuracy is used as the individual fitness.

As seen from the above steps, some strategies for reducing the time consumption of Q-NAS include training for a restricted number of epochs and on a subset of the training set. Another restriction is that networks that took more than 90 minutes to train are evaluated with a fitness value equal to 0.

At the end of the evolution process, the best network found is trained using all dataset samples and for an extended number of epochs. The best architecture retrained is the one used for comparison with other works.

Finally, incorporating a quantum-inspired optimization algorithm reduces the search space for potential neural architectures. In addition, quantum-inspired strategies can explore the architecture space more intelligently, leading to faster convergence on optimal architectures while maintaining or improving performance metrics such as accuracy and robustness [83].

3.3 SegQNAS

This section presents SegQNAS [91]. SegQNAS modifies Q-NAS to search for semantic segmentation neural networks based on a U-Net like network template. In addition, since the task shifts from Q-NAS to SegQNAS, the performance estimation strategy must also change.

3.3.1 Search Space

The search space proposed for SegQNAS [91] divides the search space into topology and cell levels. The topology level is responsible for searching for the combination of blocks to compose the network, considering that the network template is U-Net-like. The following blocks were used for the topology search:

- VGG Block
- ResNet Block
- DenseNet Block
- InceptionNet Block
- Identity Block

The information of each block is presented as a diagram in Figure 3.6 and Figure 3.7, considering Conv (kxk) as a convolution $k \times k$, ReLU as an activation function, and Avg Polling as the polling function that aggregates values using the average function.

The cell is composed of a searchable block, an optional upscaling/downscaling operation depending on the cell type, and a concatenation of the skip connection and input if the skip connection is present. It is important to note that the previously presented nonscaling type occurs when neither upscaling nor downscaling is selected.

The search space's cell level is divided into three types: upscaling, downscaling, and nonscaling. The upscaling cell increases the cell level by 1 unit, doubling the feature map resolution while halving the number of channels. The downscaling cell halves the feature map resolution while doubling the number of channels, while the nonscaling cell preserves the same feature map shape as the previous cell.

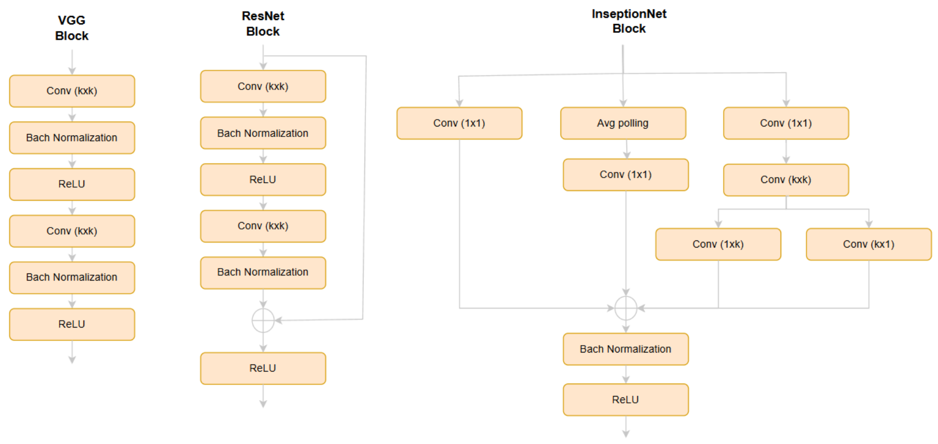


Figure 3.6: VGG16, ResNet and InceptionNet Blocks of SegQNAS. Conv(kxk) refers to a $k \times k$ convolution where k is also searchable. ReLU indicates all activations as ReLU. All blocks produce as output a feature map with the same shape as the input.

For this work, the topology and cell search space is represented by a quantum individual with probability distribution of the functions F defined by the user. Here, each function $f \in F$ combines blocks and a cell together. The Equation 3-6 represents the search space of the quantum individual, where q_i represents the quantum individual and L the number of nodes of an individual. In addition, $p_{i1}(x)$ represents the probability mass function (PMF) that defines the probability of observing a function of all F user-defined sets of functions. Figure 3.5 shows the visual representation of the numerical chromosome.

$$q_i = [p_{i1}(x), .., p_{iL}(x)], p_{ij} \in [0, F-1] \qquad (3\text{-}6)$$

The network template proposed in [91] follows a U-Net-like structure, consisting of L nodes. Each node performs a function F from the function set. The architecture has a fixed stem convolution at the beginning and fixed final convolutions to transform the feature map into a class for each pixel. However, the algorithm may generate infeasible networks, as the resulting architecture could produce an output size different from the input size. To guarantee that the network respects the design constraints, four rules were defined considering d as the cell level and D the maximum allowed level:

- Rule 1: if $d = 0$ and the cell type is *upscaling*, change the cell type to *nonscaling*.
- Rule 2: if $d = D$ and the cell type is *downscaling*, change the cell type to *nonscaling*.

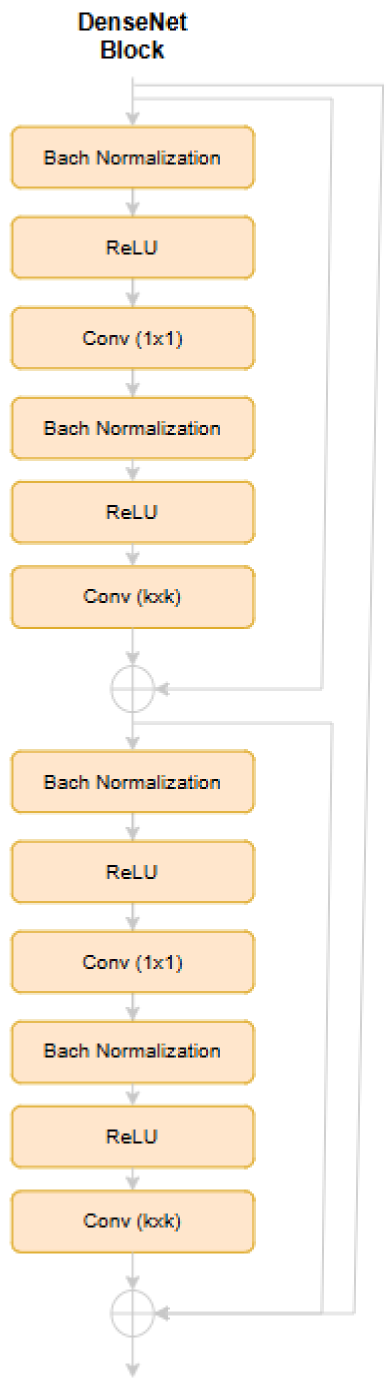


Figure 3.7: DenseNet Block of SegQNAS. Conv(kxk) refers to a $k \times k$ convolution where k is also searchable. ReLU indicates all activations as ReLU. All blocks produce as output a feature map with the same shape as the input.

- Rule 3: if $L-l=d$ and the cell type is either *downscaling* or *nonscaling* change the cell type to *upscaling*.
- Rule 4: if $L-l=d+1$ and the cell type is either *downscaling* change the cell type to *nonscaling*.

Rule 1 accounts for the constraint of not allowing levels below zero, which means that the output size of the network can never be larger than the input size. Rule 2 enforces the maximum level constraint ($d = D$), which means that the network can never have an output size smaller than the input size. Rules 3 and 4 guarantee that the output level is the same as the input ($d = 0$).

3.3.2 Search Strategy

This subsection covers the steps of the SegQNAS algorithm (Algorithm 2) used in [91] and a brief explanation on how the search strategy works.

Algorithm 2 SegQNAS

```
$t \leftarrow 0$
Initialize $Q(t)$
while $t \leq T$ do
    Generate classical population $C(t)$ observing $Q(t)$
    if t = 0 then
        Evaluate $C(t)$
        $P(t) \leftarrow C(t)$
    else
        Evaluate $C(t)$
        $P(t) \leftarrow$ best individuals from $[C(t) \cup P(t)]$
    end if
    $Q(t+1) \leftarrow$ update $Q(t)$ based on $P(t)$ values
    $t \leftarrow t+1$
end while
```

The quantum population $Q(t)$ is initialized. $Q(t)$ is composed of N quantum individuals $q_i^t, i = 1, 2, .., N$. In the initialization step, every gene starts with the same Probability Mass Function (PMF). However, the user can specify different initial values to introduce a bias toward specific functions. After that, the observation process applies probability to the sample, which function is to be selected.

Then, the classical population is generated by observing the quantum individuals. The classical individuals are evaluated, and the best-performing ones are used to update the probabilities of the quantum individuals.

The best classical individuals guide the update of the quantum chromosome, increasing the probability of selecting functions that performed well while reducing the probability of the others. The update of the quantum chromosome follows the same rule for the categorical part of the quantum individual in the Q-NAS presented in the last section. SegQNAS does not apply crossover to the generated classical individuals; however, the selection process remains the same as in the Q-NAS work presented in the last section.

It is important to note that the SegQNAS implementation does not penalize the infeasible network found during the search process. Instead, the network is adjusted to be feasible, leading to a noise searching process since the decoded network is not the same as the one that will be evaluated in the case of the infeasible network.

3.3.3 Performance Estimation Strategy

This subsection presents the process that evaluates the classical individual. After the observation process, each classical individual is decoded and evaluated. The evaluation process involves the following steps:

1. train the decoded network for 30 epochs on a subset of the training set;
2. evaluate the network's Dice-Sørensen coefficient (DSC) performance on the validation set over the last six epochs;
3. the fitness of the classical individual is calculated with an average of a 5-fold cross-validation process.

The DSC is defined by the Equation 3-7, where X is the output segmentation mask produced by the candidate network and Y is the ground truth.

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|} \tag{3-7}$$

Equation 3-8 represents the loss function used in SegQNAS.

$$Loss = 1 - DSC \tag{3-8}$$

After identifying the best architecture, this network is trained for 100 epochs using all the dataset samples. The retrained architecture is then used to compare with other works.

Finally, the SegQNAS enhances the efficiency and effectiveness of designing neural networks for the semantic segmentation task and, due to a quantum-inspired algorithm, it allows SegQNAS to explore the architecture search space more intelligently, reducing the computational burden [91].

4
SegQSNAS

In this chapter, we present our method, SegQSNAS, which evolves SegQNAS[91] designed to search exclusively for symmetrical U-Net-like networks. SegQSNAS differs from SegQNAS in the following aspects:

- **Search space**: SegQSNAS reduces the search space by searching only for symmetrical networks and, because of that, the output size will always be the same size as the input size for every solution, which removes the need of the SegQNAS feasibility check for the network depth. In addition, a self-attention block was added to the search space to improve the results, and the MobileNet and EfficientNet blocks were added to allow for more lightweight networks in the search space;
- **Search Strategy**: Q-NAS [83] performed a crossover operator only for the numerical part of the chromosome, without a crossover for the categorical part. In SegQSNAS, a two-point crossover[90] operation was added to the evolutionary process, providing an improved and less noisy evolution than the original one;
- **Performance Estimation Strategy**: The DSC function used in SegQ-NAS calculates the accurate values for binary problems. However, for multi-class problems, the computed value was overestimated because the numerator in the DSC formula does not fully account for the true overlap between multiple labels. At the same time, the denominator is inflated by the cumulative sum across all labels. The DSC equation was designed to be used in binary problems, however for multi-class problem the "one vs. all" (OvA) strategy must be used. To solve this problem, the DSC function was corrected for both binary and multi-class problems.

The following sections will detail these aspects. In addition, SegQSNAS focuses exclusively on the evolution of the network architecture; thus, it does not cover the hyperparameter search, because it does not improve the results, as confirmed by [83] work.

4.1 Search Space

The search space proposed for SegQSNAS is based on the work of [91] which divides the search space into topology and cell levels.

The topology level is responsible for searching for the combination of blocks the network should be composed of, considering that the network template is U-Net-like. The cell level of the search space is again defined into two types: downscaling or nonscaling. The downscaling cell halves the feature map resolution while doubling the number of channels. The nonscaling cell preserves the same feature map shape as the previous cell.

The upscaling cell is not part of the search space because on the SegQSNAS design, the first half of the architecture has only downscaling and nonscaling options since when the other half is decoded, all downscaling is replaced by an upscaling cell. The upscaling cell increases the cell level by 1 unit, causing the feature map resolution to double while halving the number of channels.

The chromosome representation for the SegQSNAS implementation is categorical that is the same as SegQNAS and the representation can be visualized in the Figure 3.5.

For a better understanding of the search space, Figure 4.1 shows an example of the network that can be found using the SegQNAS method. The orange nodes are the ones that are part of the search space and they are part of the encoding part of the U-NET template. The gray part is generated by mirroring the encoding part and changing downscaling cells to upscaling cells. It is important to note that the SegQNAS [91] method allows upscaling in the encoding part, but the proposed method does not allow it to be close to the U-NET architecture.

SegQSNAS removed 3×3 steam convolution fixed as the first network node. However, the final convolution was kept with the same configuration as was implemented in the SegQNAS work, which is a fixed operation responsible for getting feature map as input and transforming it into the segmentation maps. The final convolution is a 3×3 convolution with N kernels, where N is the number of classes of the dataset. If $N = 2$, the activation function of the final convolution is defined as a sigmoid. If $N > 2$, the activation function is defined as softmax.

The batch normalization and ReLU activation block are implicit for all convolutional operations.

The searchable block is a block belonging to the SegQSNAS function set shown in Figure 4.2 and listed below:

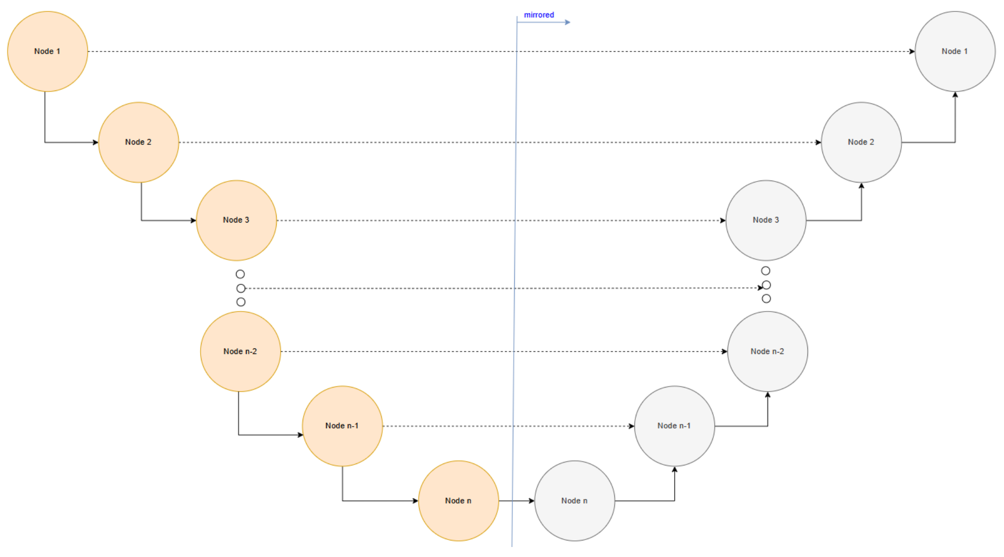


Figure 4.1: The image shows the example of network that could be found using SegQSNAS. The encoding part is presented in orange and the decoding and mirrored part is presented in gray.

- VGG Block
- ResNet Block
- DenseNet Block
- InceptionNet Block
- Identity Block
- Self-Attention Block
- MobileNet Block
- EfficientNet Block

The VGG, ResNet, DenseNet, InceptionNet and Identity blocks were implemented in SegQNAS [91], as shown in Figure 3.6 and in Figure 3.7. However, the Self-Attention, MobileNet and EfficientNet blocks were a SegQSNAS implementation presented in Figure 4.2.

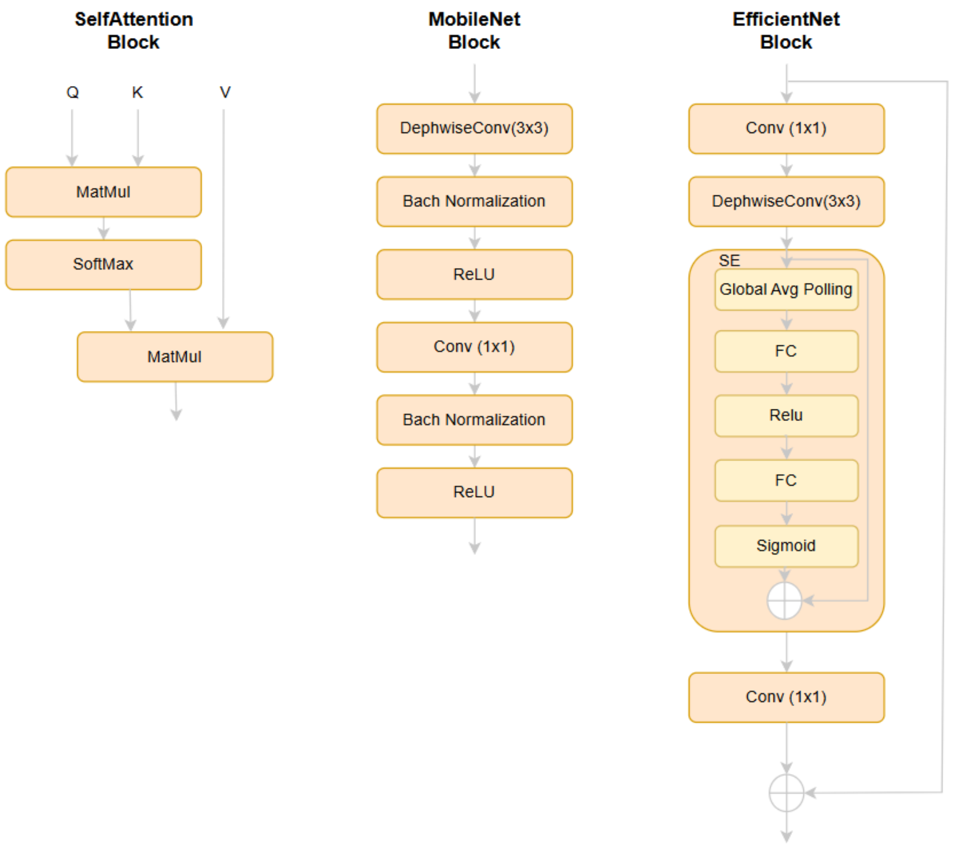


Figure 4.2: Blocks of SegQSNAS. Q stands for query vector, K for key vector and V for values vector. MatMult refers to a dot product between two vectors. Softmax is an activation function. Conv(1x1) refers to a 1×1 convolution. ReLU indicates all activations as ReLU. DepthwiseConv(3x3) is a channel-wise 3×3 convolution for efficient feature extraction. Global Avg Polling is a pooling operation that calculates the average of each feature map. FC stands for Fully Connected layer and Sigmoid is an activation function. In addition to it, there is also the Squeeze-and-Excitation (SE) presented in EfficientNet Block. All blocks produce as output a feature map with the same shape as the input.

Integrating self-attention in CNNs improves feature extraction by enabling the model to focus on relevant spatial features while ignoring less important ones. Adding attention modules to existing architectures enhances their performance in complex tasks[92, 93].

Adding MobileNet blocks was designed to reduce the number of parameters and computational complexity[94, 95, 96]. The Efficient block is enhanced by the Squeeze-and-Excitation (SE), which recalibrates channel-wise feature responses, allowing the network to focus on the most informative features[97].

EfficientNet employs a compound scaling method that uniformly scales all depth, width, and resolution dimensions. This approach allows EfficientNet to achieve state-of-the-art accuracy while being smaller and faster than other architectures[98, 99].

To summarize the search space, each cell can be defined by three parameters:

1. The cell type (downscaling, nonscaling).
2. The block (Self-Attention, MobileNet, EfficientNet, VGG, ResNet, DenseNet, InceptionNet, Identity).
3. The kernel size ($k \in \{3, 5, 7\}$) of the block convolutional operations.

The function set of SegQSNAS is the set of all combinations of cell types, blocks and kernel sizes. A caveat is that the Identity block does not need a kernel size specification as it performs no convolutional operation. All functions of the search space are listed below:

- vgg_d_3: VGG downscaling 3×3
- vgg_d_5: VGG downscaling 5×5
- vgg_d_7: VGG downscaling 7×7
- vgg_n_3: VGG nonscaling 3×3
- vgg_n_5: VGG nonscaling 5×5
- vgg_n_7: VGG nonscaling 7×7
- res_d_3: ResNet downscaling 3×3
- res_d_5: ResNet downscaling 5×5
- res_d_7: ResNet downscaling 7×7
- res_n_3: ResNet nonscaling 3×3
- res_n_5: ResNet nonscaling 5×5
- res_n_7: ResNet nonscaling 7×7
- den_d_3: DenseNet downscaling 3×3
- den_d_5: DenseNet downscaling 5×5
- den_d_7: DenseNet downscaling 7×7
- den_n_3: DenseNet nonscaling 3×3
- den_n_5: DenseNet nonscaling 5×5
- den_n_7: DenseNet nonscaling 7×7
- inc_d_3: InceptionNet downscaling 3×3

- inc_d_5: InceptionNet downscaling 5×5
- inc_d_7: InceptionNet downscaling 7×7
- inc_n_3: InceptionNet nonscaling 3×3
- inc_n_5: InceptionNet nonscaling 5×5
- inc_n_7: InceptionNet nonscaling 7×7
- $mobile\_n\_v1$: MobileNet nonscaling
- $mobile\_d\_v1$: MobileNet downscaling
- eff_n: EfficientNet nonscaling
- eff_d: EfficientNet downscaling
- $selfatt$: Self-Attention nonscaling
- ide: Identity

It is important to note that since SegQSNAS uses only symmetrical networks and searches only half of the search space, all cell types must be nonscaling or downscaling. When the second half is composed, all downscaling in the first half turns into upscaling in the other half. This logic permits SegQSNAS not to need to check the networks' feasibility, since all downscaling in the first part will have an upscaling with the same ratio in the other half of the network.

4.2 Search Strategy

This subsection covers the steps of the SegQSNAS algorithm (Algorithm 3) and briefly explains how the search strategy works.

The quantum population $Q(t)$ is initialized. $Q(t)$ is composed of N quantum individuals $q_i^t, i = 1, 2, .., N$. In the initialization step, every gene starts with the same PMF. Optionally, the user may specify different values, inserting initial bias towards some functions. After that, the observation process simply uses probabilities to sample which function is to be observed.

The classical population is generated by observing the quantum individuals. An important aspect of using a quantum individual is that, due to the stochastic nature of the observation process, each quantum individual can generate one or more classical individuals. After that, the classical individuals are evaluated, and the best individuals are used to update the probabilities of the quantum individuals.

The evaluation of each classical individual uses the fitness cache in order to reduce the usage of computing resources and the usage time during the

Algorithm 3 Q-NAS

```
$t \leftarrow 0$
Initialize $Q(t)$
while $t \leq T$ do
    Generate classical population $C(t)$ observing $Q(t)$
    if t = 0 then
        Evaluate $C(t)$
        $P(t) \leftarrow C(t)$
    else
        $C(t) \leftarrow$ recombination between $C(t)$ and $P(t)$
        Evaluate $C(t)$
        $P(t) \leftarrow$ best individuals from $[C(t) \cup P(t)]$
    end if
    $Q(t+1) \leftarrow$ update $Q(t)$ based on $P(t)$ values
    $t \leftarrow t + 1$
end while
```

evolution process[100]. The fitness cache allows the network to be evaluated only once, the first time it appears during the evolution process; any time the same network appears, it uses the cached value and does not train the network again.

The best classical individuals guide the update of the quantum chromosome. The update of the quantum chromosome aims to increase the probability that functions observed in a good-performing classical individual should be increased while the remaining functions decrease. SegQSNAS uses the same heuristic as Q-NAS[83] for that. First, a random mask, based on the *update_quantum_rate* parameter, is generated to update the quantum individual. This mask defines which g_i will be updated, then for each node i in the mask, the following steps are executed:

1. get the function for node i in the best classical individual
2. calculate $update\_value = r \times 0.05$ where r is a random number in the interval $[0, 1]$
3. increase the probability of observing that function by *update_value*
4. decrease the probability of observing the other functions by $\frac{update\_value}{F-1}$

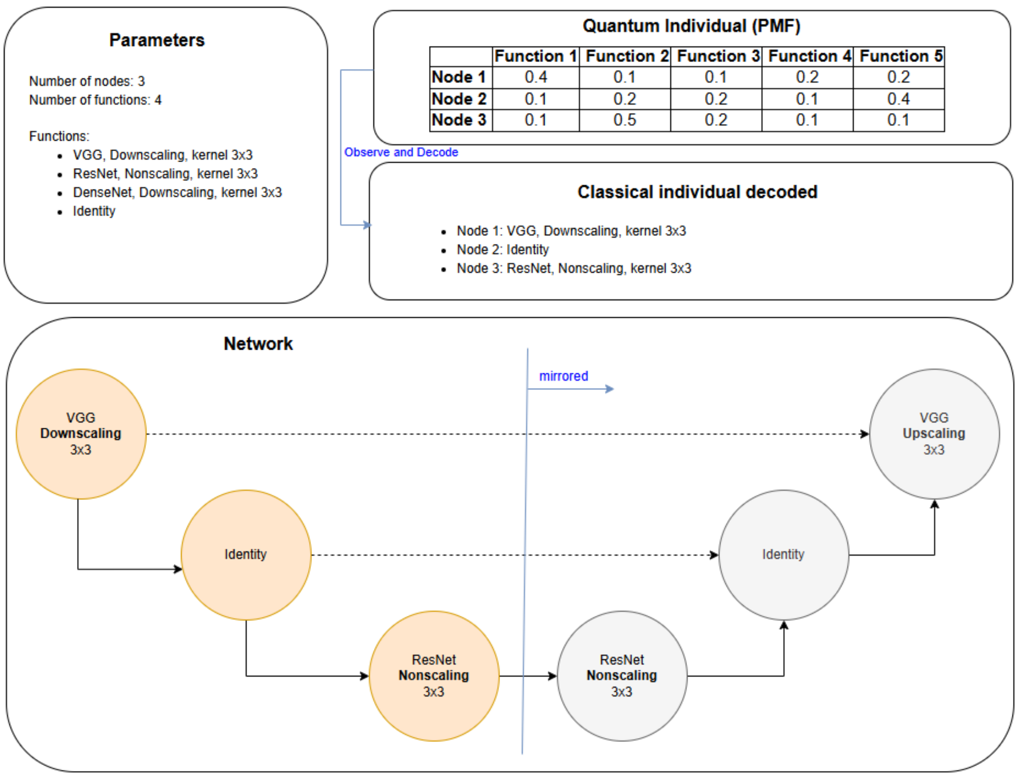
After the initial generation, there will be a previous $P(t)$, and consequently, it will be possible to apply the crossover operation. In SegQSNAS, the two-point crossover operation is used to improve the evolutionary process convergence [101]. In addition, SegQSNAS uses the steady-state approach for selecting $P(t)$ which is the same implemented in Q-NAS[83] and used in

SegQNAS[91]. That means that if the classical population has the size K, $P(t)$ will save the K best individuals from the set $P(t) \cup Q(t)$.

As mentioned in Chapter 3, SegQNAS' search strategy could lead to an infeasible network, which forced some changes at the cell level during the decoding process of the individual to the network, always to create a feasible individual. However, due to the search strategy applied to SegQSNAS, which uses a symmetric network, the same problem does not occur since all the solutions in the search space are feasible.

SegQNAS has only feasible solutions because it only searches for the first half of a U-Net-like network. Consequently, the search space consists exclusively of nonscaling and downscaling cells. It allows the network to always return to the first level when a network is decoded, since all downscaling cells change to upscaling cells on the other half of the network.

Figure 4.3 shows how observation of the quantum individual and decoding of the classical individual occur during the evolution process. The example provided is based on a search for an architecture with three nodes in the encoding part using four available functions, which means that the final architecture could have at most six nodes. The current probability distribution for each node of a quantum individual is presented in the image and, for the observation, the function with higher probability in each node. After observation, function 1 was selected for the first node, function 5 for the second node, and function 2 for the last node. After decoding, it will be possible to identify that the first node is composed by a VGG block with downscaling cell and kernel $3x3$, the second node is composed by an identity block and the last node is composed by a ResNet block with nonscaling cell and kernel $3x3$.



Parameters

Number of nodes: 3
Number of functions: 4

Functions:
- VGG, Downscaling, kernel 3x3
- ResNet, Nonscaling, kernel 3x3
- DenseNet, Downscaling, kernel 3x3
- Identity

Quantum Individual (PMF)

	Function 1	Function 2	Function 3	Function 4	Function 5
Node 1	0.4	0.1	0.1	0.2	0.2
Node 2	0.1	0.2	0.2	0.1	0.4
Node 3	0.1	0.5	0.2	0.1	0.1

Classical individual decoded

- Node 1: VGG, Downscaling, kernel 3x3
- Node 2: Identity
- Node 3: ResNet, Nonscaling, kernel 3x3

Figure 4.3: The image shows the example of a observation of the quantum individual and the decoding of the classical individual.

4.3 Performance Estimation Strategy

In this section, the evaluation process for the classical individual is presented. Following the observation phase, each classical individual is decoded and evaluated. The evaluation process involves the following steps:

1. train the decoded network for a limited number of epochs on a subset of the training set;
2. evaluate the network by computing the average DSC across all classes, excluding the background class;
3. calculate the validation score of the network on the validation set for the last 20% epochs;
4. the fitness of the classical individual is calculated with an average of a 5-fold cross-validation process.

The DSC function used in SegQNAS[91] (Equation 3-7) calculates the correct values only for binary problems. In order to solve this for multi-class problems, the DSC function was corrected to calculate the DSC for each class of the problem except for the background class ($c = 1$) and averaging the result of each class c in all possible classes of the problem (C), as presented in Equation 4-1.

$$AVG\_DSC = \frac{1}{|C|}\sum_{i=1}^{|C|} \mathrm{DSC}_i, c \in C, c > 1 \tag{4-1}$$

The loss function was adjusted by using the update AVG_DSC instead of DSC, being presented in Equation 4-2.

$$Loss = 1 - AVG\_DSC \tag{4-2}$$

The training protocol employs ADAM optimizer with a learning rate 10^-3 and polynomial decay 0.9. Data augmentation techniques were applied with horizontal flipping, shifting, scaling, rotating, and random crop operations.

Cross-validation was employed for each evaluated individual to mitigate noise in the evaluation process. The individual evaluation using the validation set for the last 20% epochs is presented in Equation 4-3, considering that for each epoch, all classical individuals have the evaluation performed by applying the 5-fold as presented in Equation 4-4.

The AVG_DSC is calculated using the validation set for every fold for each classical individual for the last 20% epochs applying the 5-fold were then averaged. Finally, this average was taken as the fitness value of the individual that generated the network.

$$V\_AVG\_DSC = \frac{1}{5}\sum_{i=1}^{5} \mathrm{AVG\_DSC}_i \tag{4-3}$$

$$\text{5-fold cross validation} = \frac{1}{5}\sum_{i=1}^{5} \mathrm{V\_AVG\_DS}_i \tag{4-4}$$

This evaluation strategy during the evolution process was proposed due to resource constraints. Some techniques, such as training for a reduced number of epochs and samples, were applied to reduce the time consumed in the performance estimation process.

After finding the best architecture, 5-fold cross-validation is performed, and each network is trained for a higher number of epochs using the whole dataset.

5 Experiments

In this chapter, we present the experiments conducted to evaluate SegQS-NAS. The first section presents all the datasets used, the second section presents the setup used to perform the experiments, and finally, the third section presents all the experiments and discusses the results.

5.1 Datasets

This section describes the datasets used to perform the experiments. The four datasets used in this work were extracted from the Medical Segmentation Decathlon challenge [21]. The datasets were selected to better understand how SegQSNAS works on binary and multi-class datasets.

5.1.1 Spleen Dataset

The spleen dataset contains 61 3D-CT scans of patients undergoing chemotherapy treatment for liver metastases at Memorial Sloan Kettering Cancer Center. From the 61 3D-CT scans, ground truth was provided for 41 scans. In addition, this dataset has a binary target, where '0' represents the background class and '1' the spleen.

The annotation was semi-automatized using the Scout Application, followed by manual adjustment by an expert abdominal radiologist. Some examples of images from the dataset are presented in Figure 5.1.

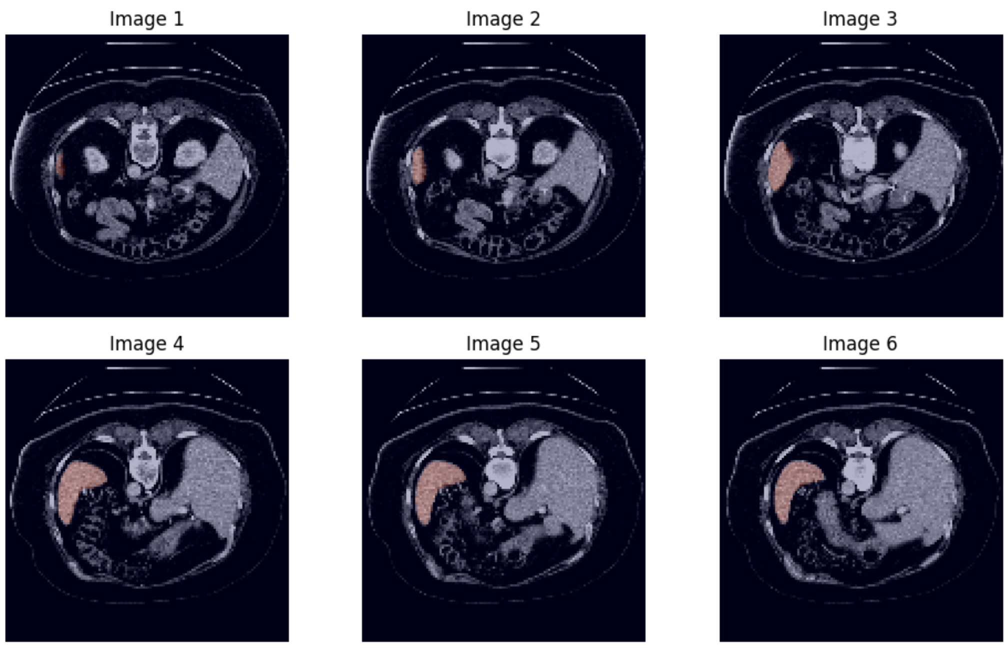


Figure 5.1: Six sampled images from spleen dataset. The orange mask represents the spleen.

5.1.2 Prostate Dataset

The prostate dataset provided by Radboud University contains 48 prostate multiparametric magnetic resonance imaging (MRI) scans. It has two target classes: the peripheral zone (PZ) of the prostate and the transition zone (TZ). Of the 48 mp-MRI scans, ground truth was provided for 32 scans.

The image was generated from transverse T2-weighted scans and the annotations were generated manually for the entire dataset. Figure 5.2 presents some examples of images from the dataset.

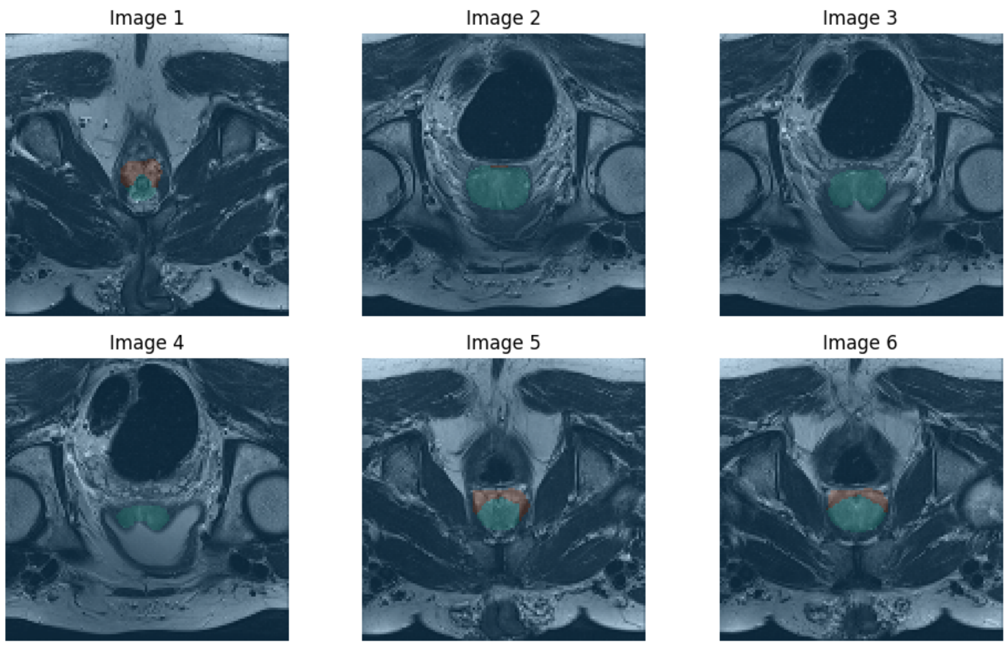


Figure 5.2: Six sampled images from prostate dataset generated from transverse T2-weighted scans. The orange mask represents the prostate peripheral zone (PZ) and the green mask represents the transition zone (TZ).

5.1.3 Liver Dataset

The liver dataset contains 201 contrast-enhanced CT images originally provided from several clinical sites. It has two target classes: the liver and cancer. Of the 201 contrast-enhanced CT scans, ground truth was provided for 131 scans.

The images were generated with an in-plane resolution of 0.5 to 1.0 mm, and slice thickness of 0.45 to 6.0 mm. The radiologists manually generated annotations for the entire dataset. Some examples of images from the dataset are presented in Figure 5.3.

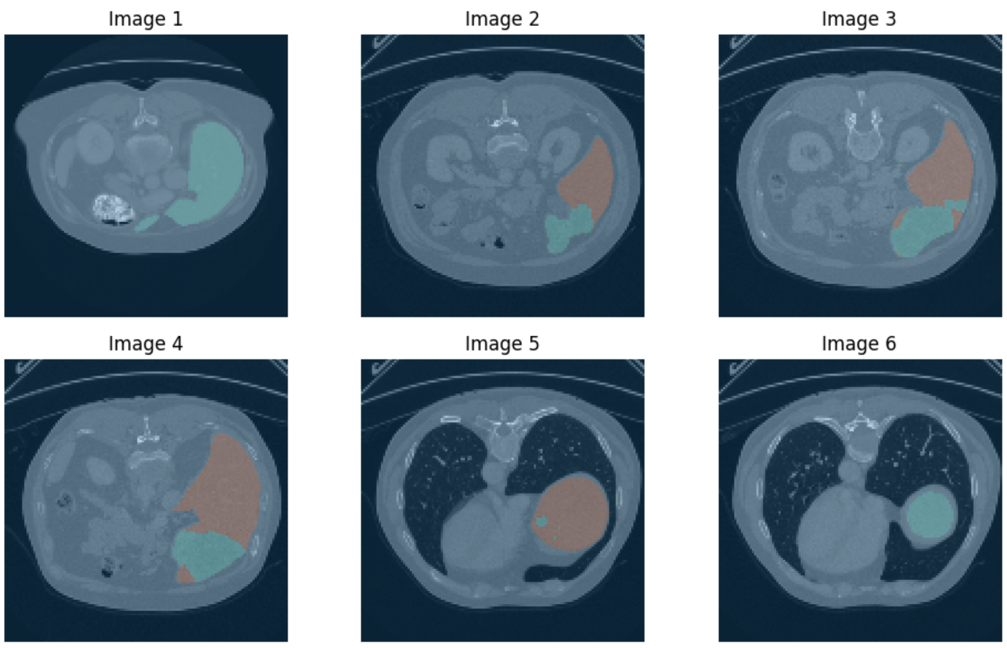


Figure 5.3: Six sampled images from liver dataset. The orange mask represents the liver and the green mask represents the cancer.

5.1.4 Brain Tumor Dataset

The brain tumor dataset contains 750 multiparametric MRI scans with three target classes: edema (E), enhanced tumor (ET), and non-enhanced tumor (NET). Of the 750 multiparametric MRI scans, ground truth was provided for 484 scans.

The images were generated from native (T1) and post-Gadolinium(Gd) contrast T1-weighted (T1-Gd), native T2-weighted (T2), and T2 FluidAttenuated Inversion Recovery (T2-FLAIR) volumes. A board-certified expert neuroradiologist approved the annotation of the entire dataset. Figure 5.4 presents some examples of images from the dataset.

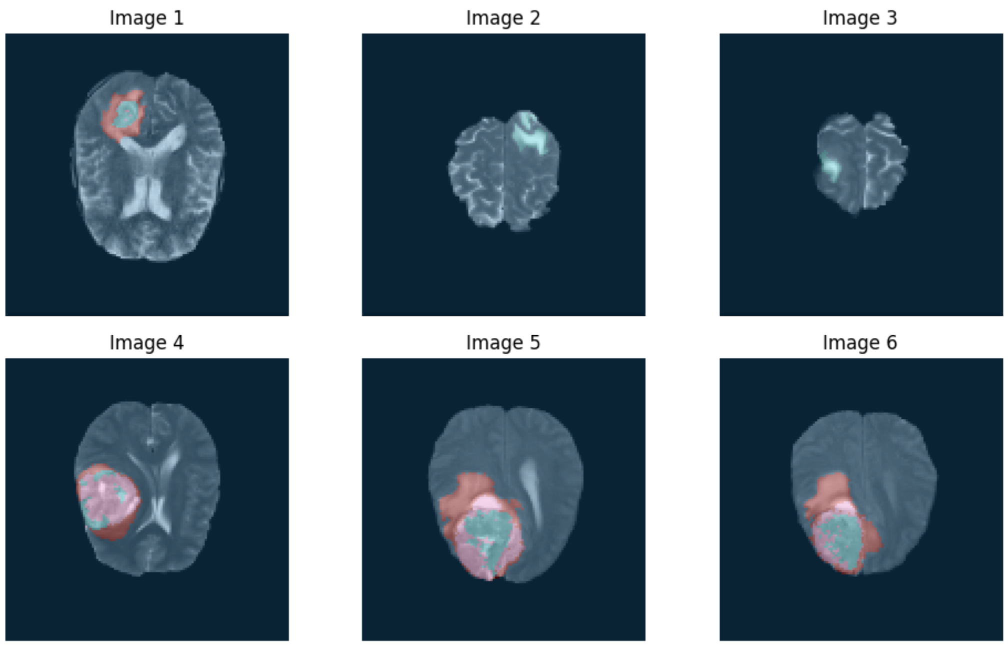


Figure 5.4: Six sampled images from brain tumor dataset generated from transverse T2-weighted scans. The orange mask represents the edema, the green mask represents the non-enhanced tumor and the pink mask represents the enhanced tumor.

5.2 Experimental Setup

The experiments were executed on a GPU cluster with three A30 24Gb GPUs, 64 CPUs of Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz and 188 GiB of RAM.

5.3 Experiments and Results

The purpose of the experiments is to evaluate the proposed adaptation of SegQNAS, named SegQSNAS. Four experiments were designed using different datasets with binary and multiclass targets:

- Experiment 1: Spleen Dataset;
- Experiment 2: Prostate Dataset;
- Experiment 3: Liver Dataset;
- Experiment 4: Brain Tumor Dataset.

The function set (the search space) of SegQSNAS is the set of all cell types, blocks and kernel sizes combinations, as specified in Chapter 4. As mentioned before, the Identity block performs no operation. All functions of the search space, that was presented in Chapter 4, are listed below:

– vgg_d_3: VGG downscaling 3×3
– vgg_d_5: VGG downscaling 5×5
– vgg_d_7: VGG downscaling 7×7
– vgg_n_3: VGG nonscaling 3×3
– vgg_n_5: VGG nonscaling 5×5
– vgg_n_7: VGG nonscaling 7×7
– res_d_3: ResNet downscaling 3×3
– res_d_5: ResNet downscaling 5×5
– res_d_7: ResNet downscaling 7×7
– res_n_3: ResNet nonscaling 3×3
– res_n_5: ResNet nonscaling 5×5
– res_n_7: ResNet nonscaling 7×7
– den_d_3: DenseNet downscaling 3×3
– den_d_5: DenseNet downscaling 5×5
– den_d_7: DenseNet downscaling 7×7
– den_n_3: DenseNet nonscaling 3×3
– den_n_5: DenseNet nonscaling 5×5
– den_n_7: DenseNet nonscaling 7×7
– inc_d_3: InceptionNet downscaling 3×3
– inc_d_5: InceptionNet downscaling 5×5
– inc_d_7: InceptionNet downscaling 7×7
– inc_n_3: InceptionNet nonscaling 3×3
– inc_n_5: InceptionNet nonscaling 5×5
– inc_n_7: InceptionNet nonscaling 7×7
– $mobile\_n\_v1$: MobileNet nonscaling
– $mobile\_d\_v1$: MobileNet downscaling
– eff_n: EfficientNet nonscaling
– eff_d: EfficientNet downscaling
– $selfatt$: Self-Attention nonscaling

– *ide*: Identity

In all subsequent experiments, the dataset was divided into training and test sets, with 80% of the images allocated for training and 20% for testing. During the evolution and retraining process, the evaluation was performed using a five-fold cross-validation applied to the train set. Furthermore, all images in the dataset for all experiments were Z-normalized. The following subsections provide further details on the experiments and present the results.

To provide a clearer comparison with the SegQSNAS search algorithm, a Random Search baseline was established for each experiment. In each experiment, the random search algorithm selected 50 random solutions, which were then trained for 100 epochs using the complete dataset. In particular, each of these randomly chosen solutions was generated from the same search space as SegQSNAS, and their full architecture was constructed by mirroring the encoding portion, defined by the search space, in the decoding part taking into account that downscaling cells in the encoding part become upscaling cells in the decoding part.

5.3.1 Experiment 1 - Spleen Dataset

The spleen dataset contains 41 3D-CT scans with annotations, which were used to generate the dataset for this experiment.

It is essential to note that although the machine used for this experiment has three GPUs, the resources are shared, and only one GPU was available when the experiment was executed.

The evolution parameters applied to this experiment are presented in Table 5.1. The number of nodes represents the maximum number of chained nodes allowed for the first half of the architecture, meaning that the maximum size of the architecture is 10. The number of quantum individuals indicates the number of quantum individuals that are instantiated and used to generate new individuals. The repetition parameter indicates the number of observations each quantum individual will undergo. The update quantum rate is a number that varies from 0 to 1, when 0 means that there will be no update for the quantum individual and 1 means that the update will always be done. The update quantum gen is a number that varies from 0 to 1, when 0 means that there will be no update for the chromosome of the quantum individual and 1 means that the update will always be done. The probability mass functions represent the probability distribution of the categorical chromosome, which corresponds to the functions of the architecture. In this case, all blocks of the search space have the same probability, uniformly distributed across all functions designed with the same block. The number of generations represents the duration of the evolutionary process. The dataset, image resolution and batch size are the parameters used to train each classical individual during the number of epochs described in the training epochs parameter and evaluated in the last e epochs described in the evaluation epochs parameter.

All parameters for this experiment are the same as those used on the SegQNAS work for better comparison since SegQSNAS is a modified version of SegQNAS.

Table 5.1: SegQSNAS Evolution parameters

Parameter	**Value**
Number of nodes	5
Number of quantum individual	9
Repetition	1
Update quantum rate	0.9
Update quantum gen	1.0
Probability mass function	Same probability for every function
Number of generations	100
Dataset	entire dataset
Image Resolution	128x128
Batch Size	32
Training Epochs	30
Evaluation Epochs	last 6

Due to the stochastic nature of this experiment, we ran four evolutionary processes and the best result is presented. The evolution process is illustrated in Figure 5.5, where it can be observed that every time a new solution with a better evaluation is found, the difference between the ten best classical individuals increases. However, due to the crossover usage, the algorithm exploits it more and the other classical individuals converge faster. This behavior can be observed through the bars presented in Figure 5.5, for example, in generation 1 all individuals are very diverse and, because of that, it has a higher bar, however, in 15 generations the bar size reduces to the minimum before the evolutionary process found better solutions and the difference between the individuals increases and the bar size also increases.

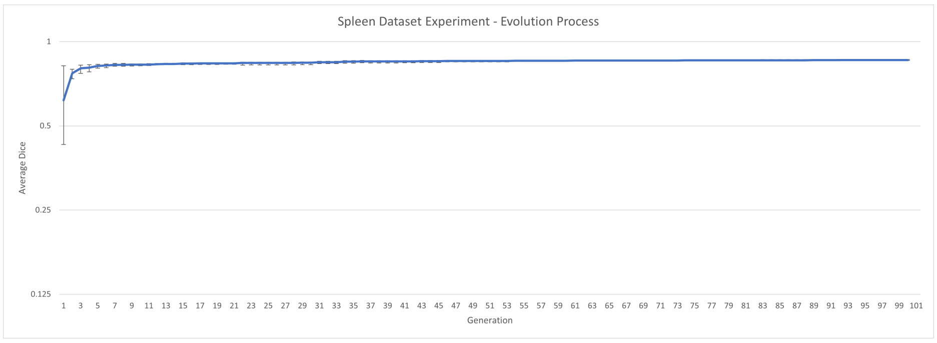


Figure 5.5: Classical individual fitness during process evolution for spleen dataset experiment

The evolutionary process converged relatively early in this experiment, and the optimal architecture was identified at generation 53. Therefore, we can conclude that it does not need all 100 generations to converge, and if there were fewer generations, it would lead to a better GPU days score. Table 5.2 presents the best architecture found.

Table 5.2: Best architecture found for spleen dataset experiment

Node	**Description**
vgg_d_3	VGG downscaling 3×3
eff_d	EfficientNet downscaling
mobile_d_v1	MobileNet downscaling
res_d_7	ResNet downscaling 7×7
inc_d_7	InceptionNet downscaling 7×7
inc_u_7	InceptionNet upscaling 7×7
res_u_7	ResNet upscaling 7×7
mobile_u_v1	MobileNet upscaling
eff_u	EfficientNet upscaling
vgg_u_3	VGG upscaling 3×3

The best individual has a classic U-Net-like architecture that only contains the downscaling cell in the first half, compressing the information; then, as imposed by the mirror strategy of the SegQSNAS approach, the architecture comprises only the upscaling cell in its second half. This design, with its five levels, typically results in a higher parameter count as a result of the increased depth. In addition, the dense block was not utilized, likely because the prostate dataset task is not particularly complex. During the evolution, the self-attention block was not considered a good choice for this problem, and it is following the opposite direction of recent work that shows improvements in the final results. However, the usage of self-attention block causes, in many individuals, out of memory of the GPU due to hardware limitations, and it causes the evolution to not consider it as a good option to the best architecture. Additionally, the fact that the identity block is not used means that a shallower architecture is not enough to achieve good results in this dataset.

Changes in the function distribution of quantum nodes from one quantum individual are presented when comparing Figure 5.6, which contains the function distribution at the beginning of the evolutionary process where all values are the same, with Figure 5.7, which presents the function distribution of quantum nodes from one quantum individual at the end of the evolutionary process. Due to the high cardinality of functions in this experiment, the division of the stacked bar was created concatenating all functions from the block family. In this case, it is possible to notice that the probability of Identity, DenseNet and SelfAttention blocks are the smallest in all nodes and converges to the best architecture found that has none of those blocks selected.

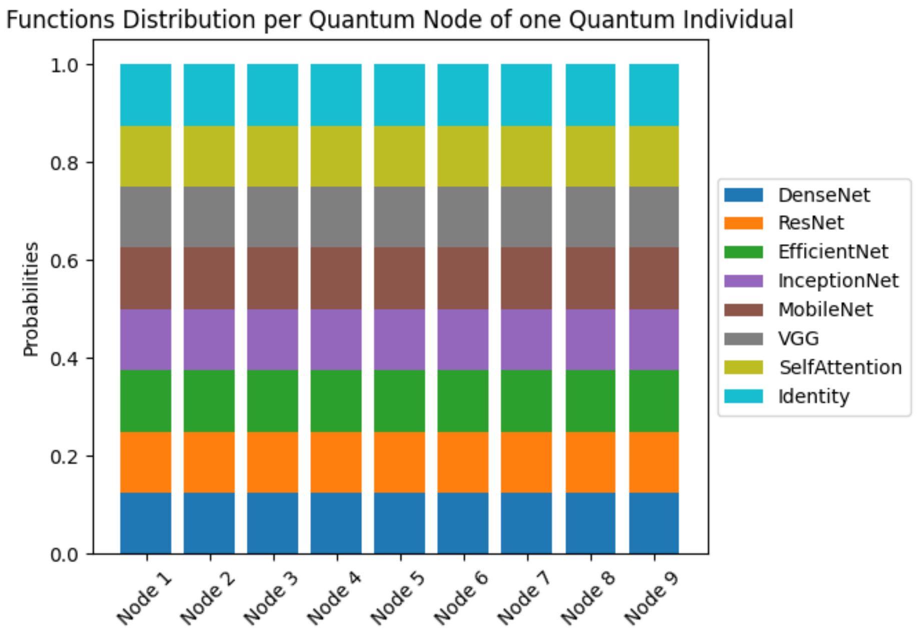


Figure 5.6: Functions distributions of a quantum individual at the start of the evolutionary process for the spleen dataset experiment

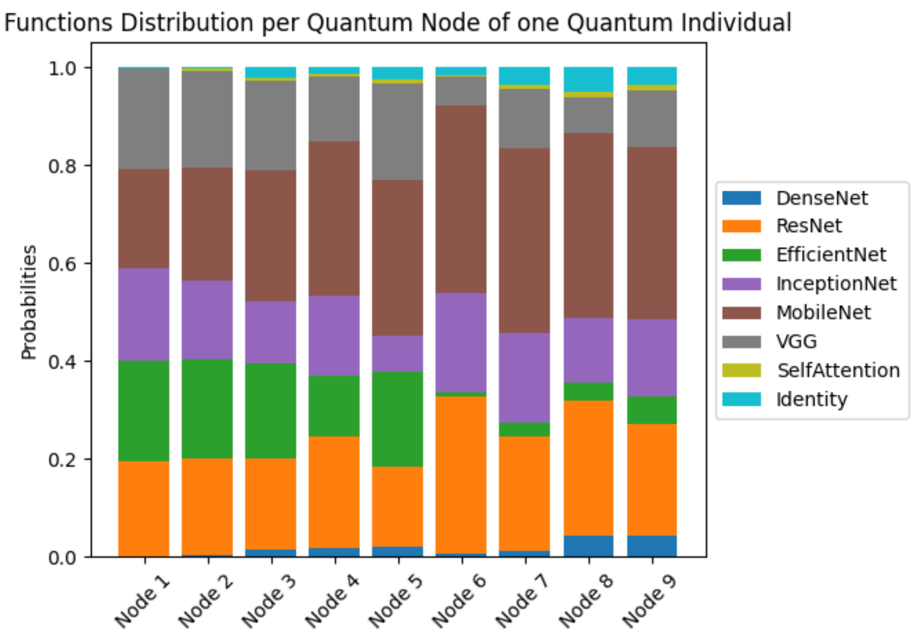


Figure 5.7: Functions distributions of a quantum individual at the end of the evolutionary process for the spleen dataset experiment

After finding the best architecture during the evolution process, the model is retrained using all dataset images for 100 epochs with 100 initialization; other training parameters presented in Table 5.1 were not changed for the retraining process. The number of initializations is selected as high as possible for a better evaluation score.

Figure 5.8 shows the output of the best network for three images sampled from the test set. It is possible to see that the predicted label is very similar to the real one.

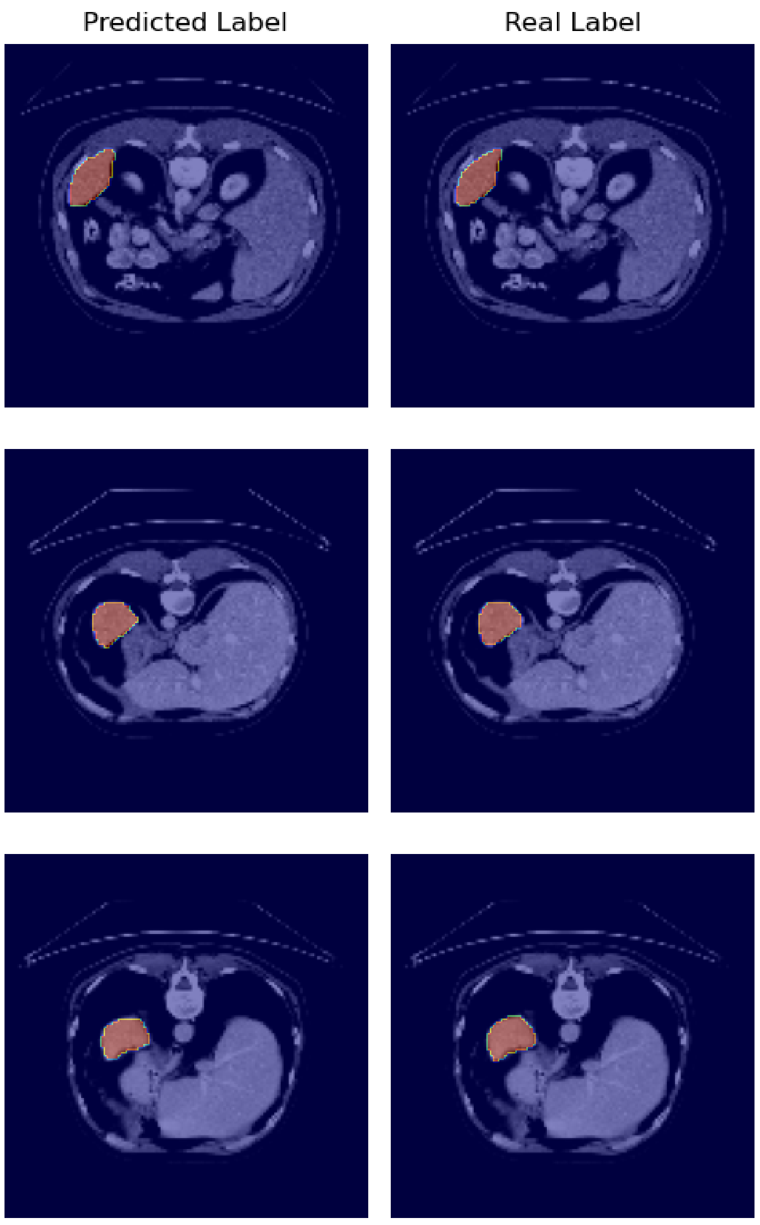


Figure 5.8: Three sampled images from the spleen dataset with real and predicted labels. The orange mask represents the spleen.

Table 5.3 compares the best result of SegQSNAS in the test set with other works.

Table 5.3: Spleen Dataset Experiments Results

Experiment	**DSC**	**#Params**	**GPU Days**
Random Search	0.9010 ± 0.004	1.6×10^6	0.34
SegQSNAS	0.9358 ± 0.004	2.5×10^6	6.45
SegQNAS_2[91]	0.9560 ± 0.004	5.3×10^6	4.15
SegQNAS_4[91]	0.9582 ± 0.008	1.0×10^6	4.29
UNETR[102]	0.9640	92.6×10^6	NA

The SegQNAS_2 experiment was designed to generate a U-Net network, similar to what occurs with SegQSNAS, by limiting the search space to a symmetric network only. The SegQNAS_4 experiment has a search space that concatenates topology search with cell search, the same as that used in SegQSNAS. The UNETR model uses a combination of CNNs and transformers, which allows better feature extraction and, consequently, a better performance in image tasks. Because of this characteristic, the UNETR network has a higher parameter count.

As presented in Table 5.3, UNETR outperformed SegQNAS and SEGQSNAS, but only by a maximum of 0.03 in the DSC score. However, the marginal improvement cost has almost 93 times more parameters than the lighter model presented.

When SegQNAS and SegQSNAS are compared, similar performance and parameter counts can be observed. SegQNAS achieved the second highest DSC score with slightly fewer parameters. However, it should be noted that the architecture discovered in the SegQNAS_4 experiment presented an upscaling cell in the first half of the architecture, and this cell type is not part of the SegQSNAS search space.

On the other hand, the SegQNAS_2 experiment discovers an architecture that is part of the SegQSNAS search space. Compared to it, SegQSNAS has a close dice score, with less than half the parameters of the second SegQNAS experiment.

It is important to notice that the cache system was made using RAM memory, so on every restart it resets. It cause the GPU Days on SegQSNAS experiment to be higher in this case.

This experiment only validates that the quantum algorithm presented in SegQNAS and SegQSNAS can find architectures with fewer parameters that achieve close performance compared to state-of-the-art architectures. In addition, the baseline random search corroborates this finding, as it does not identify an optimal architecture when compared to SegQNAS and SegQSNAS.

5.3.2 Experiment 2 - Prostate Dataset

The prostate dataset contains 32 prostate multiparametric MRI scans with annotations, which were used to generate the dataset for this experiment.

In this experiment, only one of the three GPUs on the machine was used due to the availability of shared resources.

The evolution parameters applied to this experiment are identical to those used in the previous experiment (see Table 5.1). Again, all parameters for this experiment are the same as those used in the SegQNAS work for a better comparison, since SegQSNAS is a modified version of SegQNAS.

Due to the stochastic nature of this experiment, we ran four evolutionary processes, and the best result is presented. The evolution process shown in Figure 5.9 can be interpreted in the same way as in Figure 5.5. In this case, the first 30 epochs had more search space exploration; after that, the evolution process prioritized exploitation over exploration.

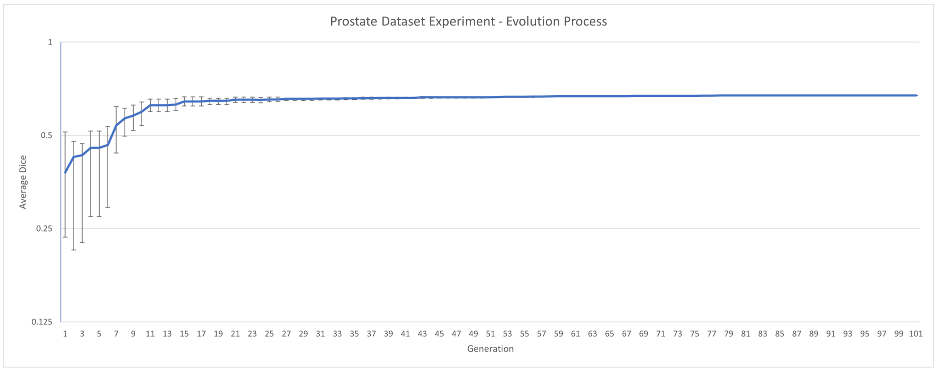


Figure 5.9: Classical individual fitness during process evolution for prostate dataset experiment

The evolutionary process converged pretty early in this experiment. The best architecture was found in generation 55, and after generation 77, the nine best classical individuals always presented the same evaluation score. In this experiment, we can conclude that, similarly to the first experiment, it was not necessarily 100 generations that were required to converge, and with a lower number of generations, it could lead to a better GPU days score. Table 5.4 presents the best architecture found.

Table 5.4: Best architecture found for prostate dataset experiment

Node	**Description**
den_n_3	DenseNet nonscaling 3×3
inc_d_5	InceptionNet downscaling 5×5
vgg_d_7	VGG downscaling 7×7
vgg_d_5	VGG downscaling 5×5
vgg_n_3	VGG nonscaling 3×3
vgg_n_3	IVGG nonscaling 3×3
vgg_u_5	VGG upscaling 5×5
vgg_u_7	VGG upscaling 7×7
inc_u_5	InceptionNet upscaling 5×5
den_n_3	DenseNet nonscaling 3×3

In this experiment, the best individual contains more non-scaling cells throughout the architecture, resulting in a lower parameter count. Additionally, the ResNet block block was not utilized, likely because the prostate dataset task is not particularly complex. During the evolution, the self-attention block was not considered a good choice for this problem and the explanation is the same as that provided Experiment 1 (Subsection 5.3.1).

Additionally, no MobileNet and EfficientNet were used in this experiment because lightweight blocks do not capture all the patterns needed for the task. Although the task is not complex enough to warrant the use of ResNet, it does not mean that the task is easy enough to be handled with only lightweight blocks. In terms of the identity block not being used, it means that a shallower architecture does not perform well in this dataset.

Changes in the function distribution of quantum nodes from one quantum individual are highlighted when comparing Figure 5.10, which contains the function distribution at the beginning of the evolutionary process, with Figure 5.11, which contains the function distribution of quantum nodes from one quantum individual at the end of the evolutionary process. In Figure 5.11 it can be seen that VGG, InceptionNet and DenseNet blocks have the highest probability of all functions and, nevertheless, the functions from this families of blocks appear in the best architecture found.

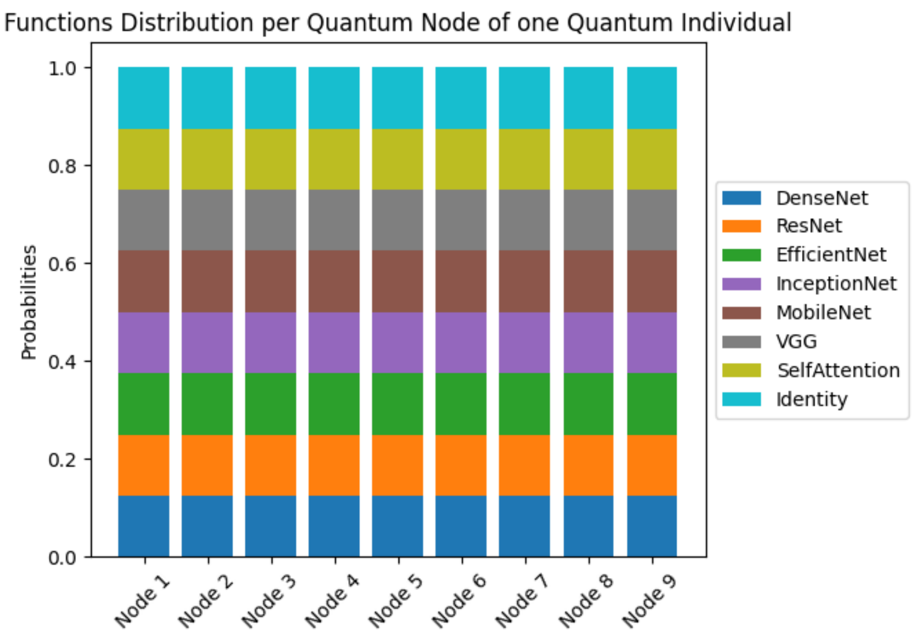


Figure 5.10: Functions distributions of a quantum individual at the start of the evolutionary process for the prostate dataset experiment

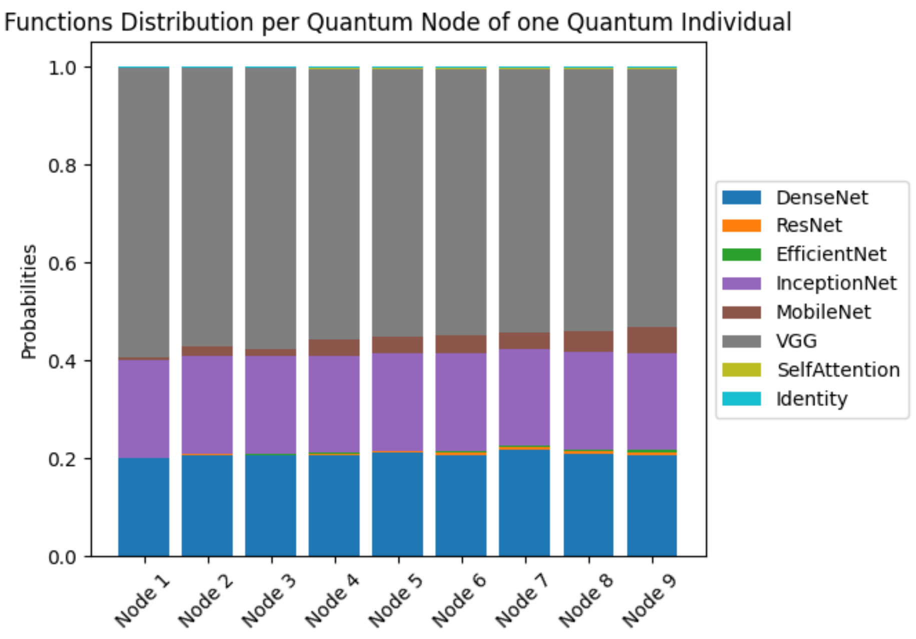


Figure 5.11: Functions distributions of a quantum individual at the end of the evolutionary process for the prostate dataset experiment

After identifying the optimal architecture during the evolution process, the model is retrained using all dataset images for 100 epochs with 100 initializations. Again, the other training parameters presented in Table 5.1 remained unchanged for the retraining process.

Figure 5.12 shows the output of the best network for three images sampled from the test set. The network output is close to the real labels; however, it is evident that in the first sample, the mask format was not precise. Additionally, for the last sample, it is also possible to notice that the orange mask (TZ) is oversized.

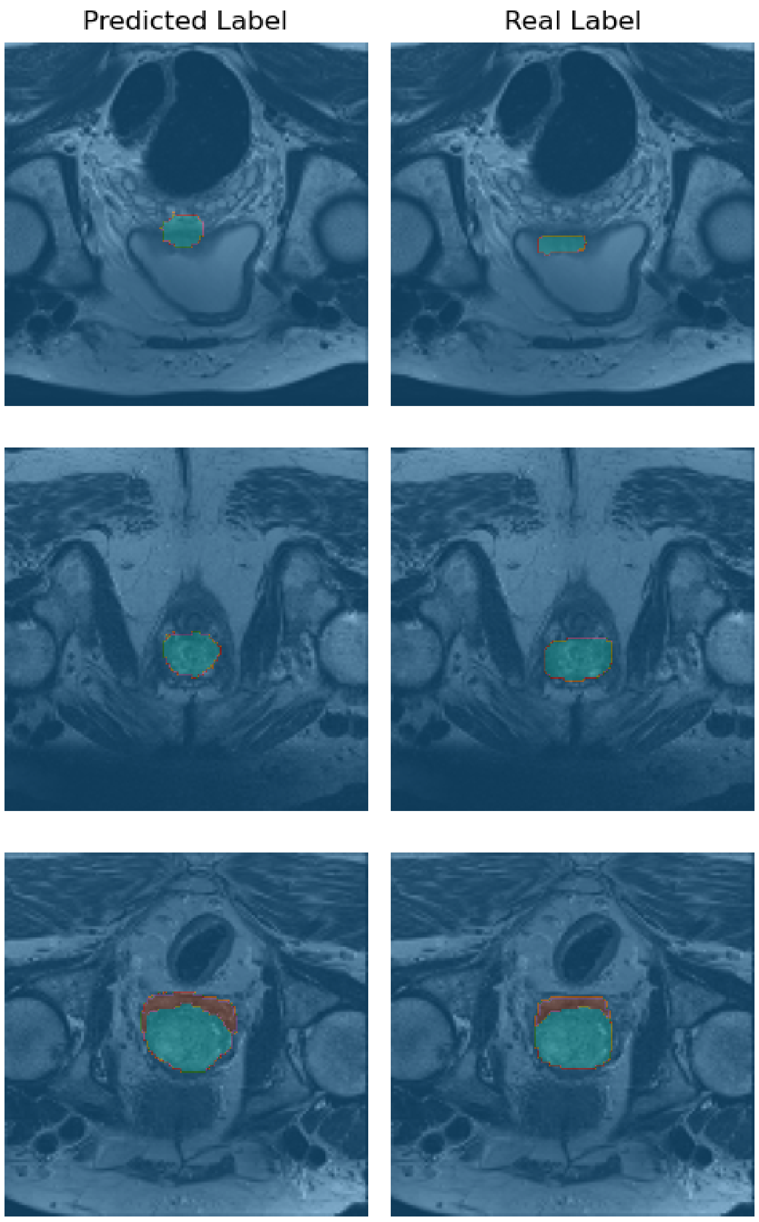


Figure 5.12: Three sampled images from prostate dataset with real and predicted labels. The orange mask represents the prostate peripheral zone (PZ) and the green mask represents the transition zone (TZ).

Table 5.5 compares the best result of SegQSNAS in the test set with other models.

Table 5.5: Prostate Dataset Experiments Results

Experiment	**Avg. DSC**	**PZ DSC**	**TZ DSC**	**#Params**	**GPU Days**
Random Search	0.7575 ± 0.002	0.5972	0.8270	1.4×10^6	0.09
SegQSNAS	0.7924 ± 0.004	0.6225	0.8403	0.5×10^6	1.08
SegQNAS_2[91]	0.6682 ± 0.066	NA	NA	5.7×10^6	1.74
SegQNAS_4[91]	0.6821 ± 0.067	NA	NA	16.4×10^6	1.83
SwinUNETR[104]	0.8240	0.7565	0.8915	62.0×10^6	NA

SegQNAS_2 and SegQNAS_4 have the same characteristics presented in Experiment 1 (Subsection 5.3.1). The SwinUNETR model is a modification of the UNETR model that was presented in Subsection 5.3.1.

Although showing a maximum gain of 0.04 in DSC compared to SegQNAS and SEGQSNAS (Table 5.5), SwinUNETR achieved this improvement at a significant computational cost, with approximately 124 times more parameters.

When SegQSNAS and SegQNAS were compared, SegQSNAS achieved a higher DSC score with fewer parameters and lower GPU days. The SegQSNAS DSC score is 0.11 higher than that of the best experiment in SegQNAS, with 32 times fewer parameters.

This experiment has two labels in addition to the background label, and due to this characteristic, SegQNAS overestimated the DSC score as presented in Chapter 4. Considering this issue, the improvement from SegQNAS work is even higher, as even with the overestimated DSC score, SegQSNAS improved the DSC score by 0.11.

This experiment validates that SegQSNAS can compete with state-of-the-art models, as evidenced by the close DSC score. In addition, due to the reduced number of network parameters, SegQSNAS can run with significantly fewer computer resources than SwinUNETR, making it a more suitable solution for cases with limited computing resources. As presented in Experiment 1 (Subsection 5.3.1), the SegQSNAS also surpass the baseline that used random search.

5.3.3 Experiment 3 - Liver Dataset

The liver dataset contains 131 contrast-enhanced CT scans with annotations that were used to generate the dataset for this experiment.

In this experiment, more resources were available, allowing the use of two of the three GPUs on the machine.

The evolution parameters applied to this experiment are presented in Table 5.6. In this case, the dataset size for the evolution process was reduced to improve the computational burden of training with the whole dataset. For the first two experiments, the entire dataset was used in the evolution process to match the parameters used by SegQNAS for a more accurate comparison. Since SegQNAS only evaluated the spleen and prostate datasets, the parameters could be adjusted for this experiment.

Table 5.6: SegQSNAS Evolution parameters for liver dataset experiment

Parameter	**Value**
Number of nodes	5
Number of quantum individual	9
Repetition	1
Update quantum rate	0.9
Update quantum gen	1.0
Probability mass function	Same probability for every function
Number of generations	100
Dataset	**limited dataset with 4000 images**
Image Resolution	128x128
Batch Size	32
Training Epochs	30
Evaluation Epochs	last 6

Due to the stochastic nature of this experiment, we ran two evolutionary processes and the best result is presented. The evolution process shown in Figure 5.13 can also be interpreted in the same way as explained in Figure 5.5.

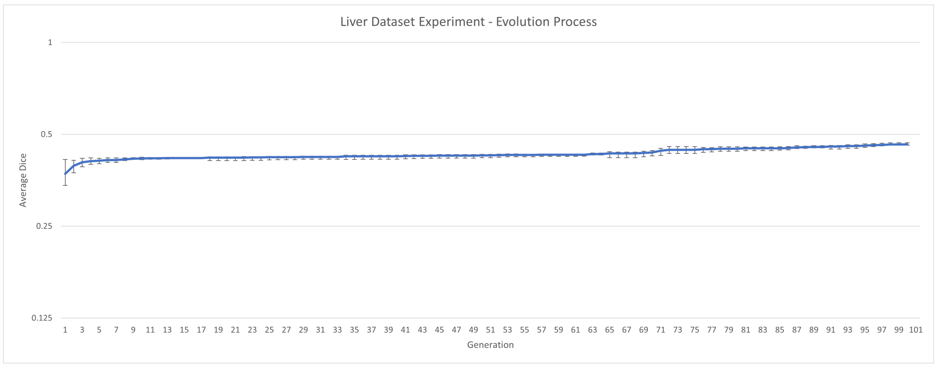


Figure 5.13: Classical individual fitness during process evolution for liver dataset experiment

The evolutionary process converged relatively early in this experiment, and the optimal architecture was identified at generation 90. It demonstrates that the number of generations set was sufficient; however, with a lower number of generations, it would not have converged to the solution found. The best architecture found is presented in Table 5.7.

Table 5.7: Best architecture found for liver dataset experiment

Node	**Description**
den_n_5	DenseNet nonscaling 5×5
vgg_d_3	VGG downscaling 3×3
eff_d	EfficientNet downscaling
vgg_d_7	VGG downscaling 7×7
eff_n	EfficientNet nonscaling
eff_n	EfficientNet nonscaling
vgg_u_7	VGG downscaling 7×7
eff_u	EfficientNet upscaling
vgg_u_3	VGG upscaling 3×3
den_n_5	DenseNet nonscaling 5×5

The best individual has no self-attention and ResNet, and the conclusion is similar to the one presented in Experiment 2 (Subsection 5.3.2), indicating that the task was not complex enough to require these types of blocks. The InceptionNet block was not part of the best architecture because, for this dataset, creating feature maps at different scales was not necessary. In addition to it, the identity block was also not used either, which means that it does not need a shallower architecture to achieve good results in this dataset.

Changes in the function distribution of quantum nodes from one quantum individual can be observed by comparing Figure 5.14, which contains the initial distribution, with Figure 5.15, which represents the final distribution. In this case, it is possible to notice that the probability of Identity, MobileNet and SelfAttention blocks are the smallest in all nodes and converges to the best architecture found that has none of those blocks selected.

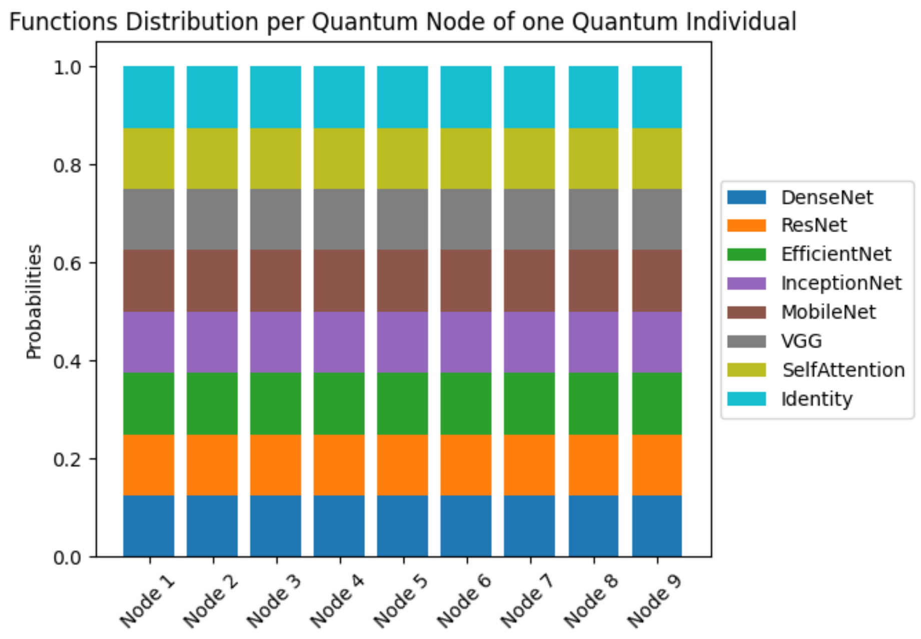


Figure 5.14: Functions distributions of a quantum individual at the start of the evolutionary process for the liver dataset experiment

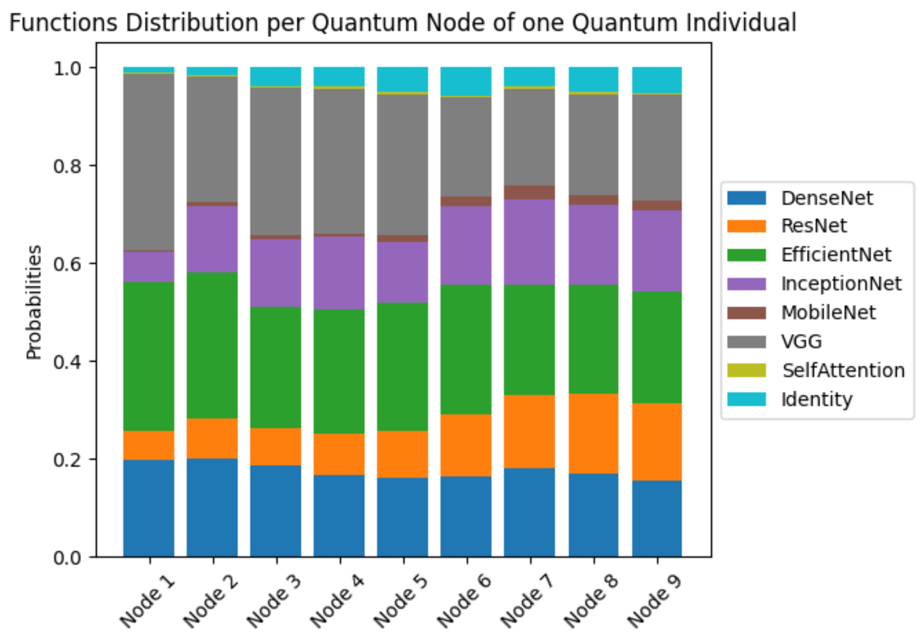


Figure 5.15: Functions distributions of a quantum individual at the end of the evolutionary process for the liver dataset experiment

After finding the best architecture during the evolution process, the model is retrained using all dataset images for 200 epochs with 10 initializations.

Figure 5.16 shows, as in the other experiments, good results, although they are not as precise as the real labels.

In Table 5.8 the comparison of SegQSNAS with other models is presented, providing a comprehensive overview of SegQSNAS capabilities.

Table 5.8: Liver Dataset Experiments Results

Experiment	**Avg. DSC**	**Liver DSC**	**Cancer DSC**	**#Params**	**GPU Days**
Random Search	0.5014 ± 0.012	0.7512	0.2945	0.11×10^6	0.13
SegQSNAS	0.7576 ± 0.002	0.9011	0.6141	10.5×10^6	5.00
SwinUNETR[104]	0.8552	0.9535	0.7568	62.0×10^6	NA
UNetFormer+[105]	0.7256	0.9479	0.5032	24.4×10^6	NA
UNetFormer[105]	0.7689	0.9573	0.5805	59.0×10^6	NA

The SwinUNETR was already presented in Experiment 2 (Subsection 5.3.2). The UNETFormer combines the encoder-decoder structure of U-Net

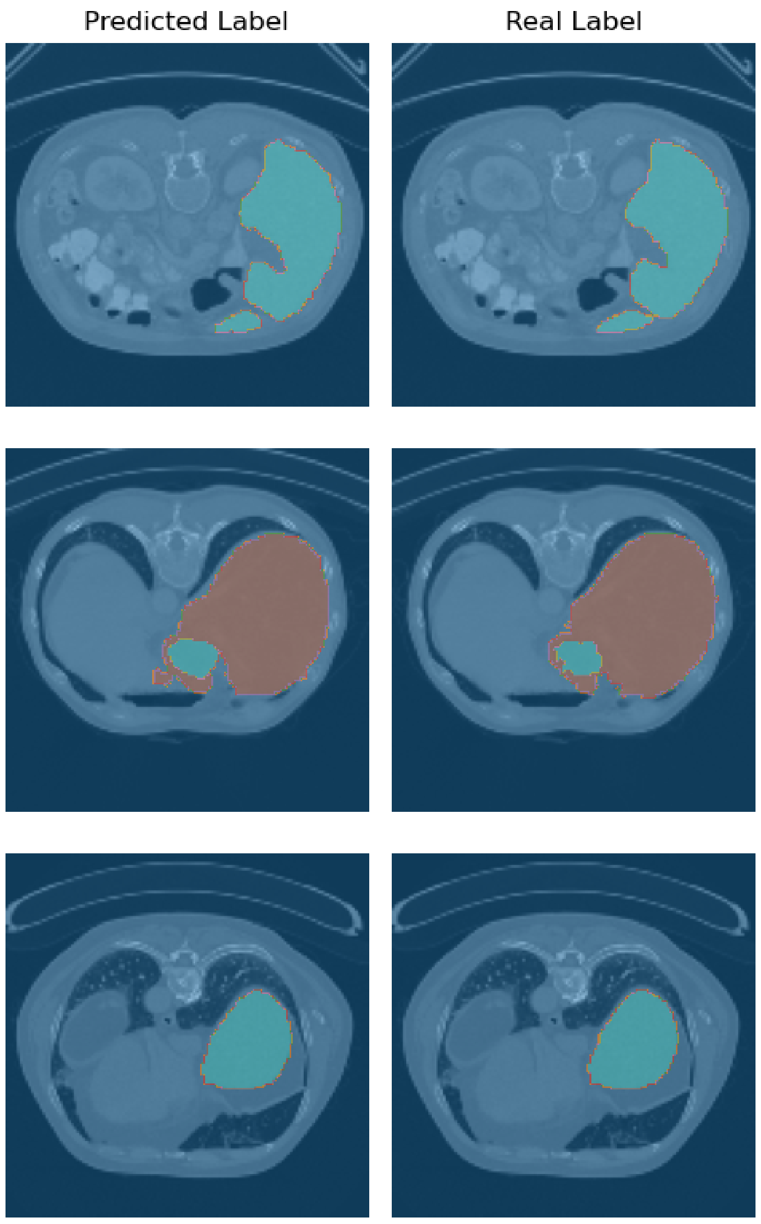


Figure 5.16: Three sampled images from liver dataset with real and predicted labels. The orange mask represents the liver and the green mask represents the cancer.

with the self-attention mechanisms characteristic of Vision Transformers. The UNETFormer+ is a modified version of UNETFormer that incorporates additional enhancements, such as refinement of the attention mechanism and improvement of pre-training strategies.

As presented in Table 5.8, SwinUNETR outperformed all models, archiving almost 0.10 more in DSC score than SegQSNAS. When comparing each isolated label, the SwinUNETR also outperformed SegQSNAS, leading to the conclusion that, for this dataset, a deeper network is needed. However, due to computer resource constraints, the maximum nodes used in this experiment were 5, which generated an architecture with a maximum of 10 nodes.

On the other hand, SegQSNAS is a better choice compared to the UNETFormer and UNETFormer+ models. SegQSNAS outperformed UNETFormer+ and was comparable to UNETFormer in terms of the DSC score. However, when comparing the number of parameters, SegQSNAS has fewer parameters than UNETFormer and UNETFormer+, and because of that, SegQSNAS could be a better solution in the cases of limited computing resources.

This experiment demonstrates that for this dataset, a deeper network is necessary because it can extract all the necessary features for the segmentation task to achieve greater accuracy. However, due to limited computer resources, the SegQSNAS experiment was unable to search for a deeper architecture.

Although hardware restrictions, it is possible to notice that compared to the Random Search, SegQSNAS continue to surpass the result and the difference become higher as the complexity of the problem increase.

5.3.4 Experiment 4 - Brain Tumor Dataset

The brain tumor dataset comprises 484 multiparametric MRI scans with annotations, which were used to generate the dataset for this experiment.

Similarly to Experiment 3 (Subsection 5.3.3), it was also possible to use two GPUs for this experiment.

The evolution parameters applied to this experiment are presented in Table 5.9. In this case, the dataset size was reduced for the evolution process, in the same way as in Experiment 3 (Subsection 5.3.3). In addition to this change, for this experiment the update quantum rate was reduced to do more exploration than exploitation during the evolution process.

Table 5.9: SegQSNAS Evolution parameters for brain tumor dataset experiment

Parameter	**Value**
Number of nodes	5
Number of quantum individual	9
Repetition	1
Update quantum rate	**0.5**
Update quantum gen	1.0
Probability mass function	Same probability for every function
Number of generations	100
Dataset	limited dataset with 4000 images
Image Resolution	128x128
Batch Size	32
Training Epochs	30
Evaluation Epochs	last 6

Due to the stochastic nature of this experiment, we ran two evolutionary processes, and the best result is presented. The evolution process is shown in Figure 5.17 and the plot interpretation is the same as in Figure 5.5.

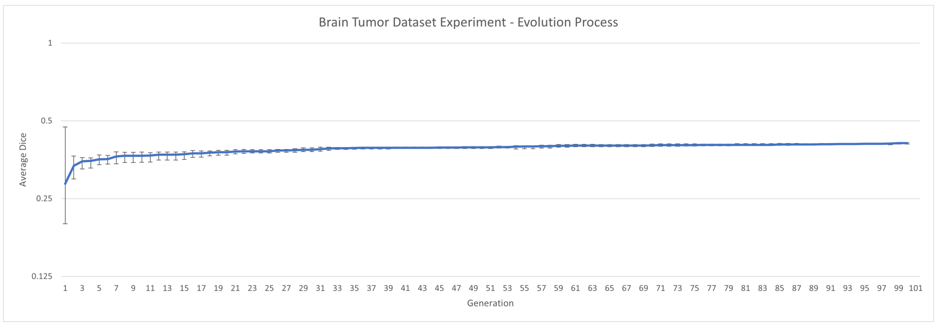


Figure 5.17: Classical individual fitness during process evolution for brain tumor dataset experiment

The evolutionary process converged relatively early in this experiment, and the optimal architecture was identified at generation 97. Although it may seem that more generations are needed to converge, this is not entirely accurate because there were minimal improvements in the best individual since generation 50, which suggests that the search had effectively converged. The best architecture found is presented in Table 5.10.

Table 5.10: Best architecture found for brain tumor dataset experiment

Node	**Description**
res_n_3	ResNet nonscaling 3×3
mobile_d_v1	MobileNet downscaling
den_n_3	DenseNet nonscaling 3×3
den_n_7	DenseNet nonscaling 7×7
mobile_d_v1	MobileNet downscaling
mobile_u_v1	MobileNet upscaling
den_n_7	DenseNet nonscaling 7×7
den_n_3	DenseNet nonscaling 3×3
mobile_u_v1	MobileNet upscaling
res_n_3	ResNet nonscaling 3×3

The best individual found in this experiment has the same explanation presented in the others experiments for the case of not using self-attention, ResNet, InceptionNet and identity block.

For this experiment, Figure 5.18, with the initial function distribution, and Figure 5.19, with the final distribution, were provided for a better visual interpretation of its differences. It is important to note that the probability of SelfAttention blocks is the smallest in all nodes and it is also possible to see that the best architecture found has no SelfAttention Block.

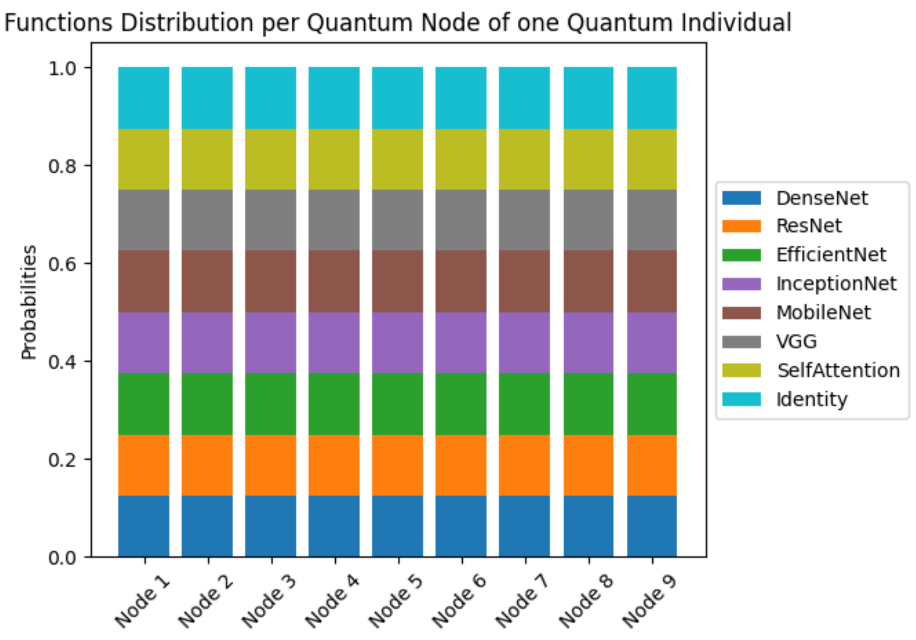


Figure 5.18: Functions distributions of a quantum individual at the start of the evolutionary process for the brain tumor dataset experiment

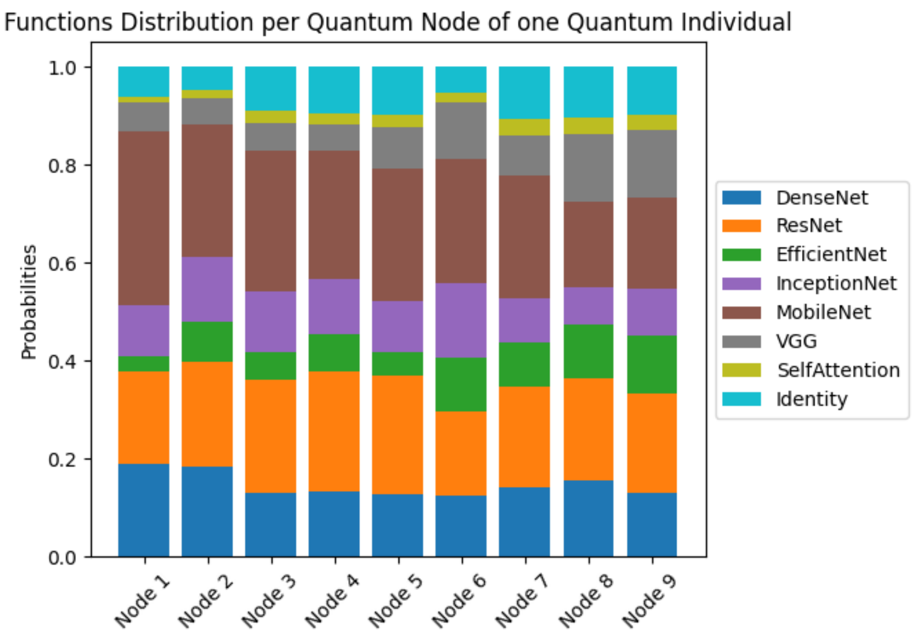


Figure 5.19: Functions distributions of a quantum individual at the end of the evolutionary process for the brain tumor dataset experiment

After finding the best architecture during the evolution process, the model is retrained using all dataset images for 200 epochs with five initializations.

Figure 5.20 shows the degradation of the network output as the number of output labels increases. This occurs because the evolutionary process was unable to find a network that performed well across the three labels of the brain tumor dataset images.

Table 5.11 shows a detailed comparison of SegQSNAS performance with other relevant models.

Table 5.11: Brain Tumor Dataset Experiments Results

Experiment	**Avg. DSC**	**E DSC**	**NET DSC**	**ET DSC**	**#Params**	**GPU Days**
Random Search	0.4787 ± 0.002	0.6033	0.4296	0.4143	0.3×10^6	0.42
SegQSNAS	0.5673 ± 0.002	0.6772	0.4627	0.5621	0.5×10^6	10.58
UNETR[102]	0.7110	0.7890	0.5850	0.7610	92.6×10^6	NA
SwinUNETR[104]	0.6435	0.7002	0.5252	0.7051	62.0×10^6	NA

The UNETR model and the SwinUNETR model were already introduced in Experiment 1 (Subsection 5.3.1) and Experiment 2 (Subsection 5.3.2), respectively.

As presented in Table 5.8, UNETR outperformed all models. When comparing each isolated label, the greater difference between UNETR and SegQSNAS is observed on the ET label, with a difference of almost 0.2 in the DSC score. Compared to SwinUNETR, the same phenomenon occurs, with a higher difference in the ET DSC score, although in this comparison, the E DSC and NET DSC values were relatively close.

In this experiment, two possible conclusions for why SegQSNAS had a higher difference in the ET DSC score. The first is that the SegQSNAS best architecture has only half a million parameters, and the ET pattern has a much more complex pattern than these numbers of parameters can detect. The other explanation is that the search strategy using the average DSC could lead to a non-optimal architecture when the number of labels increases.

Despite the non-optimal solution provided by SegQSNAS in this experiment, SegQSNAS continue to provide a better architecture when compared with Random Search results.

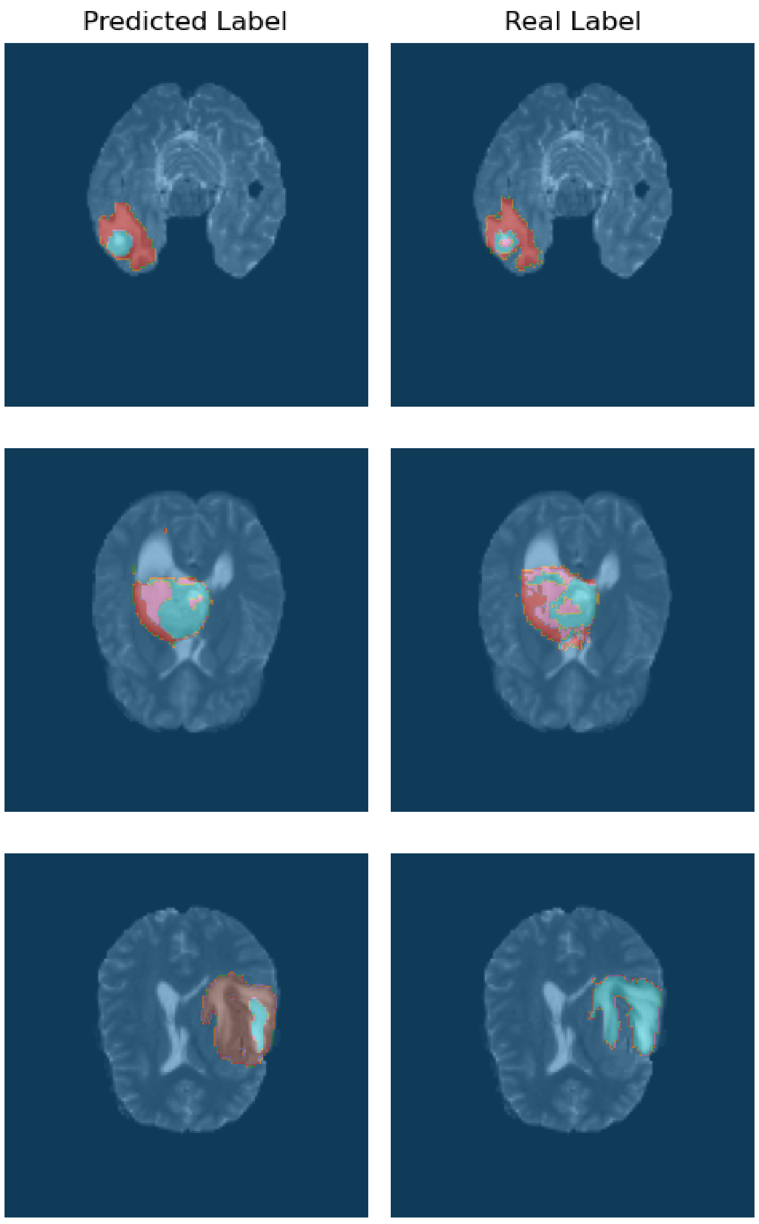


Figure 5.20: Three sampled images from brain tumor dataset with real and predicted labels. The orange mask represents the edema, the green mask represents the non-enhanced tumor and the pink mask represents the enhanced tumor.

5.4 Discussion

In this section, the experiments are discussed in general, along with the relationships between each result. From all experiments, SegQSNAS presented the best result with Experiment 2 in the prostate dataset (Subsection 5.3.2), which is a multi-class problem. In Experiment 3 (Subsection 5.3.3), although the DSC score was not the highest, SegQSNAS provided the best trade-off between the DSC score and the parameter count, considering that computational resources are important.

Considering the first and fourth experiments, SegQSNAS did not perform well. In the first experiment (Subsection 5.3.1), the best SegQSNAS result was pretty close to the one presented by the SegQNAS; however, due to the design of the SegQSNAS, the best result found by the SegQNAS work was not part of the search space. In the fourth experiment (Subsection 5.3.4), it was possible to observe that as the number of labels increases, the best network found by SegQSNAS starts to degrade, and in this case, it occurred with the ET DSC score, showing the highest score difference.

Although the experiments show that the proposed model performance is not higher than the state-of-the-art models, it is a favorable trade-off given the architectures' smaller parameter counts. However, the primary goal was higher performance, yet the search algorithm consistently generated custom networks with fewer parameters, despite this not being part of the fitness evaluation.

Considering the results of the experiments, it was possible to notice that none of the experiments, even with the best architecture, incorporated the identity block, indicating that the maximum node in the architecture could be increased.It also shows that due to hardware restrictions, the self-attention block was not correctly evaluated due to out of memory errors in such architectures. The functions that never appeared in the best architecture of all experiments are listed below:

- vgg_n_5: VGG nonscaling 5×5
- vgg_n_7: VGG nonscaling 7×7
- res_d_3: ResNet downscaling 3×3
- res_d_5: ResNet downscaling 5×5
- res_n_5: ResNet nonscaling 5×5
- res_n_7: ResNet nonscaling 7×7
- den_d_5: DenseNet downscaling 5×5
- den_d_7: DenseNet downscaling 7×7

- inc_d_3: InceptionNet downscaling 3×3
- inc_n_3: InceptionNet nonscaling 3×3
- inc_n_5: InceptionNet nonscaling 5×5
- inc_n_7: InceptionNet nonscaling 7×7
- $mobile\_n\_v1$: MobileNet nonscaling

In summary, the prostate dataset experiment demonstrated that SegQS-NAS performed well, yielding the best results in terms of DSC score, parameter count, and GPU days. It also archive of good results on the liver dataset experiment due to the best trade-off considering that it is a limited computing resource scenario. However, some experiments show some limitations in the search strategy or computational resources available. In addition, in any experiment, the self-attention block was not selected in the best architecture found, and this is indicative that the predefined maximum number of nodes or the complexity of the tested problems might not have been high enough for this operation to be selected during the evolution process.

6
Conclusions

Based on the foundations of SegQNAS, the proposed method introduced key modifications aimed at improving search efficiency, model compactness, and performance on medical imaging datasets. Throughout this work, we explored architectural constraints, expanded the functional diversity of the search space, and refined the evolutionary strategy to achieve more robust and efficient models.

The primary goal of this work was to reduce the search space by introducing a design constraint that limits exploration to symmetrical U-Net-like networks.This dissertation presented SegQSNAS, a quantum-inspired neural architecture search framework applied to semantic segmentation tasks. Additionally, this symmetrical design inherently eliminates the need for the feasibility check required by SegQNAS. Every downsampling operation in one half of the network is guaranteed to have a corresponding upsampling operation in the other half.

This goal was achieved because the original SegQNAS model often generates infeasible individuals. Moreover, the total number of nodes defines the network's size. In contrast, SegQSNAS limits the search to architectures with half the number of nodes, ensuring feasibility. When comparing SegQSNAS and SegQNAS results in the spleen and prostate experiments, we observed that SegQSNAS significantly improved the DSC score, reduced the number of parameters, and decreased GPU usage in the prostate experiment. Additionally, in the spleen dataset, it produced an architecture with comparable DSC and parameter count.

The secondary goal was to add new functions to the search space, such as self-attention, MobileNet, and EfficientNet. The self-attention function was intended to enhance the composition of the filters during convolution. However, it was not selected in the best architecture of any experiment, possibly indicating that more complex architectures (with a higher number of nodes) are needed for this feature to be effective. This means that the problem was not complex enough to replace any other function with self-attention to improve the result.

However, MobileNet and EfficientNet were selected as the best architecture in all experiments except the prostate dataset experiment. These blocks enabled the SegQSNAS evolutionary process to find lightweight customized networks, reducing the number of parameters and GPU computation time (GPU days).

The third goal was to improve the evolutionary process by reducing noise and better exploiting the best solutions. To achieve this goal, a crossover over the classical categorical individual was implemented using the two-point crossover operation. The evolutionary process showed that each time a better-performing solution was discovered, the diversity among the top ten classical individuals increased. Thanks to the crossover operator, the algorithm was able to exploit this diversity more effectively, resulting in faster convergence.

The last goal was to correct the metrics and loss implementation in SegQNAS regarding multi-class problems. The DSC function used in SegQNAS overestimates the multi-class problem because the numerator in the DSC formula does not fully account for the true overlap between multiple labels, while the denominator is inflated due to the sum over all labels. The correction was based on the usage of the average of the DSC calculated for each label. In the prostate dataset experiment, the SegQSNAS implementation led the evolution process to a better solution even compared to an overestimated DSC score.

In future work, we will extend our experiments to datasets with a larger number of labels, investigating the limitations of the method in high-complexity scenarios. We also intend to scale our experiments to a higher maximum number of nodes, allowing the discovery of more complex network architectures. Furthermore, we will extend the work to implement a population diversity control module and dynamic parameters to be applied to the evolution process, aiming to optimize the balance between exploration and exploitation. This will include testing different search strategies, such as a hierarchical approach with an initial topological search and a subsequent hyperparameter search, exploring different crossover operators, and testing various mutation operations.

Bibliography

[1] YOUNG, T.; HAZARIKA, D.; PORIA, S. ; CAMBRIA, E.. **Recent trends in deep learning based natural language processing [review article]**. IEEE Computational Intelligence Magazine, 13(3):55–75, Aug. 2018.

[2] VARGAS, R.; MOSAVI, A. ; RUIZ, R.. **Deep learning: A review**. Oct. 2018.

[3] HWANG, J.-J.. **“impact of deep learning and applications in biomedicine”**. Biomedical Journal of Scientific amp; Technical Research, 42(2), Feb. 2022.

[4] FAUST, O.; HAGIWARA, Y.; HONG, T. J.; LIH, O. S. ; ACHARYA, U. R.. **Deep learning for healthcare applications based on physiological signals: A review**. Computer Methods and Programs in Biomedicine, 161:1–13, July 2018.

[5] YU, Y.; LI, M.; LIU, L.; LI, Y. ; WANG, J.. **Clinical big data and deep learning: Applications, challenges, and future outlooks**. Big Data Mining and Analytics, 2(4):288–305, Dec. 2019.

[6] CHETAN, M. R.; GLEESON, F. V.. **Radiomics in predicting treatment response in non-small-cell lung cancer: current status, challenges and future perspectives**. European Radiology, 31(2):1049–1058, Aug. 2020.

[7] GOODFELLOW, I.; BENGIO, Y. ; COURVILLE, A.. **Deep Learning**. Adaptive computation and machine learning. MIT Press, 2016.

[8] DEBELEE, T. G.; KEBEDE, S. R.; SCHWENKER, F. ; SHEWAREGA, Z. M.. **Deep learning in selected cancers’ image analysis—a survey**. Journal of Imaging, 6(11):121, Nov. 2020.

[9] MIJWIL, M. M.; MUTAR, D. S.; MAHMOOD, E. S.; GÖK, M.; UZUN, S. ; DOSHI, R.. **Deep learning applications and their worth: A short review**. Asian Journal of Applied Sciences, 10(5), Nov. 2022.

[10] SWAMINATHA RAO, L. P.; JAGANATHAN, S.. **Adaptive bayesian contextual hyperband: A novel hyperparameter optimization approach**. IAES International Journal of Artificial Intelligence (IJ-AI), 13(1):775, Mar. 2024.

[11] GAY, S. S.; KISLING, K. D.; ANDERSON, B. M.; ZHANG, L.; RHEE, D. J.; NGUYEN, C.; NETHERTON, T.; YANG, J.; BROCK, K.; JHINGRAN, A.; SIMONDS, H.; KLOPP, A.; BEADLE, B. M.; COURT, L. E. ; CARDENAS, C. E.. **Identifying the optimal deep learning architecture and parameters for automatic beam aperture definition in 3d radiotherapy**. Journal of Applied Clinical Medical Physics, 24(12), Sept. 2023.

[12] LAKHMIRI, D.; DIGABEL, S. L. ; TRIBES, C.. **Hypernomad: Hyperparameter optimization of deep neural networks using mesh adaptive direct search**, 2019.

[13] LIU, C.; CHEN, L.-C.; SCHROFF, F.; ADAM, H.; HUA, W.; YUILLE, A. L. ; FEI-FEI, L.. **Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation**. In: 2019 IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR). IEEE, June 2019.

[14] LIU, Y.; SUN, Y.; XUE, B.; ZHANG, M.; YEN, G. G. ; TAN, K. C.. **A survey on evolutionary neural architecture search**. IEEE transactions on neural networks and learning systems, 2021.

[15] GONG, X.; CHANG, S.; JIANG, Y. ; WANG, Z.. **Autogan: Neural architecture search for generative adversarial networks**. In: 2019 IEEE/CVF INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV). IEEE, Oct. 2019.

[16] MAZIARZ, K.; TAN, M.; KHORLIN, A.; GEORGIEV, ; GESMUNDO, A.. **Evolutionary-neural hybrid agents for architecture search**. 2018.

[17] SINHA, N.; CHEN, K.. **Neural architecture search using progressive evolution**. 2022.

[18] XIANG, S.; HE, Y.. **A quantum-inspired evolutionary algorithm with elite group guided**. Applied Mechanics and Materials, 738-739:323–333, 2015.

[19] AHMAD, R.; AWAIS, M.; KAUSAR, N.; TARIQ, U.; CHA, J. ; BAILI, J.. **Leukocytes classification for leukemia detection using quantum inspired deep feature selection**. Cancers, 15:2507, 2023.

[20] SZWARCMAN, D.; CIVITARESE, D. ; VELLASCO, M.. **Quantum-inspired evolutionary algorithm applied to neural architecture search**. Applied Soft Computing, 120:108674, 2022.

[21] ANTONELLI, M.; REINKE, A.; BAKAS, S.; FARAHANI, K.; KOPP-SCHNEIDER, A.; LANDMAN, B. A.; LITJENS, G.; MENZE, B.; RONNEBERGER, O.; SUMMERS, R. M. ; OTHERS. **The medical segmentation decathlon**. Nature communications, 13(1):1–13, 2022.

[22] MINAEE, S.; BOYKOV, Y. Y.; PORIKLI, F.; PLAZA, A. J.; KEHTARNAVAZ, N. ; TERZOPOULOS, D.. **Image segmentation using deep learning: A survey**. IEEE transactions on pattern analysis and machine intelligence, 2021.

[24] AAKANKSHA; SETH, A. ; SHARMA, S.. **Semantic segmentation: A systematic analysis from state-of-the-art techniques to advance deep networks**. Journal of Information Technology Research, 15(1):1–28, Mar. 2023.

[25] WANG, W.; FU, Y.; PAN, Z.; LI, X. ; ZHUANG, Y.. **Real-time driving scene semantic segmentation**. IEEE Access, 8:36776–36788, 2020.

[26] CHEN, J.; YU, J.; WANG, Y. ; HE, X.. **Elanet: an efficiently lightweight asymmetrical network for real-time semantic segmentation**. Journal of Electronic Imaging, 33(01), Jan. 2024.

[27] ZHAO, H.; SHI, J.; QI, X.; WANG, X. ; JIA, J.. **Pyramid scene parsing network**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), July 2017.

[28] FENG, D.; HAASE-SCHÜTZ, C.; ROSENBAUM, L.; HERTLEIN, H.; GLAESER, C.; TIMM, F.; WIESBECK, W. ; DIETMAYER, K.. **Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges**. IEEE Transactions on Intelligent Transportation Systems, 22(3):1341–1360, 2020.

[29] MA, L.; LIU, Y.; ZHANG, X.; YE, Y.; YIN, G. ; JOHNSON, B. A.. **Deep learning in remote sensing applications: A meta-analysis and**

review. ISPRS journal of photogrammetry and remote sensing, 152:166–177, 2019.

[30] ABDULLAH, F.; JALAL, A.. **Semantic segmentation based crowd tracking and anomaly detection via neuro-fuzzy classifier in smart surveillance system**. Arabian Journal for Science and Engineering, p. 1–18, 2022.

[31] SUGANYADEVI, S.; SEETHALAKSHMI, V. ; BALASAMY, K.. **A review on deep learning in medical image analysis**. International Journal of Multimedia Information Retrieval, 11(1):19–38, 2022.

[32] BRIOUYA, H.; BRIOUYA, A. ; CHOUKRI, A.. **Exploration of image and 3d data segmentation methods: an exhaustive survey**. TELKOMNIKA (Telecommunication Computing Electronics and Control), 22(2):413, Apr. 2024.

[33] ULKU, I.; AKAGÜNDÜZ, E.. **A survey on deep learning-based architectures for semantic segmentation on 2d images**. Applied Artificial Intelligence, 36(1), Feb. 2022.

[34] GENG, Q.; ZHOU, Z. ; CAO, X.. **Survey of recent progress in semantic image segmentation with cnns**. Science China Information Sciences, 61(5), Nov. 2017.

[35] OTSU, N.. **A threshold selection method from gray-level histograms**. IEEE transactions on systems, man, and cybernetics, 9(1):62–66, 1979.

[36] NOCK, R.; NIELSEN, F.. **Statistical region merging**. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(11):1452–1458, 2004.

[37] DHANACHANDRA, N.; MANGLEM, K. ; CHANU, Y. J.. **Image segmentation using k-means clustering algorithm and subtractive clustering algorithm**. Procedia Computer Science, 54:764–771, 2015.

[38] NAJMAN, L.; SCHMITT, M.. **Watershed of a continuous function**. Signal Processing, 38(1):99–112, 1994. Mathematical Morphology and its Applications to Signal Processing.

[39] MICHAEL KASS, A. W.; TERZOPOULOS, D.. **Snakes: Active contour models**. International Journal of Computer Vision, 1:321–331, 1998.

[40] BOYKOV, Y.; VEKSLER, O. ; ZABIH, R.. **Fast approximate energy minimization via graph cuts.** IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(11):1222–1239, 2001.

[41] PLATH, N.; TOUSSAINT, M. ; NAKAJIMA, S.. **Multi-class image segmentation using conditional random fields and global classification.** In: PROCEEDINGS OF THE 26TH ANNUAL INTERNATIONAL CONFERENCE ON MACHINE LEARNING, p. 817–824, 2009.

[42] HOCHBAUM, D. S.. **An efficient algorithm for image segmentation, markov random fields and related problems.** Journal of the ACM (JACM), 48(4):686–701, 2001.

[43] STARCK, J.-L.; ELAD, M. ; DONOHO, D. L.. **Image decomposition via the combination of sparse representations and a variational approach.** IEEE transactions on image processing, 14(10):1570–1582, 2005.

[44] MINAEE, S.; WANG, Y.. **An admm approach to masked signal decomposition using subspace representation.** IEEE Transactions on Image Processing, 28(7):3192–3204, 2019.

[45] GHOSH, S.; DAS, N.; DAS, I. ; MAULIK, U.. **Understanding deep learning techniques for image segmentation.** ACM Computing Surveys (CSUR), 52(4):1–35, 2019.

[46] LI, Y.. **The application of semantic segmentation on 2d images.** Highlights in Science, Engineering and Technology, 31:88–96, Feb. 2023.

[47] GUO, Y.; LIU, Y.; GEORGIOU, T. ; LEW, M. S.. **A review of semantic segmentation using deep neural networks.** International Journal of Multimedia Information Retrieval, 7(2):87–93, Nov. 2017.

[48] WANG, J.; ZENG, X.; SHAN, D.; ZHOU, Q. ; PENG, H.. **Image target recognition based on improved convolutional neural network.** Mathematical Problems in Engineering, 2022:1–11, 2022.

[49] ZHANG, H.; JOLFAEI, A. ; ALAZAB, M.. **A face emotion recognition method using convolutional neural network and image edge computing.** IEEE Access, 7:159081–159089, 2019.

[50] QIN, G.; QIN, G.. **Virtual reality video image classification based on texture features.** Complexity, 2021, 2021.

[51] KOFLER, C.; MUHR, R. ; SPÖCK, G.. **Classifying image stacks of specular silicon wafer back surface regions: performance comparison of cnns and svms.** Sensors, 19:2056, 2019.

[52] LECUN, Y.; BENGIO, Y.; HINTON, G. ; OTHERS. **Deep learning. nature, 521 (7553), 436-444.** Google Scholar Google Scholar Cross Ref Cross Ref, p. 25, 2015.

[53] LIU, W.; WEN, Y.; YU, Z. ; YANG, M.. **Large-margin softmax loss for convolutional neural networks.** 2016.

[54] LE, S. N.. **English handwriting recognition based on the convolutional neural network.** Applied and Computational Engineering, 5:146–151, 2023.

[55] WANG, J.; LIU, M.; ZENG, X. ; HUA, X.. **Spectral convolution feature-based spd matrix representation for signal detection using a deep neural network.** Entropy, 22:949, 2020.

[56] SOEKARTA, R.; ARAS, S. ; ASWAD, A. N.. **Hyperparameter optimization of cnn classifier for music genre classification.** Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi), 7:1205–1210, 2023.

[57] ZHOU, Z.; SIDDIQUEE, M. R.; TAJBAKHSH, N. ; LIANG, J.. **Unet++: a nested u-net architecture for medical image segmentation.** Lecture Notes in Computer Science, p. 3–11, 2018.

[58] CHEN, J.; LU, Y.; YU, Q.; LUO, X.; ADELI, E.; WANG, Y.; LÜ, L.; YUILLE, A. ; ZHOU, Y.. **Transunet: transformers make strong encoders for medical image segmentation.** 2021.

[59] RONNEBERGER, O.; FISCHER, P. ; BROX, T.. **U-net: Convolutional networks for biomedical image segmentation**, 2015.

[60] JHA, D.; RIEGLER, M.; JOHANSEN, D.; HALVORSEN, P. ; JOHANSEN, H. D.. **Doubleu-net: a deep convolutional neural network for medical image segmentation.** 2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS), p. 558–564, 2020.

[61] DHALLA, S.; MITTAL, A. ; GUPTA, S.. **Leukosegmenter: a double encoder-decoder based network for leukocyte segmentation from blood smear images.** 2021.

[62] `GALDRÁN, A.; CARNEIRO, G. ; BALLESTER, M. G.. **Double encoder-decoder networks for gastrointestinal polyp segmentation**. Pattern Recognition. ICPR International Workshops and Challenges, p. 293–307, 2021.

[63] BADRINARAYANAN, V.; CIPOLLA, R.. **Segnet: a deep convolutional encoder-decoder architecture for image segmentation**. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39:2481–2495, 2017.

[64] GAO, F.; HE, Y.; WANG, J.; MA, F.; YANG, E. ; HUSSAIN, A.. **Pixel-wise segmentation of sar imagery using encoder-decoder network and fully-connected crf**. Advances in Brain Inspired Cognitive Systems, p. 155–165, 2020.

[65] OKTAY, O.; SCHLEMPER, J.; FOLGOC, L. L.; LEE, M.; HEINRICH, M.; MISAWA, K.; MORI, K.; MCDONAGH, S.; HAMMERLA, N. Y.; KAINZ, B.; GLOCKER, B. ; RUECKERT, D.. **Attention u-net: learning where to look for the pancreas**. 2018.

[66] ELSKEN, T.; METZEN, J. H. ; HUTTER, F.. **Neural architecture search: A survey**. The Journal of Machine Learning Research, 20(1):1997–2017, 2019.

[67] WISTUBA, M.; RAWAT, A. ; PEDAPATI, T.. **A survey on neural architecture search**. arXiv preprint arXiv:1905.01392, 2019.

[68] LIU, C.; ZOPH, B.; NEUMANN, M.; SHLENS, J.; HUA, W.; LI, L.-J.; FEI-FEI, L.; YUILLE, A.; HUANG, J. ; MURPHY, K.. **Progressive neural architecture search**. In: PROCEEDINGS OF THE EUROPEAN CONFERENCE ON COMPUTER VISION (ECCV), p. 19–34, 2018.

[69] DE JONG, K.. **Evolutionary computation: a unified approach**. In: PROCEEDINGS OF THE 2016 ON GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE COMPANION, p. 185–199, 2016.

[70] MONTECINO, D. A.; PEREZ, C. A. ; BOWYER, K. W.. **Two-level genetic algorithm for evolving convolutional neural networks for pattern recognition**. IEEE Access, 9:126856–126872, 2021.

[71] WANG, L.; ZHAO, Y.; JINNAI, Y.; TIAN, Y. ; FONSECA, R.. **Neural architecture search using deep neural networks and monte carlo tree search**. Proceedings of the AAAI Conference on Artificial Intelligence, 34:9983–9991, 2020.

[72] DONG, X.; YANG, Y.. **Searching for a robust neural architecture in four gpu hours**. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), p. 1761–1770, 2019.

[73] JIANG, W.; ZHANG, X.; SHA, E. H.; YANG, L.; ZHUGE, Q.; SHI, Y. ; HU, J.. **Accuracy vs. efficiency: achieving both through fpga-implementation aware neural architecture search**. 2019.

[74] AWAD, N. H.; MALLIK, N. ; HUTTER, F.. **Differential evolution for neural architecture search**. 2020.

[75] ZHANG, H.; JIN, Y.; CHENG, R. ; HAO, K.. **Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance**. IEEE Transactions on Evolutionary Computation, 25:371–385, 2021.

[76] QU, X.; WANG, J. ; XIAO, J.. **Evolutionary algorithm enhanced neural architecture search for text-independent speaker verification**. 2020.

[77] HAN, X.; LI, C.; WANG, Z. ; LIU, G.. **Ndarts: a differentiable architecture search based on the neumann series**. Algorithms, 16:536, 2023.

[78] WHITE, C.; NEISWANGER, W. ; SAVANI, Y.. **Bananas: bayesian optimization with neural architectures for neural architecture search**. 2019.

[79] LIU, C.. **Neural architecture search in embedding space**. 2019.

[80] XU, Y.; WANG, Y.; HAN, K.; TANG, Y.; JUI, S.; XU, C. ; XU, C.. **Renas:relativistic evaluation of neural architecture search**. 2019.

[81] SYED, M.; SRINIVASAN, A. A.. **Generalized latency performance estimation for once-for-all neural architecture search**. 2021.

[82] LOPES, V.; ALIREZAZADEH, S. ; ALEXANDRE, L. A.. **Epe-nas: efficient performance estimation without training for neural architecture search**. Lecture Notes in Computer Science, p. 552–563, 2021.

[83] SZWARCMAN, D.; CIVITARESE, D. ; VELLASCO, M.. **Quantum-inspired neural architecture search**. In: 2019 INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN), p. 1–8. IEEE, July 2019.

[84] KERN, J.; HENWOOD, S.; MORDIDO, G.; DUPRAZ, E.; BEY, A. A. E.; SAVARIA, Y. ; LEDUC-PRIMEAU, F.. **Fast and accurate output error estimation for memristor-based deep neural networks**. IEEE Transactions on Signal Processing, 72:1205–1218, 2024.

[85] ŞATIR, E.; BASER, E.. **Optimization of interval type-2 fuzzy logic controller using real-coded quantum clonal selection algorithm**. Elektronika Ir Elektrotechnika, 28:4–14, 2022.

[86] MONTIEL, O.; RUBIO, Y.; OLVERA, C. ; RIVERA, A.. **Quantum-inspired acromyrmex evolutionary algorithm**. Scientific Reports, 9, 2019.

[87] ZAMANIA, M.; ALINAGHIANA, M. ; M, A. Y.. **A new improved quantum evolutionary algorithm with multiplicative update function**. International Journal on Computational Science Amp; Applications, 6:17–33, 2016.

[88] MORIYAMA, Y.; IIMURA, I.; OHNO, T. ; NAKAYAMA, S.. **An experimental study on optimization in permutation spaces by quantum-inspired evolutionary algorithm using quantum bit representation**. Journal of Signal Processing, 19:227–234, 2015.

[89] DA CRUZ, A. V. A.; VELLASCO, M. M. B. R. ; PACHECO, M. A. C.. **Quantum-Inspired Evolutionary Algorithm for Numerical Optimization**, p. 19–37. Springer Berlin Heidelberg, 2007.

[90] KHAN, I.. **Assessing different crossover operators for travelling salesman problem**. International Journal of Intelligent Systems and Applications, 7:19–25, 2015.

[91] CARLOS, G.; FIGUEIREDO, K.; HUSSAIN, A. ; VELLASCO, M.. **Segqnas: Quantum-inspired neural architecture search applied to medical image semantic segmentation**. In: 2023 INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN), p. 1–8, 2023.

[92] ZHANG, Y.; LI, J. ; JIANG, J.. **Indoor scene recognition method based on multi-scale feature attention mechanism**. 3rd International Conference on Artificial Intelligence, Automation, and High-Performance Computing (AIAHPC 2023), 2023.

[93] `LÓPEZ-CIFUENTES, A.; ESCUDERO-VIÑOLO, M.; BESCÓS, J. ; GARCÍA-MARTÍN, . **Semantic-aware scene recognition**. Pattern Recognition, 102:107256, 2020.

[94] CHEN, J.; ZHANG, D.; SUZAUDDOLA, M.; NANEHKARAN, Y. A. ; SUN, Y.. **Identification of plant disease images via a squeeze-and-excitation mobilenet model and twice transfer learning**. IET Image Processing, 15:1115–1127, 2020.

[95] GHANI, S. A. D.; INTAN, I. ; RIZAL, M.. **Mobilenet classifier for detecting chest x-ray images of covid-19 based on convolutional neural network**. ILKOM Jurnal Ilmiah, 15:488–497, 2023.

[96] WANG, W.; HU, Y.; ZOU, T.; LIU, H.; WANG, J. ; WANG, X.. **A new image classification approach via improved mobilenet models with local receptive field expansion in shallow layers**. Computational Intelligence and Neuroscience, 2020:1–10, 2020.

[97] QI, F.; YANG, C.; BAO, S. ; MA, S.. **Micro-expression recognition based on dcbam-efficientnet model**. Journal of Physics: Conference Series, 2504:012062, 2023.

[98] TAN, M.; LE, Q. V.. **Efficientnet: rethinking model scaling for convolutional neural networks**. 2019.

[99] WANG, J.; LIU, Q.; XIE, H.; YANG, Z. ; ZHOU, H.. **Boosted efficientnet: detection of lymph node metastases in breast cancer using convolutional neural networks**. Cancers, 13:661, 2021.

[100] MIAHI, E.; MIRROSHANDEL, S. A. ; NASR, A.. **Genetic neural architecture search for automatic assessment of human sperm images**. 2019.

[101] SAPRA, DOLLYAND PIMENTEL, A. D.. **Designing convolutional neural networks with constrained evolutionary piecemeal training**. Applied Intelligence, 52(15):17103–17117, Dec 2022.

[102] HATAMIZADEH, A.; YANG, D.; ROTH, H. R. ; XU, D.. **Unetr: transformers for 3d medical image segmentation**. 2021.

[104] TANG, Y.; YANG, D.; LI, W.; ROTH, H.; LANDMAN, B.; XU, D.; NATH, V. ; HATAMIZADEH, A.. **Self-supervised pre-training of swin transformers for 3d medical image analysis**, 2022.

[105] HATAMIZADEH, A.; XU, Z.; YANG, D.; LI, W.; ROTH, H. R. ; XU, D.. **Unetformer: a unified vision transformer model and pre-training framework for 3d medical image segmentation**. 2022.