

# 博士学位請求論文

## (2022)

論文題目

**Automated Deep Neural Networks with Gene  
Expression Programming of Cellular Encoding**  
Towards the Applications in Remote Sensing Image Understanding

学籍番号

18D5252

氏名

CLIFFORD BRONI-BEDIAKO

指導教員

PROFESSOR MASAYASU ATSUMI

**創 価 大 学 大 学 院  
工 学 研 究 科**



Discover your potential

# **Automated Deep Neural Networks with Gene Expression Programming of Cellular Encoding**

Towards the Applications in Remote Sensing Image Understanding

by

Clifford Broni-Bediako

A dissertation submitted  
in partial fulfilment of the requirements for the degree of  
**Doctor of Philosophy**

Information Systems Science  
Graduate School of Science and Engineering  
Soka University, Hachioji, Tokyo

February 2022



This PhD dissertation has been partially supported by the Makiguchi Foundation for Education (<https://www.daisakuikeda.org/>) under the Soka University Makiguchi Foundation for Education, International Student Scholarship.

© Clifford Broni-Bediako, 2022

*Dissertation submitted to the Graduate School of Science and Engineering, Soka University.*

All rights reserved. No part of this publication may be reproduced or transmitted, in any form or by any means, without permission.

Permission is herewith granted to Soka University to circulate and copy for non-commercial purposes, at its discretion, the request of individuals or institutions.



# **Automated Deep Neural Networks with Gene Expression Programming of Cellular Encoding**

Towards the Applications in Remote Sensing Image Understanding

by

Clifford Broni-Bediako

(18D5252)

## **Dissertation Review Committee**

---

Prof. Naoya Torii            Graduate School of Science and Engineering,  
Soka University, Tokyo.

Prof. Tatsuo Unemi        Graduate School of Science and Engineering,  
Soka University, Tokyo.

Prof. Masayasu Atsumi    Graduate School of Science and Engineering,  
Soka University, Tokyo.

February 2022

*False facts are highly injurious to the progress of science,  
for they often endure long; but false views, if supported by  
some evidence, do little harm, for everyone takes a  
salutary pleasure in proving their falseness.*

—**Charles Darwin**

(1809–1882)

*Effort and hard work construct the bridge that connects  
your dreams of reality.*

—**Daisaku Ikeda**

(President of Soka Gakkai International)

*To my lovely mum, Comfort Yaa Achiaa.*

# Acknowledgement

I would like to thank my supervisor, Professor Masayasu Atsumi, for his invaluable advice, support and suggestions that helped guide my research during my years as a doctoral student. Without his guidance, my dream of becoming a researcher in the field of Artificial Intelligence would not have been possible.

My heartfelt gratitude goes to Professor Tatsuo Unemi and Professor Norihiko Shinomiya at Department of Information Systems Science, Soka University. Without doubt, your kind support and suggestions during my student exchange and doctoral study periods have been one of the pillars that has brought me to this far. I am really grateful for your time. Also, I would like to thank Professor Ferdinand A. Katsriku of University of Ghana, Ghana, for proofreading my manuscripts for journal publications.

I cannot fully express my sincere thanks to my mother, my siblings, my fiancée and family friends for their prayers, financial and psychological support, encouragement, and trust that enable me to reach where I am today. I am really honoured and forever grateful. I also owe thanks to Mr. Tatsuro Suzuki and his family in Kikugawa, Shizuoka that I consider as my family in Japan since I met their son, Yuichi, at Soka University in September 2017. I am very grateful for their warm support and prayers.

My doctoral study was supported in part by the Soka University Makiguchi Foundation for Education, International Student Scholarship. This financial support is much appreciated. I also like to express my deep gratitude to all the staff at the International Students Office and the Graduate School of Science and Engineering Office, Soka University. I say big thank you to all for your kind support.

Lastly, to my friends and lab mates, I owe you thanks for your help and the good moment we had together, I say a big thank you.

# Abstract

Deep neural networks (DNNs) such as convolutional neural networks (CNNs) have enabled remarkable progress in the application of machine learning and artificial intelligence. Research scientists are gearing up for adopting DNN methods to their respective domain problems. Automated neural architecture search (NAS), also known as automated DNN (AutoDNN), aims to automate the architecture search of neural networks to enable researchers adopt DNN methods with ease, and with little or no expertise in deep learning.

As metaheuristic approach, automated NAS requires a representation scheme to encode the candidate solutions (architectures). Direct encodings of genetic algorithms and genetic programming have been widely employed in automated NAS methods. Though easy to implement, direct encoding cannot be easily modularized and the lack of distinctive separation of genotype and phenotype spaces limits their functional complexity. Therefore, it may be difficult for direct encodings to evolve modules (building-blocks) with shortcut and multi-branch connections which can improve training and enhance network performance in image understanding tasks.

This work presents a novel generative encoding, called *symbolic linear generative encoding* (SLGE), that combines the complementary strengths of gene expression programming (GEP) and cellular encoding (CE) for automatic architecture search of deep neural networks for image understanding. In particular, evolving modularized CNNs with shortcut and multi-branch modularity properties (similar to the ones commonly adopted by human experts) for remote sensing (RS) image understanding tasks such as scene classification and semantic segmentation. GEP is known for its simplicity in implementation and multi-gene chromosomes with flexible genetic modification, whereas CE has the ability to produce modular artificial neural networks (ANNs). Both GEP and CE are well established evolutionary computation methods which have experienced a lot of development and theoretical study. A large part of this previous work involves architecture search of ANNs in a small scale, and therefore this work provides the possibility for CNNs architecture development for image understanding tasks, particularly in the field of RS.

We adopt two automated NAS search strategies: *random search with early-stopping* and



---

*evolutionary algorithm*, to automatically evolve modularized CNNs architectures for classification of RS imagery scenes and semantic segmentation of aerial/satellite imagery respectively. Two types of multi-class image scene classification tasks were performed: single-label scene classification and multi-label scene classification, using four different remotely-sensed imagery datasets, to validate the expressiveness and tractability of SLGE representation space. Moreover, we constructed a two-separate SLGE representation spaces: normal cell and atrous spatial pyramid pooling (ASPP) cell. Then, using evolutionary algorithm with genetic operators such as uniform mutation, two-point crossover and gene crossover, we joint search for a normal cell and an ASPP cell as a pair of cells to build a modularized encoder-decoder CNN architecture for solving RS image semantic segmentation problem. Three RS semantic segmentation benchmarks were used to verify the performance of the SLGE architecture representation. By doing this, we also validated the effectiveness and robustness the proposed SLGE architecture representation. The results position SLGE architecture representation amongst the best of the state-of-the-art systems.

# Contents

<b>Acknowledgement</b>	<b>iv</b>
<b>Abstract</b>	<b>v</b>
<b>Contents</b>	<b>vii</b>
<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>xiii</b>
<b>List of Acronyms</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Context . . . . .	1
1.2 Motivation . . . . .	3
1.3 Research Aims and Objectives . . . . .	4
1.4 Contributions . . . . .	5
1.5 Outline of the Dissertation . . . . .	6
1.6 References . . . . .	7
<b>2 Background and Related Work</b>	<b>9</b>
2.1 Deep Learning and Deep Neural Networks . . . . .	9
2.2 Metaheuristic Optimization Methods . . . . .	15
2.3 Automatic Architecture Search of DNNs . . . . .	19
2.4 Remote Sensing Image Understanding with CNNs . . . . .	22
2.5 Summary . . . . .	23
2.6 References . . . . .	23
<b>3 Symbolic Linear Generative Encoding</b>	<b>34</b>
3.1 SLGE Background and Motivation . . . . .	34

3.2	The Symbolic Linear Generative Encoding . . . . .	38
3.3	Preliminary Experiments on CIFAR-10 and CIFAR-100 . . . . .	42
3.4	Extending SLGE to Remote Sensing Image Understanding . . . . .	45
3.5	Summary . . . . .	46
3.6	References . . . . .	47
<b>4</b>	<b>Extending SLGE to Remote Sensing Image Scene Classification</b>	<b>50</b>
4.1	Introduction . . . . .	51
4.2	Methodology . . . . .	52
4.3	Experiments . . . . .	54
4.4	Discussion . . . . .	64
4.5	Summary . . . . .	66
4.6	References . . . . .	66
<b>5</b>	<b>Extending SLGE to Aerial/Satellite Image Segmentation</b>	<b>70</b>
5.1	Introduction . . . . .	70
5.2	Methodology . . . . .	72
5.3	Experiments . . . . .	78
5.4	Results and Discussion . . . . .	82
5.5	Summary . . . . .	87
5.6	References . . . . .	87
<b>6</b>	<b>Conclusions and Future Work</b>	<b>93</b>
6.1	SLGE: Design and Development . . . . .	93
6.2	Extending SLGE to RS Image Understanding . . . . .	94
6.3	Future Work and Concluding Remarks . . . . .	95
6.4	References . . . . .	96
	<b>Appendices</b>	<b>97</b>
<b>A</b>	<b>Source Code</b>	<b>98</b>
<b>B</b>	<b>Paper I</b>	<b>99</b>
<b>C</b>	<b>Paper II</b>	<b>100</b>
<b>D</b>	<b>Paper III</b>	<b>101</b>
<b>E</b>	<b>List of Publications</b>	<b>102</b>

# List of Figures

2.1	A schematic representation of an artificial neuron [10]. . . . .	10
2.2	A schematic diagram of backpropagation algorithm with a mathematical model of artificial neuron [18]. . . . .	12
2.3	An example of 2D convolution operation [4]. The resulting feature map is a weighted sum of the pixels $(a, b, c, \dots, l)$ of an input image and a kernel of size $2 \times 2$ which represents the weights $(w, x, y, z)$ that are learned during training of the network. Stride of 1 pixel is used with no zero paddings. . . . .	13
2.4	An example of max pooling operation [30]. . . . .	14
2.5	An example of fully-connected layer with a 2D array of pixel values [31]. . . . .	14
2.6	An example of a classical convolutional neural network architecture [48]. . . . .	15
2.7	A general framework of evolutionary algorithms. . . . .	18
3.1	A genotype-phenotype representation in GEP: (a) is a GEP chromosome with only one gene of head length 4, and (b) is the expression-tree of the chromosome in (a) with which is expressed mathematically in Equation 3.2. . . . .	36
3.2	(a) An example of CE grammar-tree consists of four different instructions. The instructions are applied sequentially, based on CE procedure, from the initial node 0 and grows into the final feed-forward neural network in (b). . . . .	37
3.3	A schematic representation of SLGE genotype. . . . .	39
3.4	The graphical illustration of <i>SQE</i> , <i>CPI</i> and <i>CPO</i> graph transformation functions of cellular encoding (CE). . . . .	40
3.5	A schematic representation of SLGE phenotype. This is the phenotype (cell) representation of the chromosome in Figure 3.3. . . . .	41
3.6	Examples of CIFAR-10 dataset images from the 10 classes [32]. . . . .	43
3.7	SLGE single-gene vs multi-gene chromosomes on CIFAR-10. . . . .	45

3.8	Visualization of the cell or building-block for the best discovered network by SLGE on CIFAR-10 dataset. . . . .	45
3.9	Examples of RS images: (a) Single-label RGB high-resolution aerial images from NWPU-RESISC45 dataset [38], and (b) Multi-label multispectral satellite images from BigEarthNet dataset [39]. . . . .	46
4.1	The entire CNN architecture is constructed by stacking an auto-generated cell (see Fig. 4.3) as <i>cell blocks</i> marked in yellow. The architecture begins with two $3\times 3$ convolution layers followed by several auto-generated cell blocks repeated $N$ times with average pooling layers inserted in between them to downsample the spatial resolutions. . . . .	52
4.2	SLGE linear structure of the best discovered network. <i>SEQ</i> , <i>CPO</i> , <i>CPI</i> and <i>END</i> are transformation functions (refer to section 3.2 of chapter 3 for details), and $1\times 1$ , $3\times 3$ and $3\times 3$ Dw (depthwise) are convolution operations. . . . .	53
4.3	SLGE module of the best discovered network. A cell representation of the linear string shown in Fig. 4.2. The convolution operations without successor are depthwise concatenated to produce an output, and if an operation has more than one predecessor, the feature maps of the predecessors are added together. This is a common approach in cell-based search [13]. . . . .	53
4.4	Search results of the 400 proxy networks trained according to the proxy task described in section 4.2.2. In each search space, we performed 5 experiments by sampling 20 individual cells per experiment to build 20 different proxy architectures and trained each for 100 epochs. The results show accuracy distribution of 20 networks per experiment in each search space. It took 9.6 GPU days to train and evaluate all the 400 proxy networks. . . . .	57
4.5	Confusion matrix of our best network on NWPU-RESISC45 (Table 4.3). Diagonal values are correct predictions (%), non-diagonals values are incorrect predictions (%) and the dash (–) means no incorrect prediction. . . . .	59
4.6	An example of most confused classes, <i>palace</i> and <i>church</i> , in NWPU-RESISC45 (see Fig. 4.5). It shows the similarity in textural and structural features among the images. . . . .	59
4.7	Confusion matrix of our best network on AID benchmark (Table 4.4). Diagonal values are correct predictions (%), non-diagonal values are incorrect predictions (%) and the dash (–) means no incorrect prediction. . . . .	62
4.8	Examples of confused classes in AID (see Fig. 4.7). This shows the textural similarity among the images: (a) <i>square</i> and <i>centre</i> classes (b) <i>bare-land</i> and <i>desert</i> classes. . . . .	62

4.9	An example of the BigEarthNet RGB images with the true multi-labels and the predicted multi-labels by our best network. The green labels are true labels correctly predicted; blue labels are true labels that were not predicted; and red labels are wrongly predicted labels. . . . .	64
5.1	A network structure diagram of the modularized encoder-decoder CNN architecture. The normal cell and ASPP cell described in section 5.2.1 are used in the encoder and the decoder respectively. . . . .	73
5.2	Schematic representation of SLGE pair of chromosomes. The pair of chromosomes of the best network, SLGENet-3, in Tables 5.3, 5.4 and 5.5. (a) A chromosome of 3 genes with head length of 3 for a normal cell (see Fig. 5.3a). The head of a gene may consists of <i>SEQ</i> , <i>CPO</i> , <i>CPI</i> or <i>END</i> (CE graph transformation functions), and a tail of operations defined for normal cell. (b) A chromosome of 3 genes with head length of 1 for ASPP cell (see Fig. 5.3b). The head of gene may be <i>CPO</i> or <i>END</i> with a tail which consists of possible operations defined for ASPP cell. The ‘AU4’, ‘AU5’ and ‘AU7’ are atrous depthwise-separable convolution with rates 4, 5 and 7 respectively; and ‘AP4’ is an average spatial pooling with size 4. See section 5.2.1 for the operations defined for normal cell and ASPP cell. . . . .	74
5.3	Evolutionary discovered pair of cells of the best network, SLGENet-3, in Tables 5.3, 5.4 and 5.5. The (a) normal cell and (b) ASPP cell are the phenotypes of the chromosomes illustrated in Fig. 5.2a and Fig. 5.2b respectively. (a) The convolution operations without successor are depthwise concatenated to produce an output, and if an operation has more than one predecessor, the feature maps of the predecessors are added together. (b) The feature maps of all branches within the cell are depthwise concatenated to produce an output. . . . .	74
5.4	Illustration of two-point crossover operation. (a) Two selected parent chromosomes for the crossover operation. (b) The generated offspring. The red elements in Parent A are exchanged with the blue elements in Parent B to generate Offspring A and B respectively. . . . .	76
5.5	Illustration of gene crossover operation. (a) Two selected parent chromosomes for the crossover operation. (b) The generated offspring. The gene with red elements in Parent A are exchanged with the gene with blue elements in Parent B to generate Offspring A and B respectively. . . . .	77
5.6	The overview of SLGE evolutionary neural architecture search. . . . .	77

---

5.7	Evolutionary trajectories of the proposed method in searching for the best CNN architecture on Vaihingen dataset. (a) Trajectory of the pair of chromosomes ‘Pair 1’ (see Table 5.1). (b) Trajectory of the pair of chromosomes ‘Pair 2’ (see Table 5.1).	81
5.8	Examples of semantic segmentation results on ISPRS Vaihingen dataset. $256 \times 256$ image patches of Vaihingen area 2. The predicted segmentation maps are produced by the best network, SLGENet-3. . . . .	84
5.9	Examples of semantic segmentation results on ISPRS Potsdam dataset. $256 \times 256$ image patches of Potsdam area 3_13. The predicted segmentation maps are produced by the best network, SLGENet-3. . . . .	85
5.10	Examples of semantic segmentation results on ISPRS UAVid dataset. $400 \times 400$ image patches of UAVid seq37/000300.png. The predicted segmentation maps are produced by the best network, SLGENet-3. . . . .	86

# List of Tables

4.1	The four different SLGE search spaces use in the experiments. . . . .	53
4.2	The results of top 20 networks found via proxy task and fully trained on NWPU-RESISC45 in the large-scale architecture settings (see Fig. 1). Boldface is the best result. . . . .	57
4.3	Comparisons between our method and the baseline network and the state-of-the-art transfer learning (feature extraction and fine-tuning) methods with 20/80 training-testing ratio on NWPU-RESISC45 dataset. Here our method is auto-generated architectures discovered on NWPU-RESISC45. The baseline results are based on our training. Boldface is the best result. . . . .	58
4.4	Comparisons between our method and the baseline network and the state-of-the-art transfer learning (feature extraction and fine-tuning) methods with 50/50 training-testing ratio on AID dataset. Here our method is auto-generated architectures discovered on NWPU-RESISC45 which are evaluated on AID. The baseline results are based on our training. Boldface is the best result. . . . .	60
4.5	Comparisons between our method and the state-of-the-art handcrafted CNN models and gradient-based NAS method fully trained on NWPU-RESISC45 (NWPU) and AID with 80/20 training-testing ratio. The results of the state-of-the-art handcrafted models are reported in [43]. Boldface is the best result. . . . .	61
4.6	Comparisons between our method and the baseline network and the state-of-the-art fine-tuned pre-trained benchmark models on EuroSAT in RGB space and RGB-NIR space. Here our method is auto-generated architectures discovered on NWPU-RESISC45 which are evaluated on EuroSAT dataset. Boldface is the best result. . .	63



4.7	Comparisons between our method and the baseline network and the state-of-the-art handcrafted CNN models on BigEarthNet in RGB space and RGB-NIR space. Here our method is auto-generated architectures discovered on NWPU-RESISC45 which are evaluated on BigEarthNet dataset. The results of the state-of-the-art handcrafted models are reported in [36]. Boldface is the best result. . . . .	64
5.1	The two pairs of chromosomes settings for Normal cell and ASPP cell used in the experiments. Chromosome settings for ASPP cell is the same for both pairs. . . . .	81
5.2	Baseline results on Vaihingen datasets. The values are the overall accuracy of the ten baseline networks, which five each were randomly generated from the two pairs of chromosomes settings in Table 5.1. Boldface is the best results. . . . .	82
5.3	Comparison between our method and the baseline and the state-of-the-art models (NAS and handcrafted) on the Vaihingen test set. The values are the per-class $F_1$ scores, mean $F_1$ and overall pixel-wise classification accuracy. Boldface is the best result. . . . .	83
5.4	Comparison between our method and the state-of-the-art handcrafted models on the Potsdam test set. The values are the per-class $F_1$ scores, mean $F_1$ and overall pixel-wise classification accuracy. Boldface is the best result. . . . .	85
5.5	Comparison between our method and the results reported in Lyu <i>et al.</i> [66] and the online leaderboard on the UAVid dataset. The values are the per-class IoU, mean IoU and the overall pixel-wise classification accuracy. Boldface is the best result . . .	86

# List of Acronyms

<b>AI</b>	Artificial Intelligence
<b>ANN</b>	Artificial Neural Network
<b>ASPP</b>	Atrous Spatial Pyramid Pooling
<b>AutoDNN</b>	Automated Deep Neural Network
<b>CE</b>	Cellular Encoding
<b>CPI</b>	Copy Input
<b>CPO</b>	Copy Output
<b>CNN</b>	Convolutional Neural Network
<b>DNN</b>	Deep Neural Network
<b>EA</b>	Evolutionary Algorithm
<b>GA</b>	Genetic Algorithm
<b>GEP</b>	Gene Expressing Programming
<b>GP</b>	Genetic Programming
<b>ISPRS</b>	International Society for Photogrammetry and Remote Sensing
<b>ML</b>	Machine Learning
<b>NAS</b>	Neural Architecture Search
<b>PAR</b>	Parallel
<b>REC</b>	Recursive
<b>RS</b>	Remote Sensing
<b>SEQ</b>	Sequential
<b>SLGE</b>	Symbolic Linear Generative Encoding
<b>SLGENet</b>	Symbolic Linear Generative Encoding Network
<b>UAV</b>	Unmanned Aerial Vehicle

# Chapter 1

## Introduction

*Robots [automations] are not going to replace humans, they are going to make their jobs much more humane. Difficult, demeaning, demanding, dangerous, dull—these are the jobs robots [automations] will be taking.*

—**Sabine Hauert**

(Co-founder of Robohub.org)

This chapter provides an overview of the work presented in this PhD dissertation. Section 1.1 briefly presents the context of the work in this dissertation, to give better understanding of the different research fields surrounding the work. Section 1.2 describes the motivation for the work presented in this dissertation. In section 1.3, the core aims of the work and the objectives we propose to achieve are presented. The contributions which have been derived from this dissertation are presented in section 1.4. Finally, section 1.5 outlines the structure of the dissertation.

### 1.1 Context

In a broad perspective, this dissertation involves a research work in the area of artificial intelligence (AI) and machine learning (ML). AI is considered as an old field of computer science with a focus on making intelligent machines, that is, developing machines that can operate or act intelligently in a wide range of situations. The field of ML is a subfield of AI that is concerned with making it possible for machines to learn automatically from data or experience. In other words, ML aims at making it possible for machines to extract meaningful information from data that it would be possible to automate it.

Deep learning is relatively a new field of ML but its fundamental concept of neural networks was theorized and implemented in 1943 by McCulloch and Pitts. Deep learning methods use multiple layers of neural networks to extract insightful features automatically from data and learn

---

a model representation from such features. The work by Krizhevsky *et al.* in 2012, AlexNet, has remarkable transformed the landscape of ML algorithms. The ML community has come to conclusion that deep neural networks such as convolutional neural networks (CNNs) can yield better performance on plethora of AI/ML problems with large volumes of data compared to statistical ML-based techniques. The visual perception tasks such as image classification and semantic image segmentation are among the key notable AI/ML problems which CNNs have shown excellent results. The performance of CNNs depends largely on the network architecture (and the network training). The work in this dissertation concerns an automated design of deep neural networks (AutoDNN) for visual perception tasks. In particular, automated neural architecture search (NAS) for CNN in remote sensing image understanding tasks. Thus, this work involves *search and optimization*, *neural networks* and *image understanding*, which are key disciplines in the field of AI/ML.

In general, in a search problem, the aim is to find a goal condition within a large set of objects (search space) by systematically examining within the internal representation of the search space for a path that satisfies the goal condition. And optimization problem can be formulated as a search problem which concerns finding (near)-optimal solution with respect to some objective function. Exact optimization methods are efficient for problems that can be solved with polynomial time. However, many optimization problems like combinatorial problems with a large set of feasible solutions become intractable and can not be solved using exact optimization methods. Because the time for solving these problems with exact optimization methods increases exponentially. Metaheuristics optimization methods use problem-invariant and widely applicable search strategy to solve wide range of difficult problems including combinatorial problems with less computational effort. These optimization methods use a search strategy that search through the search space on a meta-level. Metaheuristics employ some degree of randomness to find (near)-optimal solutions for hard optimization problems. They are often considered as “black-box” algorithms which we can plug-in arbitrary search operators, representation and objective function as needed to solve a particular complex problem. In combinatorial problems, metaheuristics can provide a sufficiently good solution with less computational effort. AutoDNN (automated NAS) can be described as a combinatorial optimization problem which involves searching for the (near)-optimal DNN architecture  $a^*$  from a large set of possible candidate solutions  $\mathcal{A}$  that maximizes an objective function  $f(a)$  with respect to certain constraints. Thus in this dissertation, AutoDNN is considered as an optimization problem which can best be solved within a reasonable computational effort using metaheuristic approach.

The other aspect of the work in this dissertation involves image understanding tasks in the field of remote sensing (RS). RS images are remotely-sensed data of the Earth observation. They

are acquired by satellite constellations such as Landsat <sup>1</sup> and Sentinel <sup>2</sup> which are placed in the space, and also by drones and airborne means. Approximately, 1.6TB of compressed optical imagery data are produced per day by Sentinel-2A and 2B satellites only and they observe the entire planet each week. Such a massive amount of remotely-sensed imagery provide a valuable resource with which to address many important social and environmental issues. Despite the specific features of remotely-sensed imagery with respect to spectral, spatial and radiometric resolutions, the understanding and interpretation of these images involves extracting information from the images and visual perception tasks as computer vision.

Computer vision is a field of AI which is concerned with developing techniques to enable machines and systems understand and interpret digital images such as photographs, videos and other visual inputs. In other words, computer vision enables machines to see, observe and understand their environment by extracting meaningful information from digital images. The problem of visual perception seems very simple since it is trivially solved by human. However, it largely remains an unsolved problem for computers and machines. This is attributed to both our limited understanding of the human vision and the complex nature of visual perception in a dynamic environment. For about 60 years, researchers in computer vision have been working hard to develop techniques for machines to imitate the human vision's ability to interpret dynamic scenes from visual signals, for example, to recognize objects in photographs. The emergence of deep learning has significantly improved this effort, and currently almost every computer vision task is carried out with DNNs. DNNs approach has also shown a remarkable performance in the field of RS image understanding. The common cause of computer vision and RS image understanding is to seek to understand and interpret imagery data. Thus far, the work in this dissertation is can be placed at the intersection of *deep learning (neural architecture search)*, *search and optimization (metaheuristic)*, and *remote sensing image understanding (computer vision)*. We propose to implement automatic architecture search of deep neural networks to search for convolutional neural networks for understanding and interpretation of imagery data, particularly, remote sensing images.

## 1.2 Motivation

Deep neural networks (DNNs) such as convolutional neural networks (CNNs) have enabled remarkable progress in the application of machine learning and artificial intelligence. CNNs methods have achieved excellent results in a large variety of problems with multiple modalities such as images, speech, and text. Research scientists and engineers are gearing up for adopting CNNs to solve problems of their respective disciplines. The performance of CNNs depends on a non-trivial

---

<sup>1</sup><https://landsat.gsfc.nasa.gov/>

<sup>2</sup><https://sentinels.copernicus.eu/web/sentinel/home>

manual crafting of their architectures (and the training of the parameters). Besides being error-prone and time-consuming, designing and developing a well-performing CNN architectures manually for specific tasks requires a certain level of expertise and experience in deep learning. Neural architecture search (NAS), also known as AutoDNN, has emerged to automate the architecture design of deep neural networks to enable researchers adopt CNNs and other deep learning methods with easy, and with little or no expertise in deep learning.

As metaheuristic optimization problem, NAS requires a representation scheme to encode the candidate solutions (neural network architectures). The direct encodings of genetic algorithms (GA) and genetic programming (GP) have been widely employed in NAS methods to encode candidate solutions in the search space [1, 2]. Though easy to implement, direct encoding cannot be easily modularize and the lack of distinctive separation of genotype and phenotype spaces limits their functional complexity [3–5]. It may be difficult for direct encodings to evolve modules (building-blocks) with shortcut (skip) and multi-branch connections [6] which can improve training and enhance network performance [7, 8]. Thus, some researchers have adopted human experts designed modules such as ResNet-Blocks [9] and DenseNet-Blocks [10] into direct encoding to evolve modularized CNN architectures [11–13]. Evolving modularized CNN architectures with modularity properties of shortcut and multi-branch connections commonly adopted by human experts represents one of the key motivations of this work.

The alternative to direct encoding is a generative encoding which can produce modular and regular structures [3, 5]. The second motivation of this work is pertained to exploring the strengths of two generative encodings, gene expression programming (GEP) and cellular encoding (CE), with the aim of harnessing their complementary strengths into a new encoding scheme, which we called *symbolic linear generative encoding* (SLGE). GEP was introduced by Ferreira in 2001 [4], and it inherits the advantages of the GA and GP. GEP is known for its simplicity in implementation and multi-gene chromosomes with flexible genetic modification. On CE, it was proposed by Gruau in 1994 [14]. CE has the ability to produce modular neural networks using graph grammar. Both GEP and CE are well established evolutionary computation methods which have experienced a lot of development and theoretical study [15–17]. A large part of this previous work involves architecture search of artificial neural networks (ANNs) in a small scale, and therefore this work provides the possibility for CNNs architecture development.

### 1.3 Research Aims and Objectives

Based on the aforementioned motivations and from application perspective, the main aims of work in this dissertation are as follows:

1. To explore the capability of GEP with CE in evolving CNNs architectures. In particular, to investigate the viability of combining the encoding schemes of GEP and CE into a new scheme called SLGE for CNNs architecture representation.
2. To investigate whether the multi-gene chromosomes and modularity properties of GEP and CE respectively can be used to evolve network modules with shortcut connections and multi-branch connections to develop modularized CNN architectures.
3. To investigate the expressiveness and tractability of the representation space developed with GEP of CE (SLGE) for evolving modularized CNN architectures.
4. To investigate the suitability of applying GEP of CE (SLGE) to evolve CNNs for visual perception tasks such as image classification and image semantic segmentation and its robustness when transferred to others benchmarks.

And to achieve these aims, we propose the accomplishment of the following objectives:

1. Design and development of a novel generative encoding, SLGE, which adopt the simplicity features of GEP with modularity properties of CE to model representation space of CNN architectures.
2. Development of (metaheuristic) optimization algorithms able to optimize the architecture search of CNNs automatically based on objective (1), given specific visual perception tasks.
3. Evaluation of the automatically evolved CNNs on various visual perception tasks in order to validate that they are competitive to, if not better than, manually designed ones. The tasks should cover a spectrum of remote sensing image understanding tasks such as scene classification and aerial image segmentation.

## 1.4 Contributions

The scientific contributions derived from the work in this dissertation are summarized as follows:

1. We introduced a novel encoding scheme, SLGE, which extends GEP to AutoDNN by injecting the modularity features of CE into the linear representation of GEP to evolve modularized CNN architectures.
2. We demonstrated that SLGE can discover modules with shortcut and multi-branch connections commonly adopted by human experts and develop modularized CNN architectures of arbitrary complexity with fewer parameters.

3. We achieved results that are competitive to, or even exceed, human experts designed networks in various RS image understanding tasks. For each of the tasks used in the evaluation, the results of the best automatically discovered CNNs architecture contributed to the state-of-the-art.
4. By evolving and evaluating CNN architectures via random search policy with early-stopping and evolutionary algorithm on remotely-sensed imagery data, we have extended AutoDNN approach to the field of RS image understanding.

## 1.5 Outline of the Dissertation

The dissertation is organized on the basis of 2 papers published in international journals and a paper presented at international conference.

Chapter 2 describes the necessary background of the work in this dissertation which include deep neural networks and metaheuristic optimization methods such as random search strategy and evolutionary algorithm. Also, a review of relevant related work in neural architecture search and remote sensing image understanding are presented.

Chapter 3 presents a comprehensive description of SLGE, the underlying representation scheme of the NAS method of interest in this dissertation. The description includes the fundamental algorithm and distinguishing features of SLGE; and a discussion of implementation and preliminary experiments which have been performed. Preliminary experiments on the CIFAR-10 and CIFAR-100 image classification benchmarks to validate the approach are reported. The algorithm and the preliminary results have been published in: C. Broni-Bediako, Y. Murata, L. H. B. Mormille, and M. Atsumi, “Evolutionary NAS with gene expression programming of cellular encoding”. In *Proceedings of 2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 2670–2676, 2020, DOI: 10.1109/SSCI47803.2020.9308346.

In Chapter 4, SLGE is extended to RS image understanding tasks and evaluated experimentally in various RS scene classification benchmarks. Several experiments with different configurations of SLGE hyperparameters are performed using random search with early-stopping strategy to demonstrate the expressiveness and tractability of the representation space developed with SLGE to evolve modularized CNN architectures. The description of the experiments performed and the analysis of the experimental results have been published in: C. Broni-Bediako, Y. Murata, L. H. B. Mormille and M. Atsumi, “Searching for CNN architectures for remote sensing scene classification”, *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–13, 2021, DOI: 10.1109/TGRS.2021.3097938.

Chapter 5 presents an extension of SLGE algorithm which enables the construction of two-



separate search space representations, to joint search for pair of encoder and decoder modules using evolutionary algorithm to build modularized encoder-decoder CNN architectures for semantic segmentation of RS images. Several experiments are presented to validate the effectiveness SLGE. And the discovered architectures are transferred to other benchmarks to demonstrate their robustness. The proposed method achieved significant results with affordable computational effort. The algorithm and the experimental results have been published in: C. Broni-Bediako, Y. Murata, L. H. B. Mormille and M. Atsumi, “Evolutionary NAS for Aerial Image Segmentation with Gene Expression Programming of Cellular Encoding”, *Neural Computing and Applications*, 2021, DOI: 10.1007/s00521-021-06564-9.

Chapter 6 provides the final discussion and concluding remarks with suggestions for further improvement of SLGE for evolving modularized CNN architectures.

## 1.6 References

- [1] E. Galvan and P. Mooney, “Neuroevolution in deep neural networks: Current trends and future challenges,” *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 01, pp. 1–1, 2021.
- [2] G. A. Vargas-Hákim, E. Mezura-Montes, and H.-G. Acosta-Mesa, “A review on convolutional neural networks encodings for neuroevolution,” *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2021.
- [3] J. Fekiač, I. Zelinka, and J. C. Burguillo, “A review of methods for encoding neural network topologies in evolutionary computation,” in *Proceedings 25th ECMS*, 2011, pp. 410–416.
- [4] C. Ferreira, “Gene expression programming: a new adaptive algorithm for solving problems,” *Complex Systems*, vol. 13, no. 2, pp. 87–129, 2001.
- [5] J. Clune, “Evolving artificial neural networks with generative encodings inspired by developmental biology,” Doctoral Dissertation, Michigan State University, 2010.
- [6] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, p. 4278–4284.
- [7] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.
- [8] T. Liu, M. Chen, M. Zhou, S. S. Du, E. Zhou, and T. Zhao, “Towards understanding the importance of shortcut connections in residual networks,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [10] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.
- [11] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, “Completely automated cnn architecture design based on blocks,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2019.

- 
- [12] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 350–364, 2020.
- [13] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE transactions on cybernetics*, vol. 50, no. 9, pp. 3840–3854, 2020.
- [14] F. Gruau, "Neural network synthesis using cellular encoding and the genetic algorithm." Doctoral Dissertation, L'universite Claude Bernard-lyon I, 1994.
- [15] F. Gruau, D. Whitley, and L. Pyeatt, "A comparison between cellular encoding and direct encoding for genetic neural networks," in *Proceedings of the 1st annual conference on genetic programming*. MIT Press, 1996, pp. 81–89.
- [16] F. Gruau, "Artificial cellular development in optimization and compilation," in *Towards Evolvable Hardware*. Berlin, Heidelberg: Springer, 1996, vol. 1062, pp. 48–75.
- [17] J. Zhong, L. Feng, and Y.-S. Ong, "Gene expression programming: A survey," *IEEE Computational Intelligence Magazine*, vol. 12, no. 3, pp. 54–72, 2017.

## Chapter 2

# Background and Related Work

*The brain is a statistical, probabilistic system, with logic and mathematics running as higher-level processes. The computer is a logical, mathematical system, upon which higher-level statistical, probabilistic systems, such as human language and intelligence, could possibly be built.*

— **George B. Dyson**  
(Turing's Cathedral)

The work in this dissertation is built upon deep learning and metaheuristic optimization, and it is towards the applications in image understanding tasks in the field of remote sensing. This chapter briefly introduces the basic concept of deep learning with a focus on convolutional neural networks in section 2.1. Metaheuristic optimization methods, random search strategy and evolutionary algorithm, which are adopted in this work are briefly described in section 2.2. In sections 2.3 and 2.4, the review of relevant related work in automatic architecture search of deep neural networks and remote sensing image understanding with convolutional neural networks are presented respectively.

## 2.1 Deep Learning and Deep Neural Networks

Deep learning is a subfield of machine learning in artificial intelligence that is concerned with computational models which use multiple processing layers to learn representations of data with multiple levels of abstraction [1]. These processing layers are based on the fundamental concepts of artificial neural networks, which are repeatedly stacked to form the computational models called deep neural networks (DNNs). Artificial neural networks (ANNs), commonly referred to as neural networks, have been around for more than 70 years [2]. The emergence of deep learning in the early 2000s rekindled neural network research, which has ameliorated the performance in visual perception tasks and other cognitive tasks such as language and speech recognition [1, 3]. There are several architectures of DNNs, including:

- Multilayer perceptrons (MLP), the simplest and the oldest neural network architecture;
- Convolutional neural networks (CNNs), the architecture mostly used for image and video processing; and
- Recurrent neural networks (RNNs), the architecture which are used largely for sequential data such as text and times series.

This section presents a brief introduction to deep learning. In particular, the basic concepts of neural networks and the CNNs which are commonly used for visual perception tasks. LeCun *et al.* [1] and Bengio *et al.* [3] provide a comprehensive overview of deep learning. For further information and implementation details, we refer to the text *Deep Learning* by Ian Goodfellow, Yoshua Bengio and Aaron Courville (2016) [4].

### 2.1.1 Basic Concepts of Neural Networks

A neural network is a machine learning technique able to solve mathematically ill-defined problems with a network of computationally simple elements which are inspired by biological neural networks [2, 5, 6]. The basic definition given by K. Gurney in his book titled *An Introduction to Neural Networks* is: “A neural network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the interunit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns” [7]. Neural networks can be used for both regression and classification problems. Most importantly, they are universal approximators. Which means they are capable of arbitrarily accurate approximation to any complex function if the network has enough hidden neurons [8, 9].

#### 2.1.1.1 Artificial Neuron

The fundamental building block in neural networks is the mathematical model of an artificial neuron that accepts a set of inputs  $\mathbf{x} = \{1, 2, \dots, m\}$  and consists of the following three basic components in addition to the inputs  $\mathbf{x}$  and the output  $\mathbf{y}$  as shown in Figure 2.1.

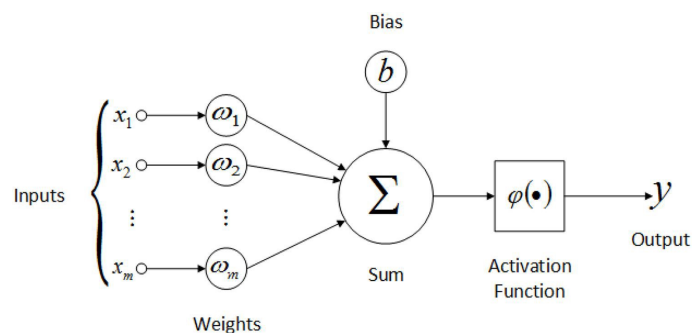


Figure 2.1: A schematic representation of an artificial neuron [10].

1. The connecting edges which provide weights  $\omega_i$  to the input values  $x_i$  and a bias term  $b$ .
2. The summation function  $\Sigma$  that sums the weighted input values and the bias as an input for the activation function  $\varphi$  as defined in Equation (2.1).

$$\Sigma = \langle \omega_i, x_i \rangle + b \quad (2.1)$$

3. The activation function  $\varphi$  which acts on the weighted input values given in Equation (2.1) to produce an output value  $\mathbf{y}$ . The activation function that maps the summation function to the output value of the neuron is given in Equation (2.2)

$$\mathbf{y} = \varphi(\Sigma) = \varphi(\langle \omega_i, x_i \rangle + b) \quad (2.2)$$

The output  $\mathbf{y}$  could be unidimensional or a multidimensional of  $d$  values.

Historically, the original activation function was a threshold function. To enable neural networks learn complex nonlinear phenomenon, nonlinear functions are largely used recently. The common activation functions are the sigmoid function  $\sigma(x)$  which restricts input values to the range of (0, 1), the hyperbolic tangent function  $\tanh(x)$  which also restricts input values to the range of (-1, 1), and the rectified linear unit function  $\phi(x)$  which returns  $\max(0, x)$  as the output of the input values [6, 11, 12]. Because of vanishing gradient problem [13] that sigmoid and hyperbolic tangent functions might cause in neural networks, the default activation function in DNNs such as CNNs is the rectified linear units (ReLU) which is not susceptible to vanishing gradient and can enable DNNs to learn faster and perform better [12, 14].

In generally, the weights  $\omega_i$ , also called parameters, are estimated from learning samples via stochastic gradient decent (SGD). The work by Rumelhart *et al.* [15] and LeCun *et al.* [16, 17] on backward propagation algorithm, though not a learning algorithm, but its efficiency in computing gradient to find the parameters that minimize the cost function of a neural network has made DNNs a practical tool in solving real-world problems.

### 2.1.1.2 Backward Propagation

The backward propagation algorithm, also known as backpropagation for short, has a long history. It was proposed in the 1970s, but its potency was realized in the 1980s [15, 17]. The algorithm has two distinct phases called forward pass (feedforward calculation) and backward pass (error-backpropagation), which alternates several times through the layers of the neural network during the learning process to find the parameters that minimize the cost function of the network as shown in Figure 2.2. The algorithm estimates the parameters in the network such that for every input

value in the training sample, the network produces an output value which closely matches the prescribed target value. A brief summary of backpropagation algorithm is given in Algorithm 1. We refer to Goodfellow *et al.* [4] for further information and implementation details.

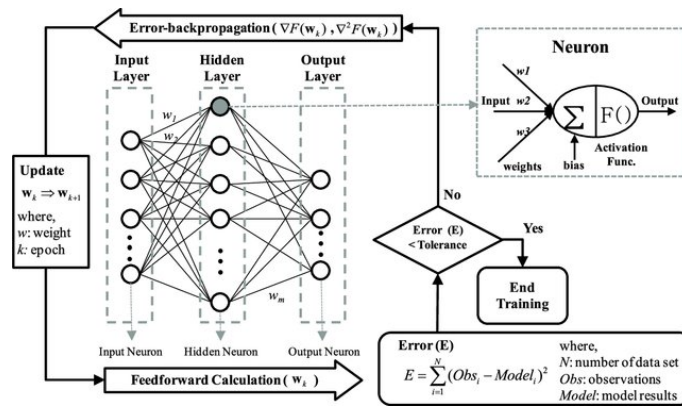


Figure 2.2: A schematic diagram of backpropagation algorithm with a mathematical model of artificial neuron [18].

---

**Algorithm 1** A simple description of backpropagation algorithm.

---

- 1 Initialize the weights randomly and stop condition.
  - 2 For each input set compute the activation of the hidden units and output units.
  - 3 Calculate the error at each output, then the derivative of the activation w.r.t each parameter.
  - 4 Pass the error from the output back to the hidden layer.
  - 5 Update the weights from the hidden to output layer.
  - 6 Exit and return the weights if stop condition is met, else go to Step 2.
- 

### 2.1.2 Convolutional Neural Networks

In traditional ANNs, learning normally requires extraction of variables of interest, called features, by the human experts. It needs a lot of experience to extract image features for visual perception tasks [19]. CNNs are deep neural networks that have the ability to automatically extract and learn image features, and have achieved the state-of-the-art performance in visual perception tasks [20–23]. The idea behind CNN was first put forward in 1979 by Fukushima in his work: the neocognitron [24, 25]. The CNNs proposed by LeCun *et al.* [16] revolutionized the field of computer vision, and now they are widely used for visual perception tasks such as image classification, image segmentation and object recognition. The success of CNNs in visual perception tasks is largely because they can act directly on patches of image with three colour channels (RGB), thereby preserving the spatial relationship among the pixels in an image [1, 20]. CNNs are perhaps the most popular and used DNNs architecture in many other applications, including audio and speech processing, language

processing, among others [26–29]. The basic components of CNNs architecture are: convolutional layers, pooling layers and fully-connected layers. A brief description is given here and we refer to Goodfellow *et al.* [4] for further information and implementation details.

### Convolutional Layer

The convolution layer uses kernels, also known as filters, to compute neurons that are connected to local patches of input image, by computing the dot product between their weights and the local patches of the input image. As shown in Figure 2.3, the convolution operation is performed by scanning through an input image with a fixed size kernel. The kernel is shifted by the stride of  $s$  pixels, and to preserve the input volume size,  $p$  zero paddings must be added to the boarder of the input. Mathematically, for 2-dimensional input image, the convolution operation is expressed as:

$$(K * I)(i, j) = \sum_{m, n} K(m, n) I(i + n, j + m) \tag{2.3}$$

where  $K$  is a convolution kernel applied to a 2D input image  $I$ . The resulting output is normally fed into an activation function, generally the ReLU function, to produce an activation map, also called feature map.

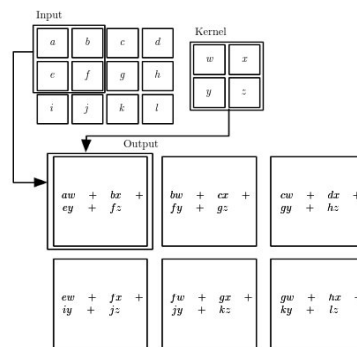


Figure 2.3: An example of 2D convolution operation [4]. The resulting feature map is a weighted sum of the pixels  $(a, b, c, \dots, l)$  of an input image and a kernel of size  $2 \times 2$  which represents the weights  $(w, x, y, z)$  that are learned during training of the network. Stride of 1 pixel is used with no zero paddings.

### Pooling Layer

Incorporating pooling layer into the CNNs provides for invariance of object class to translations, which is important for object identification. The pooling layer also reduces the spatial dimension of the input volume and it is referred as downsampling operation. Pooling layers are usually placed in between convolutional layers and thus reduces number of parameters and computational cost. As the convolutional operation, pooling operation is performed with stride  $s$  on local patches of the input image. The common pooling operations in CNNs are max and average pooling where

the maximum and mean value of local patches are computed, respectively. Figure 2.4 depicts max pooling operation with a kernel of size  $2 \times 2$  and a stride of 2 pixels.

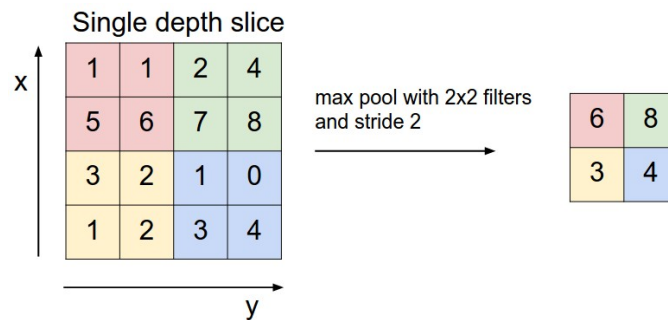


Figure 2.4: An example of max pooling operation [30].

### Fully-Connected Layer

The fully-connected layer is an ordinary multilayer perceptron. As the name implies, in this layer each input is connected to all neurons. Generally, after stacking several convolution and pooling layers, the last activation map volume is transformed into a vector and then fully-connected layers are added at the end of the architecture to compute class scores. We lose the spatial relationships within the data when the input volume is transformed into a vector as input to the fully-connected layer as shown in Figure 2.5.

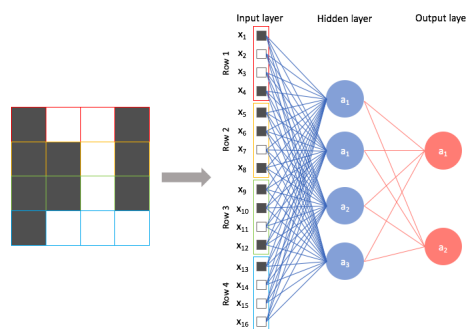


Figure 2.5: An example of fully-connected layer with a 2D array of pixel values [31].

#### 2.1.2.1 Common CNN Architectures

For almost a decade now, after the celebrated AlexNet [20] which set a new record on the ImageNet [32] classification competition in 2012, various CNN architectures have been proposed [33–35]. CNN architecture design is a crucial factor to its performance. Designing CNN architectures is more of engineering than science. In most classical CNNs such as AlexNet [20] and VGG-Net [36], a chain of a convolution layer plus a ReLU activation function followed by a pooling layer is repeatedly stacked several times, then fully connected layers are added at the end (see Figure 2.6). For a



typical classification tasks, as shown in Figure 2.6, the computed class scores of the network are normally fed into a softmax function to predict the class distribution.

Recent remarkable progress in the performance of CNNs is largely due to structural re-organization of the processing-unit and the development of novel building-blocks (modules), as well as the increase in the network depth [35]. Such innovative architectures have been introduced for image classification tasks, which include GoogLeNet [37], ResNet [38], Inception-ResNet [39], DenseNet [40], CapsuleNet (CapsNet) [41], among others. For example, GoogLeNet used multiple output channels and a novel modules (called inception blocks) which are repeatedly stacked, ResNet introduced a shortcut (skip) connection which is an additive transformation also known as residual connection, and Inception-ResNet combined the power of inception blocks and residual connections to construct CNN architectures.

In other visual perception tasks such as object detection and semantic segmentation, many interesting architectures are built based on ResNet, VGG and Inception. These architectures include Region-based CNN (R-CNN) [21], Fast R-CNN [42], Mask R-CNN [43], Single Shot Multibox Detector (SSD) [44] and Fully Convolutional Neural Network (FCN) [45]. Also by combining these architectures with RNN, Vinyals *et al.* [46] and Herdade *et al.* [47] extended CNNs to image captioning task. In addition to these aforementioned CNN architectures, there are many other novel architectures in the literature that have been proposed for many niche applications. We refer to Khan *et al.* [33] and Alzubaidi *et al.* [35] for comprehensive review of CNN architectures. The large diversity in CNN architectures truly suggests that architecture design of deep neural networks is indeed an important problem and a method to automate the process could be very useful.

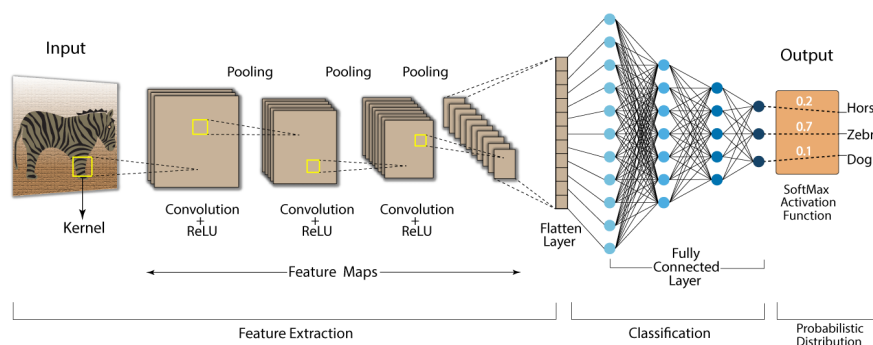


Figure 2.6: An example of a classical convolutional neural network architecture [48].

## 2.2 Metaheuristic Optimization Methods

Historically, metaheuristic algorithms have been employed for searching for neural network architectures of traditional ANNs, a comprehensive review has been presented in Ojha *et al.*

[49]. In recent years, a lot of work has been geared towards automatic architecture search of DNNs, particularly CNN architectures, due to their outstanding performance in computer vision and natural language processing [50, 51]. A large variety of metaheuristic algorithms have been adopted for automatic CNNs architecture search [52, 53], which include random search [54, 55], evolutionary algorithms [56, 57] and particle swarm optimization [58, 59].

Metaheuristic is a term coined by Glover in 1986 [60] which refers to approximate general-purpose search and optimization methods which are used to find a/an (near)-optimal solution in search space of candidate solutions. Metaheuristics are well known as efficient methods for combinatorial optimization problems with properties such as high dimensionality, parameter interaction, multimodality and non-differentiability which are difficult to be solved by exact optimization methods with a reasonable effort. Metaheuristics use problem-invariant and widely applicable search strategy with intensification (exploitation) and diversification (exploration) mechanisms for searching through a search space to find a sufficiently good solution with less computational effort [61, 62].

Usually, metaheuristic algorithms are often considered as “black-box” which one can plug-in arbitrary *search operators*, *representation* and *objective function* as needed to solve a particular complex problem. Metaheuristic algorithms can be categorized into two groups: a single-solution based and population based algorithms. Single-solution based metaheuristic algorithms improve a single solution, whereas population-based metaheuristic algorithms improve a number of solutions iteratively until the termination condition is satisfied. Random search, tabu search and iterated local search are examples of single-solution based; and evolutionary algorithms and swarm-based algorithms are examples of population based [62, 63].

Here, we briefly describe random search and evolutionary algorithm which have been employed in this work to search for CNN architectures automatically. And refer to Rothlauf *et al.* [61] and Weise *et al.* [63] for further information and implementation details of metaheuristic algorithms.

### 2.2.1 Random Search

Random search, also called random sampling algorithm, refers to an algorithm that repeats creating random solutions from across the entire search space until the termination criterion is satisfied [64]. It is the simplest hyperparameter optimization method [65] and can find better models by effectively searching a larger, less promising search space [66]. Bergstra *et al.* [66] have shown that random search is able to find models that are good or better within a small fraction of the computation time compare with grid search.

In random search, the representation of the search space  $\mathcal{X}$  can be any data structure that is suitable for the candidate solutions to a given problem and able to apply the objective function

to determine the quality of the solutions. The pure random search [67] typically uses uniform sampling distribution as search operator to generate feasible candidate solution for a given problem. It has been shown that pure random search can converge to a solution within a distance of the global optimum with probability one [68, 69]. Other variants of random search algorithm include fixed step size random search [70], adaptive step size random search [71] and optimized relative step size random search [72]. A pure random search algorithm is briefly describes in Algorithm 2 [63].

---

**Algorithm 2** A description of random search algorithm.

---

- 1 Initialize the best objective value  $b$  to infinity.
  - 2 Generate a random point (uniform sampling)  $x$  in search space  $\mathcal{X}$ .
  - 3 Map the generated point  $x$  to a candidate solution  $y$  using the representation mapping function  $y = \gamma(x)$ .
  - 4 Compute objective value of  $y$  using the objective function  $z = f(y)$ .
  - 5 If  $z$  is better than best objective value  $b$ , then:
    - i. Set the value of  $b$  to  $z$ .
    - ii. Store the candidate solution  $y$  for future retrieval.
  - 6 If termination criterion is not satisfied, return to step 2.
  - 7 Return best objective value  $b$  and best candidate solution  $y$ .
- 

## 2.2.2 Evolutionary Algorithm

There exists a number of evolutionary algorithms (EAs) in the literature [73, 74]. In this section, we briefly describe EAs and two classical EAs, genetic algorithm (GA) [75] and genetic programming (GP) [76], that are relevant background of the work in this dissertation.

EAs are subclass of evolutionary computation which are inspired by biological evolution mechanisms (such as mutation, recombination and selection) as search operators that are applied over individuals in a population to find the best solution for a given problem [77]. The model or abstraction of biological evolution mechanisms is based on the Darwinian theory of evolution [78]. As inspired by nature, the terms such as genotype and phenotype are also commonly used in EAs. The genotype (also called chromosome) encodes all the genetic information that describe an individual solution to a given problem and phenotype represents an instance of the candidate solution. The phenotypic appearance of a solution determines its quality, therefore the quality of different candidate solutions is compared in phenotype space. In other words, the genetic modification mechanisms are applied in genotype space to create new candidate solutions. Then, the candidate solutions are evaluated in the phenotype space to determine their quality with respect to the objective function of a given problem. The success of an EA largely depends on the choice of genotype-phenotype representation and the choice of genetic variation mechanisms (search operators) [61, 77, 79].

In EAs, we randomly generate an initial population of candidate solutions to a given problem encoded in certain representation. Then an objective function, also called fitness function, is applied to determine their quality. For next generation, some candidate solutions are selected based on their fitness value for reproduction. This is achieved via genetic modification mechanisms: recombination (crossover) and mutation. Crossover is applied to two selected candidate solutions known as parents to generate a new candidate solution (child), whereas mutation is applied to only one candidate solution to generate a new child. Then, the generated children, that is the new set of candidate solutions, are evaluated to determine their fitness values. This is an iterative process and it continues until a termination criterion is satisfied or a sufficiently good solution is found [61, 77]. Figure 2.7 is a generic framework of evolutionary algorithm. EAs are often simple at a high-level as they do not implement knowledge specific to the problem as in heuristic search [61], but become increasingly complex as partial domain knowledge are encoded into the system to guide the search process [74].

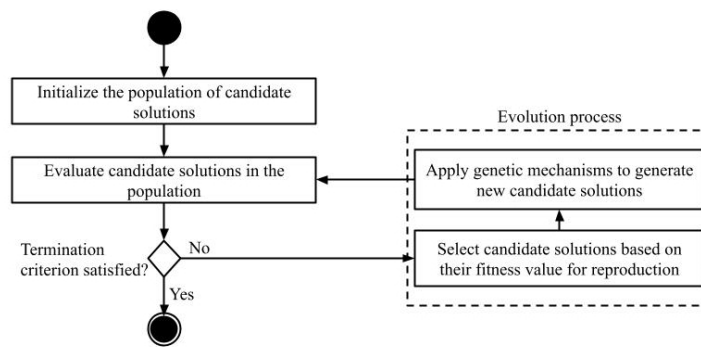


Figure 2.7: A general framework of evolutionary algorithms.

## Genetic Algorithms

Genetic Algorithm (GA) [75] is an EA developed by Holland and his collaborators in the 1960s and 1970s [80, 81] for adaptive search and adaptive system design. It is the simplest and most common among the EAs, and it is adept at evolving rule-based systems. In GA, the candidate solutions are represented in the genotype space as chromosomes of fixed-length binary strings using direct encoding scheme [82, 83]. The direct encoding scheme encodes the candidate solutions to a problem into its “natural” problem space. Hence in GA, the genotype space and the phenotype space are clearly not separated as seen in nature. As an EA, GA apply various genetic modification mechanisms (such as crossover, mutation, and selection) as search operators via evolutionary process (see Figure 2.7) to search for the best solution to a given problem. Because of the simple bit-string representation of the chromosomes in GA, the genetic modification operations are easily performed, and any variation made in the genotype space always results in syntactically correct chromosomes.

## Genetic Programming

Genetic Programming (GP) [84] is an EA introduced in 1992 by Koza [76] for evolving computer programs or mathematical equations that solve given problems. In contrast to GA, the candidate solutions in GP represents executable programs or mathematical expressions which are encoded as chromosomes of tree structures (called parse-trees) of varying shape and size. Traditionally, in GP, the chromosomes are represented by a LISP expression. The individuals that encode possible candidate solutions to a given problem are programs that, when executed, become the candidate solutions to the problem. Thus, GP evolves computer programs rather than binary strings (as in the case of GA) using various genetic modification mechanisms. Like GA, GP also uses direct encoding scheme to encode candidate solutions. However, due to the varying shape and size of the parse-trees (chromosomes) in GP, the genetic modification operations are difficult performed, the parse-trees need to be managed to ensure syntactical correctness in the genotype space [84–86].

## 2.3 Automatic Architecture Search of DNNs

There are many techniques and algorithms in the literature concerning the automatic architecture search of DNNs. Elsken *et al.* [51] categorized these approaches according to three dimensions: architecture search space, search optimization method, and performance evaluation strategy. This section briefly reviews related work of automatic architecture search of CNNs for visual perception tasks including image classification, object detection and semantic segmentation.

### 2.3.1 Architecture Search Space

The architecture search space defines the representation of candidate architectures which are to be searched to find the optimal architecture. Generally, the architecture search space of CNNs can be classified into two categories: the global search space which defines the whole network architecture, that is the macroarchitecture, and the cell-based search space which defines building-blocks, also called cell or module, for building the macroarchitecture [50, 51]. Various methods of encoding the candidate architectures in the search space are mainly grouped as direct encodings and indirect encodings (also called generative encodings or developmental encoding) [83, 87]. A comprehensive review on encoding schemes for automated neural architecture search (NAS) have been presented in Fekiač *et al.* [88] Vargas-Hákim *et al.* [89]. The success of automated NAS method largely depends on the architecture representation (as the work in this dissertation will show later). Even small a change to the encoding of the architectures can make a significant difference in the performance of NAS method [90].

Most of the earlier studies in automated NAS methods for CNN architectures employed global

search space to search for the entire network architecture [91, 92]. The global search space approach needs high computation effort and architectures discovered by this approach are not flexible, and cannot be transferred easily to other tasks [93]. Zoph *et al.* [93] was the first to propose cell-based search space, NASNet, to search for cells (modules) and repeatedly stacked them to build a CNN architecture. NASNet optimizes two types of cells: a *normal cell* that preserves the spatial resolution of input volume and a *reduction cell* which is used to reduce the feature map spatial resolutions. Later, Zhong *et al.* [94] introduced similar cell-based approach but used max-pooling layer to reduce the feature map spatial resolution, and Liu *et al.* [95] used separable convolution layer to reduce the feature map spatial resolution. Besides the flexibility and transferability of CNN architectures discovered by the cell-based approach [93], they also perform better than the global-based ones [96].

Therefore, recent studies in automated NAS for CNNs in image classification tasks have adopted cell-based search space approach to represent candidate architectures [97–99]. Combining the global-based and the cell-based search space approaches have also been explored to evolve CNNs architectures for dense image prediction tasks such as semantic segmentation [100, 101] and object detection [102, 103].

### 2.3.2 Search Optimization Method

Automated NAS using evolutionary algorithms, a field now known as neuroevolution [104], has been of interest in the AI/ML community for three decades [57, 83]. Most earlier studies in automated NAS optimized both the network architectures and their connection weights at a small scale [83]. However, since CNNs have millions of connection weights, recent studies optimized only for the network architectures and learn their connection weights via back-propagation method [57]. The growing interest in automatic architecture search of CNNs in recent years, since 2017, has seen a phenomenal development of new automated NAS methods [50, 51]. Many parallel studies have employed various search optimization methods including evolutionary algorithms [97, 105, 106], reinforcement learning [91, 107, 108], gradient-based methods [99, 109, 110], random search [111, 112] and Bayesian optimization [113, 114] to evolve CNN architectures for visual perception tasks. Among these optimization methods, evolutionary algorithms and reinforcement learning the popular methods, and more recently gradient-based methods [50, 115].

Although, the sophisticated NAS optimization algorithms such as reinforcement learning, evolutionary algorithm and gradient-based methods have achieved remarkable results. A simple random search strategy, as highlighted in Yu *et al.* [116], is a competitive optimization method in finding the best CNNs architecture for a given tasks (as the work in this dissertation will show later as well). Li & Talwalkar *et al.* [117] have performed extensive experiments to show that random search strategy is sufficiently efficient for searching CNNs architectures for several datasets

in computer vision using less computation resources. And Xie *et al.* [112] have further explored the efficiency of random search using classical random graph models to generate CNNs architectures.

Evolutionary algorithm has demonstrated as more computationally feasible optimization method for automated NAS than their reinforcement learning based competitors [56, 91, 118]. As such, the work in this dissertation also optimize the architectures of CNN with evolutionary algorithm as well. Most of the evolutionary NAS studies have employed genetic algorithm and genetic programming to evolve CNN architectures for visual perception tasks [56, 57]. For example, Liu *et al.* [95] have proposed hierarchical encoding scheme in genetic algorithm and Sun *et al.* [119] introduced chromosomes of variable-length in genetic algorithm to evolve CNN architectures of variant shapes and sizes for image classification tasks. In Suganuma *et al.* [106], Cartesian genetic programming was adopted and Assunção *et al.* [120] combined genetic algorithm with grammatical evolution to evolve CNN architectures. Galván and Mooney [56] and Baldominos *et al.* [57] have provided comprehensive survey on recent evolutionary algorithms in automated NAS methods.

### 2.3.3 Performance Evaluation Strategy

In automated NAS methods, candidate architectures need to be evaluated to enable the optimization method to find the best performing architecture for given tasks. The pioneering work by Zoph & Le [91] trained each “child” candidate architecture from scratch until it converges and then measure its accuracy on the selected dataset. This naive approach demands many GPU resources and it is a computationally expensive process. To reduce the computation effort in order to speed up the search process, recent automated NAS studies have introduced various performance evaluation strategies to evaluate the quality of a candidate architecture [121, 122].

Proxy task approach, that is using smaller dataset and fewer epochs to train and evaluate, has been widely used as the evaluation strategy of candidate architecture [94, 97, 114, 123]. Although this approach has produced good empirical results, since the networks with fewer parameters may converge faster and produce better results for a few epochs than cumbersome ones, the networks with good performance may be abandoned [116]. However, the work in this dissertation adopts the proxy task approach to show that we can find good architectures even with limited computation resources. Other studies have employed parameter sharing techniques, where network weights are shared amongst the candidate architectures [96, 99, 110]. Bender *et al.* [124] have extended the idea of weight sharing into one-shot architecture search. Moreover, network morphism has also been utilized to warm-start the candidate architectures, in order to reduces the number of training epochs for networks to convergence [108, 125, 126].

## 2.4 Remote Sensing Image Understanding with CNNs

Neural networks, traditional ANNs, were introduced in remote sensing (RS) in the late 1980s for analysis of multispectral remotely-sensed data [127]. However, due to limited training data, the RS research community shifted focus from ANNs to statistical learning methods such as support vector machines [128] and random forest [129] which perform well with few training samples. The recent advent of massive volumes of high resolution RS imagery data from various advanced earth observation technologies [130] has necessitated researchers to adopt CNN models which can automatically learn hierarchical and insightful features from large volumes of imagery data, and generalize well than the statistical learning models [131]. This section briefly reviews related work of RS image understanding tasks, in particular, scene classification and semantic segmentation tasks using CNNs.

### 2.4.1 RS Image Scene Classification with CNNs

Since 2014, CNN models that are pre-trained on general-purpose imagery datasets, especially ImageNet [32], have been employed in RS image scene classification using transfer learning approach and have achieved promising results [132, 133]. Most of the state-of-the-art works in single-label scene classification adopted pre-trained CNN models, including AlexNet [20], CaffeNet [134], VGGNet [36], GoogLeNet (Inception) [39] and ResNet [38], as feature extractors [135–137], or fine-tuned them on single-label RS image datasets [138, 139]. In multi-label RS image scene classification, Zeggada *et al.* [140] and Koda *et al.* [141] applied radial basis functions and support vector machine respectively on CNN features. Whereas Alshehri *et al.* [142] and Hua *et al.* [143] integrated attention-based RNNs into pre-trained CNN feature extractor for multi-label scene classification. Sumbul & DemİR [144] departed from pre-trained CNN models by proposing spatial resolution specific CNN branches with multi-attention strategy to classify multi-label RS image scenes. Moreover, Liu and Huang [145] proposed a weakly supervised triplet network using weakly single-label data to train CNN from the scratch. Furthermore, in Stivaktakis *et al.* [146], data augmentation strategy was introduced to train shallow CNNs on a multi-label dataset.

### 2.4.2 RS Image Segmentation with CNNs

CNNs have also emerged as the leading method in semantic segmentation of RS images (also known as land cover classification or pixel-wise classification). The first successful use of a patch-based CNNs by Mnih *et al.* [147] for roads and buildings segmentation saw other studies built upon it, by combining the patch-based CNNs with pre-segmentation and handcrafted features improve the results [148, 149]. The fully convolution network (FCN) [45] and the encoder-decoder networks such



as U-Net [150] and SegNet [151] which can directly perform pixel-wise classification without any pre-processing such as pre-segmentation have also been adopted in aerial or satellite image semantic segmentation [152–157]. Some of the previous studies have also adopted pre-trained networks such as ResNet [38], DenseNet [40] and Xception [158] with dense conditional random fields (CRFs) [159] or atrous spatial pyramid pooling (ASPP) [160] to extract multi-scale context features for aerial image semantic segmentation, and these methods have achieved significant performance [161–163].

### 2.4.3 AutoDNN in RS Image Understanding

A few works have explored automated NAS for RS image understanding tasks. Chen *et al.* [164] employed the architecture search space that was introduced in DARTS [99] with a gradient-based optimization algorithm to find CNN architectures for hyperspectral image classification. Also, using the search space of DARTS with gradient-based algorithm, Zhang *et al.* [165] and Jing *et al.* [166], evolved encoder-decoder CNN architectures for semantic segmentation of high-resolution aerial images. Finally, a simple random search strategy with cell-based search space has been used to search for CNN architectures for RS image scene classification [167]. The work in this dissertation employ random search and evolutionary algorithms with cell-based search space to find well performing CNN architectures for both single- and multi-label RS scene classification and semantic segmentation of RS images.

## 2.5 Summary

This chapter provided a brief overview of the foundations and related work for automated neural architecture search. The basic concepts of DNNs and the main components of CNNs, as well as, the two metaheuristics methods, random search and evolutionary algorithms, that are adopted for network architecture optimization were briefly discussed. The chapter reviewed related work in the context of architecture search space, search strategy and performance evaluation strategy for automated neural architecture search. Moreover, previous studies that have adopted CNNs for RS image understanding tasks, scene classification and semantic segmentation, were reviewed. The work in this dissertation draws inspiration from a lot of the topics presented in this chapter to develop a novel automated neural architecture search method for RS image understanding as presented in the later chapters.

## 2.6 References

- [1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

- 
- [2] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.
- [3] Y. Bengio, Y. Lecun, and G. Hinton, "Deep learning for ai," *Communications of the ACM*, vol. 64, no. 7, p. 58–65, 2021.
- [4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [5] D. O. Hebb, "Animal and Physiological Psychology," *Annual Review of Psychology*, vol. 1, no. 1, pp. 173–188, 1950.
- [6] S. Haykin, *Neural Networks and Learning Machines, 3/E*. Pearson Education India, 2010.
- [7] K. Gurney, *An Introduction to Neural Networks*. USA: Taylor & Francis, Inc., 1997.
- [8] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.
- [9] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.
- [10] J. B. Ahire, "The Artificial Neural Networks Handbook: Part 4 - DZone AI," Date Accessed: 2021-09-19. [Online]. Available: <https://dzone.com/articles/the-artificial-neural-networks-handbook-part-4>
- [11] H. N. Mhaskar and C. A. Micchelli, "How to Choose an Activation Function," in *Proceeding of Advances in Neural Information Processing Systems*, vol. 6, 1994.
- [12] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.
- [13] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.
- [14] B. Hanin, "Which neural net architectures give rise to exploding and vanishing gradients?" in *NeurIPS*, 2018, pp. 580–589.
- [15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [17] Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski, "A theoretical framework for back-propagation," in *Proceedings of the 1988 connectionist models summer school*, vol. 1, 1988, pp. 21–28.
- [18] S. E. Kim and I. W. Seo, "Artificial Neural Network ensemble modeling with conjunctive data clustering for water quality prediction in rivers," *Journal of Hydro-environment Research*, vol. 9, no. 3, pp. 325–339, 2015.
- [19] M. Egmont-Petersen, D. D. Ridder, and H. Handels, "Image processing with neural networks - a review," *Pattern Recognition*, vol. 35, pp. 2279–2301, 2002.
- [20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.
- [21] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
- [22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

- [23] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3156–3164, 2015.
- [24] K. Fukushima, "Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.
- [25] K. Fukushima, "Artificial vision by deep cnn neocognitron," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 76–90, 2021.
- [26] A. Adeel, M. Gogate, and A. Hussain, "Contextual deep learning-based audio-visual switching for speech enhancement in real-world environments," *Information Fusion*, vol. 59, pp. 163–170, 2020.
- [27] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.
- [28] G. Koppe, A. Meyer-Lindenberg, and D. Durstewitz, "Deep learning for small and big data in psychiatry," *Neuropsychopharmacology*, vol. 46, no. 1, pp. 176–190, 2021.
- [29] A. Al-Azzawi, A. Ouadou, H. Max, Y. Duan, J. J. Tanner, and J. Cheng, "Deepcryopicker: fully automated deep neural network for single protein particle picking in cryo-em," *BMC Bioinformatics*, vol. 21, no. 1, pp. 1–38, 2020.
- [30] "CS231n Convolutional Neural Networks for Visual Recognition," Date Accessed: 2021-09-20. [Online]. Available: <https://cs231n.github.io/convolutional-networks/>
- [31] J. Jordan, "Convolutional neural networks." Date Accessed: 2021-09-20. [Online]. Available: <https://www.jeremyjordan.me/convolutional-neural-networks/>
- [32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [33] A. Khan, A. Sohail, U. Zahoor, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5455–5516, 2020.
- [34] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53 040–53 065, 2019.
- [35] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, p. 53, 2021.
- [36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.
- [37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.
- [38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [39] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, p. 4278–4284.

- [40] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.
- [41] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, p. 3859–3869.
- [42] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
- [43] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.
- [44] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single Shot MultiBox Detector,” in *Computer Vision – ECCV 2016*, 2016, pp. 21–37.
- [45] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.
- [46] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: Lessons learned from the 2015 mscoco image captioning challenge,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, p. 652–663, 2017.
- [47] S. Herdade, A. Kappeler, K. Boakye, and J. Soares, “Image Captioning: Transforming Objects into Words,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [48] “Convolutional Neural Network | Deep Learning | Developers Breach,” Date Accessed: 2021-09-21. [Online]. Available: <https://developersbreach.com/convolution-neural-network-deep-learning/>
- [49] V. K. Ojha, A. Abraham, and V. Snášel, “Metaheuristic design of feedforward neural networks: A review of two decades of research,” *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 97–116, 2017.
- [50] M. Wistuba, A. Rawat, and T. Pedapati, “A survey on neural architecture search,” *arXiv:1905.01392*, 2019.
- [51] T. Elsken, J. H. Metzen, and F. Hutter, “Neural architecture search: A survey,” *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.
- [52] E.-G. Talbi, “Automated design of deep neural networks: A survey and unified taxonomy,” *ACM Comput. Surv.*, vol. 54, no. 2, 2021.
- [53] A. Gaspar, D. Oliva, E. Cuevas, D. Zaldívar, M. Pérez, and G. Pajares, “Hyperparameter optimization in a convolutional neural network using metaheuristic algorithms,” in *Metaheuristics in Machine Learning: Theory and Applications*, 2021, pp. 37–59.
- [54] L. Li and A. Talwalkar, “Random search and reproducibility for neural architecture search,” in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. UAI, 2019.
- [55] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann, “Evaluating the search phase of neural architecture search,” in *Proceedings of the International Conference on Learning Representations*, 2020.
- [56] E. Galvan and P. Mooney, “Neuroevolution in deep neural networks: Current trends and future challenges,” *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 01, pp. 1–1, 2021.
- [57] A. Baldominos, Y. Saez, and P. Isasi, “On the automated, evolutionary design of neural networks: past, present, and future,” *Neural Computing and Applications*, vol. 32, no. 2, pp. 519–545, 2020.

- [58] J. Jiang, F. Han, Q. Ling, J. Wang, T. Li, and H. Han, "Efficient network architecture search via multiobjective particle swarm optimization based on decomposition," *Neural Networks*, vol. 123, no. C, p. 305–316, 2020.
- [59] J. Huang, B. Xue, Y. Sun, and M. Zhang, "A flexible variable-length particle swarm optimization approach to convolutional neural network architecture design," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, 2021, pp. 934–941.
- [60] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & operations research*, vol. 13, no. 5, pp. 533–549, 1986.
- [61] F. Rothlauf, *Design of Modern Heuristics: Principles and Application*, 1st ed. Springer, Inc., 2011.
- [62] K.-L. Du and M. N. S. Swamy, *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature*, 1st ed. Birkhäuser Basel, 2016.
- [63] T. Weise, "An Introduction to Optimization Algorithms," Date Accessed: 2021-09-22. [Online]. Available: <https://thomasweise.github.io/aitoa/aitoa.html>
- [64] R. H. Storer, S. D. Wu, and R. Vaccari, "New search spaces for sequencing problems with application to job shop scheduling," *Management Science*, vol. 38, no. 10, p. 1495–1509, 1992.
- [65] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.
- [66] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.
- [67] S. H. Brooks, "A discussion of random methods for seeking maxima," *Operations Research*, vol. 6, no. 2, pp. 244–251, 1958.
- [68] F. J. Solis and R. J. B. Wets, "Minimization by random search techniques," *Mathematics of Operations Research*, vol. 6, no. 1, p. 19–30, 1981.
- [69] Z. B. Zabinsky, *Random Search Algorithms*. American Cancer Society, 2011.
- [70] L. A. Rastrigin, "The convergence of the random search method in the extremal control of many-parameter system," *Automation and Remote Control*, vol. 24, no. 10, pp. 1337–1342, 1963.
- [71] M. Schumer and K. Steiglitz, "Adaptive step size random search," *IEEE Transactions on Automatic Control*, vol. 13, no. 3, pp. 270–276, 1968.
- [72] G. Schrack and M. Choit, "Optimized relative step size random searches," *Mathematical Programming*, vol. 10, no. 1, pp. 230–244, 1976.
- [73] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, 2016, pp. 261–265.
- [74] A. N. Sloss and S. Gustafson, "2019 evolutionary algorithms review," *Genetic Programming Theory and Practice XVII*, p. 307, 2020.
- [75] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Addison-Wesley Longman Publishing Co., Inc., 1989.
- [76] J. Koza and R. John, *Genetic programming*. MIT Press, 1992.
- [77] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed. Springer Publishing Company, Incorporated, 2015.
- [78] C. Darwin, *The origin of species*. PF Collier & son New York, 1909.
- [79] F. Rothlauf and D. E. Goldberg, *Representations for Genetic and Evolutionary Algorithms*. Physica-Verlag, 2002.

- 
- [80] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975.
- [81] K. A. De Jong, “An analysis of the behavior of a class of genetic adaptive systems.” Ph.D. dissertation, USA, 1975.
- [82] A. Siddiqi and S. Lucas, “A comparison of matrix rewriting versus direct encoding for evolving neural networks,” in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, 1998, pp. 392–397.
- [83] X. Yao, “Evolving artificial neural networks,” *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.
- [84] W. Banzhaf, F. D. Francone, R. E. Keller, and P. Nordin, *Genetic Programming: An Introduction*. San Francisco: Morgan Kaufmann, 1998.
- [85] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd, 2008.
- [86] L. Vanneschi and R. Poli, “Genetic programming - introduction, applications, theory and open issues,” in *Handbook of Natural Computing*, 2012.
- [87] J. Clune, “Evolving artificial neural networks with generative encodings inspired by developmental biology,” Doctoral Dissertation, Michigan State University, 2010.
- [88] J. Fekiač, I. Zelinka, and J. C. Burguillo, “A review of methods for encoding neural network topologies in evolutionary computation,” in *Proceedings 25th ECMS*, 2011, pp. 410–416.
- [89] G. A. Vargas-Hákim, E. Mezura-Montes, and H.-G. Acosta-Mesa, “A review on convolutional neural networks encodings for neuroevolution,” *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2021.
- [90] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, “Nas-bench-101: Towards reproducible neural architecture search,” in *International Conference on Machine Learning*, 2019, pp. 7105–7114.
- [91] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” in *Proceedings of 5th ICLR*, 2017.
- [92] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, “Large-scale evolution of image classifiers,” *Journal of Machine Learning Research (JMLR)*, vol. 70, pp. 2902–2911, 2017.
- [93] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in *IEEE Conference on CVPR*, 2018, pp. 8697–8710.
- [94] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, “Practical block-wise neural network architecture generation,” in *IEEE Conference on CVPR*, 2018, pp. 2423–2432.
- [95] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, “Hierarchical representations for efficient architecture search,” in *Proceedings of Sixth ICLR*, 2018.
- [96] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, “Efficient neural architecture search via parameters sharing,” in *Proceedings of the 35th ICML*, vol. 80, 2018, p. 4095–4104.
- [97] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” in *Proceedings of the AAAI conference on Artificial Intelligence*, vol. 33, 2019, pp. 4780–4789.
- [98] H. Cai, L. Zhu, and S. Han, “ProxylessNAS: Direct neural architecture search on target task and hardware,” in *Proceedings of the International Conference on Learning Representations*. ICLR, 2019.

- [99] H. Liu, K. Simonyan, and Y. Yang, “DARTS: Differentiable architecture search,” in *Proceedings of the International Conference on Learning Representations*. ICLR, 2019.
- [100] Y. Zhang, Z. Qiu, J. Liu, T. Yao, D. Liu, and T. Mei, “Customizable architecture search for semantic segmentation,” in *2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [101] Q. Yu, D. Yang, H. Roth, Y. Bai, Y. Zhang, A. L. Yuille, and D. Xu, “C2FNAS: Coarse-to-fine neural architecture search for 3d medical image segmentation,” in *2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4125–4134.
- [102] Z. Li, T. Xi, G. Zhang, J. Liu, and R. He, “AutoDet: Pyramid Network Architecture Search for Object Detection,” *International Journal of Computer Vision*, 2021.
- [103] G. Ghiasi, T.-Y. Lin, and Q. V. Le, “NAS-FPN: Learning scalable feature pyramid architecture for object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7036–7045.
- [104] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, “Designing neural networks through neuroevolution,” *Nature Machine Intelligence*, vol. 1, no. 1, pp. 24–35, 2019.
- [105] L. Xie and A. Yuille, “Genetic CNN,” in *Proceedings of the IEEE ICCV*, 2017, pp. 1388–1397.
- [106] M. Suganuma, S. Shirakawa, and T. Nagao, “A genetic programming approach to designing convolutional neural network architectures,” in *Proceedings of the Twenty-Seventh IJCAI*, 2018, pp. 5369–5373.
- [107] B. Baker, O. Gupta, N. Naik, and R. Raskar, “Designing neural network architectures using reinforcement learning,” in *Proceedings of the 5th International Conference on Learning Representations*, 2017.
- [108] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, “Efficient architecture search by network transformation,” in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, p. 2787–2794.
- [109] K. Ahmed and L. Torresani, “Maskconnect: Connectivity learning by gradient descent,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.
- [110] S. Xie, H. Zheng, C. Liu, and L. Lin, “SNAS: stochastic neural architecture search,” in *Proceedings of the International Conference on Learning Representations*, 2019.
- [111] L. Li and A. Talwalkar, “Random search and reproducibility for neural architecture search,” in *Proceedings of Conf. on UAI*, 2019.
- [112] S. Xie, A. Kirillov, R. Girshick, and K. He, “Exploring randomly wired neural networks for image recognition,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.
- [113] K. Kandasamy, W. Neiswanger, J. Schneider, B. Póczos, and E. P. Xing, “Neural architecture search with bayesian optimisation and optimal transport,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, p. 2020–2029.
- [114] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, “Progressive neural architecture search,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [115] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, “A comprehensive survey of neural architecture search: Challenges and solutions,” *ACM Comp. Surveys*, vol. 54, no. 4, pp. 1–34, 2021.
- [116] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann, “Evaluating the search phase of neural architecture search,” in *Proceedings of ICLR*, 2020.
- [117] L. Li and A. Talwalkar, “Random search and reproducibility for neural architecture search,” in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. UAI, 2019.

- [118] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, “Completely automated cnn architecture design based on blocks,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2019.
- [119] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, “Evolving deep convolutional neural networks for image classification,” *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 394–407, 2019.
- [120] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro, “DENSER: Deep evolutionary network structured representation,” *Genetic Prog. and Evolvable Machines*, vol. 20, no. 1, pp. 5–35, 2019.
- [121] L. Weng, “Neural Architecture Search.” [Online]. Available: <https://lilianweng.github.io/lil-log/2020/08/06/neural-architecture-search.html#{#}evaluation-strategy>
- [122] G. Kyriakides and K. Margaritis, “An introduction to neural architecture search for convolutional networks,” 2020, arXiv:abs/2005.11074.
- [123] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat, “Evolving Deep Neural Networks,” in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, 2019, pp. 293–312.
- [124] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, “Understanding and simplifying one-shot architecture search,” in *International Conference on Machine Learning*, 2018, pp. 550–559.
- [125] T. Elsken, J.-H. Metzen, and F. Hutter, “Simple and efficient architecture search for convolutional neural networks,” 2017, arXiv preprint arXiv:1711.04528.
- [126] H. Jin, Q. Song, and X. Hu, “Auto-keras: An efficient neural architecture search system,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1946–1956.
- [127] J. Gao, *Digital Analysis of Remotely Sensed Imagery*, 1st ed. McGraw-Hill Professional, 2008.
- [128] G. Mountrakis, J. Im, and C. Ogole, “Support vector machines in remote sensing: A review,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 66, no. 3, pp. 247–259, 2011.
- [129] M. Belgiu and L. Drăguț, “Random forest in remote sensing: A review of applications and future directions,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 114, pp. 24–31, 2016.
- [130] S. S. Durbha, K. R. Kurte, and U. Bhangale, “Semantics and High Performance Computing Driven Approaches for Enhanced Exploitation of Earth Observation (EO) Data: State of the Art,” *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, vol. 87, no. 4, pp. 519–539, 2017.
- [131] J. Song, S. Gao, Y. Zhu, and C. Ma, “A survey of remote sensing image classification based on CNNs,” *Big Earth Data*, vol. 3, no. 3, pp. 232–254, 2019.
- [132] X. X. Zhu, D. Tuia, L. Mou, G. Xia, L. Zhang, F. Xu, and F. Fraundorfer, “Deep learning in remote sensing: A comprehensive review and list of resources,” *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 8–36, 2017.
- [133] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, “Deep learning in remote sensing applications: A meta-analysis and review,” *ISPRS Journal of Photo. and RS*, vol. 152, pp. 166–177, 2019.
- [134] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the 22nd ACM International Conference on Multimedia*, 2014, p. 675–678.



- [135] D. Zeng, S. Chen, B. Chen, and S. Li, “Improving remote sensing scene classification by integrating global-context and local-object features,” *Remote Sensing*, vol. 10, no. 5, 2018.
- [136] Y. Guo, J. Ji, X. Lu, H. Huo, T. Fang, and D. Li, “Global-local attention network for aerial scene classification,” *IEEE Access*, vol. 7, pp. 67 200–67 212, 2019.
- [137] R. Dong, D. Xu, L. Jiao, J. Zhao, and J. An, “A fast deep perception network for remote sensing scene classification,” *Remote Sensing*, vol. 12, no. 4, p. 729, 2020.
- [138] K. Nogueira, O. A. Penatti, and J. A. dos Santos, “Towards better exploiting convolutional neural networks for remote sensing scene classification,” *Pattern Recognition*, vol. 61, p. 539–556, 2017.
- [139] S. Jiang, H. Zhao, W. Wu, and Q. Tan, “A novel framework for remote sensing image scene classification.” *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 42, no. 3, 2018.
- [140] A. Zeggada, F. Melgani, and Y. Bazi, “A deep learning approach to uav image multilabeling,” *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 5, pp. 694–698, 2017.
- [141] S. Koda, A. Zeggada, F. Melgani, and R. Nishii, “Spatial and structured svm for multilabel image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 10, pp. 5948–5960, 2018.
- [142] A. Alshehri, Y. Bazi, N. Ammour, H. Almubarak, and N. Alajlan, “Deep attention neural network for multi-label classification in unmanned aerial vehicle imagery,” *IEEE Access*, vol. 7, pp. 119 873–119 880, 2019.
- [143] Y. Hua, L. Mou, and X. X. Zhu, “Relation network for multilabel aerial image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 7, pp. 4558–4572, 2020.
- [144] G. Sumbul and B. Demir, “A deep multi-attention driven approach for multi-label remote sensing image classification,” *IEEE Access*, vol. 8, pp. 95 934–95 946, 2020.
- [145] Y. Liu and C. Huang, “Scene classification via triplet networks,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 1, pp. 220–237, 2017.
- [146] R. Stivaktakis, G. Tsagkatakis, and P. Tsakalides, “Deep learning for multilabel land cover scene categorization using data augmentation,” *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 7, pp. 1031–1035, 2019.
- [147] V. Mnih and G. E. Hinton, “Learning to Detect Roads in High-Resolution Aerial Images,” in *Computer Vision – ECCV 2010*, 2010, pp. 210–223.
- [148] A. . Vo, L. Truong-Hong, D. F. Laefer, D. Tiede, S. d’Oleire-Oltmanns, A. Baraldi, M. Shimoni, G. Moser, and D. Tuia, “Processing of extremely high resolution lidar and rgb data: Outcome of the 2015 ieee grss data fusion contest—part b: 3-d contest,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 12, pp. 5560–5575, 2016.
- [149] S. Paisitkriangkrai, J. Sherrah, P. Janney, V.-D. Hengel *et al.*, “Effective semantic pixel labelling with convolutional networks and conditional random fields,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 36–43.
- [150] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*, 2015, pp. 234–241.
- [151] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

- [152] M. Volpi and D. Tuia, "Dense semantic labeling of subdecimeter resolution images with convolutional neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 2, pp. 881–893, 2017.
- [153] G. Chen, C. Li, W. Wei, W. Jing, M. Woźniak, T. Blažauskas, and R. Damaševičius, "Fully Convolutional Neural Network with Augmented Atrous Spatial Pyramid Pool and Fully Connected Fusion Path for High Resolution Remote Sensing Image Segmentation," *Applied Sciences*, vol. 9, no. 9, 2019.
- [154] D. Marmanis, K. Schindler, J. D. Wegner, S. Galliani, M. Datcu, and U. Stilla, "Classification with an edge: Improving semantic image segmentation with boundary detection," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 135, pp. 158–172, 2018.
- [155] N. Audebert, B. Le Saux, and S. Lefèvre, "Beyond RGB: Very high resolution urban remote sensing with multimodal deep networks," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 140, pp. 20–32, 2018.
- [156] F. I. Diakogiannis, F. Waldner, P. Caccetta, and C. Wu, "ResUNet-a: A deep learning framework for semantic segmentation of remotely sensed data," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 162, pp. 94–114, 2020.
- [157] R. Dong, L. Bai, and F. Li, "SiameseDenseU-Net-based Semantic Segmentation of Urban Remote Sensing Images," *Mathematical Problems in Engineering*, vol. 2020, p. 1515630, 2020.
- [158] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.
- [159] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," 2014, arXiv preprint arXiv:1412.7062.
- [160] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.
- [161] Y. Liu, S. Piramanayagam, S. T. Monteiro, and E. Saber, "Dense semantic labeling of very-high-resolution aerial imagery and lidar with fully-convolutional neural networks and higher-order CRFs," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 1561–1570.
- [162] Z. Li, R. Wang, W. Zhang, F. Hu, and L. Meng, "Multiscale features supported deeplabv3+ optimization scheme for accurate water semantic segmentation," *IEEE Access*, vol. 7, pp. 155 787–155 804, 2019.
- [163] J. Liu, Z. Wang, and K. Cheng, "An improved algorithm for semantic segmentation of remote sensing images based on deeplabv3+," in *Proceedings of the 5th International Conference on Communication and Information Processing*, 2019, p. 124–128.
- [164] Y. Chen, K. Zhu, L. Zhu, X. He, P. Ghamisi, and J. A. Benediktsson, "Automatic design of convolutional neural network for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 9, pp. 7048–7066, 2019.
- [165] M. Zhang, W. Jing, J. Lin, N. Fang, W. Wei, M. Woźniak, and R. Damaševičius, "NAS-HRIS: Automatic Design and Architecture Search of Neural Network for Semantic Segmentation in Remote Sensing Images," *Sensors*, vol. 20, no. 18, p. 5292, 2020.
- [166] W. Jing, J. Lin, and H. Wang, "Building NAS: Automatic designation of efficient neural architectures for building extraction in high-resolution aerial images," *Image and Vision Computing*, vol. 103, p. 104025, 2020.

- [167] J. Li, W. Diao, X. Sun, Y. Feng, W. Zhang, Z. Chang, and K. Fu, “Automated and lightweight network design via random search for remote sensing image scene classification,” *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLIII-B2-2020, pp. 1217–1224, 2020.

## Chapter 3

# Symbolic Linear Generative Encoding

*The key to artificial intelligence has always been the representation.*

—**Jeff Hawkins**

(Founder of Palm, Inc.)

*This use of building blocks to generate internal models is a pervasive feature of complex adaptive systems.*

—**John H. Holland**

(1929–2015)

This chapter presents a new generative encoding, *symbolic linear generative encoding* (SLGE), we introduce for automatic architecture search of DNNs. SLGE combines elements of two generative encodings, *gene expression programming* (GEP) and *cellular encoding* (CE), that were developed based on the foundations of evolutionary algorithms to evolve modularized CNN architectures. Section 3.1 describes SLGE background and motivation, then in section 3.2, SLGE is described in detail, in particular, the encoding and the mapping in the architecture search space. In addition, preliminary experimental results in general-purpose image classification domain on the CIFAR-10 and CIFAR-100 benchmarks are presented in section 3.3. The preliminary experiments aimed at verifying the effectiveness and viability of SLGE for evolving building-blocks to develop modularized CNN architectures. Finally, section 3.4 describes the application of SLGE to remote sensing image understanding.

### 3.1 SLGE Background and Motivation

In this section, we provide a brief introduction of gene expression programming (GEP) and cellular encoding (CE) as background knowledge to understand the symbolic linear generative encoding

---

Parts of this chapter were previously published in Broni-Bediako *et al.*, 2020 (see Appendix B) and Broni-Bediako *et al.*, 2021 (see Appendix D). Reprinted with permission from the publishers.

(SLGE) proposed in this dissertation. Then, the motivation for combining GEP with CE to develop SLGE for automatic architecture search of CNNs is explained.

### 3.1.1 Gene Expression Programming

GEP is an evolutionary algorithm which extends the genetic algorithm (GA) and genetic programming (GP) described in section 2.2.2. It was introduced by Ferreira in 2001 [1]. Like GP, GEP was developed to evolve computer program or mathematical expressions. However, GEP differs from GP with respect to the representation of its candidate solutions and the additional genetic modification mechanisms such as transposition and gene recombination that are used [2].

In GEP, the computer programs or mathematical expressions are represented as chromosomes of linear fixed-length strings similar to the ones in GA, and then expressed in the phenotype space as expression-trees of different shapes and sizes similar to the parse-trees in GP. Because the phenotypes in GEP are represented in tree-like structures, some literature in field of evolutionary computation describe GEP as a variant of GP [3, 4]. However, GEP can be seen as a generalization of both GA and GP techniques, since it combines the simplicity of GA and the complexity of GP. The main advantage of GEP over GA is that it can evolve complex tree-like structures as in GP with simple linear fixed-length strings, and in the case of GP, the effort of managing the tree-structures (parse-trees) to ensure syntactical correctness of the evolving programs is eliminated. Moreover, GEP explicitly separates genotype and phenotype spaces in analogue to nature using generative (developmental) encoding scheme. Therefore, with simple, linear compact chromosomes and a distinct separation of the genotype and phenotype spaces, GEP is more flexible and effective compared with GA and GP [1, 2]. GEP has well resolved a large variety of problems including symbolic regression [5], function optimization [6], time series analysis [7] and classification [8]. A comprehensive survey on GEP and its applications has provided in Zhong *et al.* [9].

The chromosomes of GEP are usually composed of several genes of equal length. A gene is structurally organized in a head and a tail format called Karva notation or K-expression with the head consists symbols that represent both functions and terminals, and the tail contains only terminals. The tail length  $t$  is expressed as:

$$t = h * (n - 1) + 1 \quad (3.1)$$

where  $h$  is the length of the gene head which is determined by the user for a given problem and  $n$  is the number of arguments of the function with most arguments (arity). The genes in a chromosome are translated into an expression-trees (phenotypes) according to the basic principles of GEP, and then converted to mathematical expression or program using a breadth first traversal method. The

fitness value of the corresponding chromosome is the evaluation of the mathematical expression or program [1]. Figure 3.1(a) is an example of a chromosome with only one gene of head length  $h = 4$ . And Figure 3.1(b) is an expression-tree of the chromosome presented in Figure 3.1(a), with which is then expressed mathematically as:

$$\sqrt{(a - c) * (3 + d)} \quad (3.2)$$

For a given problem, the number of genes for a chromosome, as well as the length of a gene head, are a priori selected. We refer to Ferreira (2006) [2] for further information and implementation details of GEP.

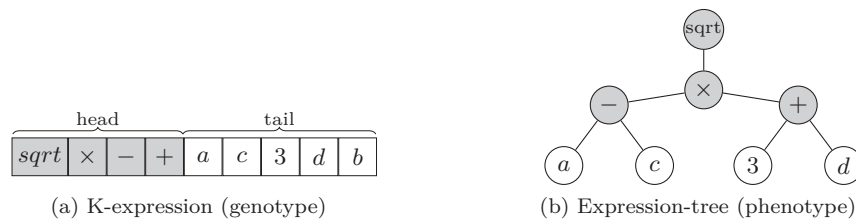


Figure 3.1: A genotype-phenotype representation in GEP: (a) is a GEP chromosome with only one gene of head length 4, and (b) is the expression-tree of the chromosome in (a) with which is expressed mathematically in Equation 3.2.

### 3.1.2 Cellular Encoding

CE was proposed by Gruau in his PhD thesis [10]. Inspired by the developmental process in a living organism, CE is a generative encoding scheme that was purposely developed for evolving ANNs. Thus, it is also known as neuron-centric encoding method. It uses graph grammar that control the division of nodes to encode ANNs. The graph grammar, also called local graph transformations, are represented as grammar-tree (called program) similar to the parse-tree in GP. The reader must not confused a *grammar-tree* that encoded as a tree, with a *tree-grammar* that rewrites trees. The labels of the grammar-tree represent the instructions for generating a neural network [11–13]. The basic instructions (graph grammar) are:

1. SEQuential division (*SEQ*): it splits current node into two and connects them in serial; the child node inherits the outputs of the parent node.
2. PARAllel division (*PAR*): it splits current node in two and connects them in parallel; both parent and child nodes share the same inputs and outputs.
3. CoPy Input division (*CPI*): it performs *SEQ*, then shares the same inputs with parent and child nodes.
4. CoPy Output division (*CPO*): it performs *SEQ*, then shares the same outputs with parent and child nodes.

5. RECURSIVE (*REC*): it recalls the instructions from the root.
6. END (*END*): it stops the developing process.

Figure 3.2(a) is an example of grammar-tree consists of (*SEQ*), (*PAR*), (*REC*) and (*END*) instructions. And Figure 3.2(b) shows the development of a neural network from the grammar-tree presented in Figure 3.2(a). For further information and implementation details, we refer to Gruau [10] and Gruau & Quatramaran [13].

CE adopts evolutionary process for evolving architectures of ANNs. In CE, the genotype is a grammar-tree with which is expressed as candidate network architecture (phenotype) via the sequence of local graph transformations. The genetic modification mechanisms are applied according to the common paradigm of GP [14] which limits its flexibility in crossover and mutation operations. The development of a neural network begins with a single initial *unit*<sup>1</sup> which has both input and output nodes, and grows into a whole ANN by executing the instructions in the grammar-tree. Then the weights of the generated network architectures are trained using back-propagation method. The grammar-tree in CE can be evolved to find modular structures for ANN architectures [15, 16]. CE has shown its efficiency on a wide range of problems, particularly in evolutionary robotics for evolving ANNs for controlling two poles on a cart and locomotion of a 6-legged robot [13, 17, 18].

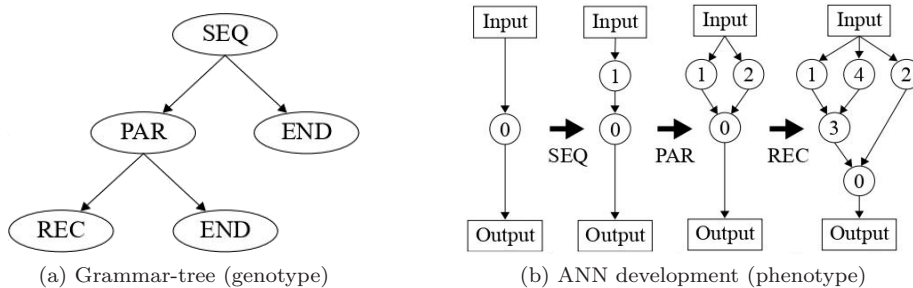


Figure 3.2: (a) An example of CE grammar-tree consists of four different instructions. The instructions are applied sequentially, based on CE procedure, from the initial node 0 and grows into the final feed-forward neural network in (b).

### 3.1.3 Combining Gene Expression Programming with Cellular Encoding

Both GEP and CE are evolutionary optimization methods, therefore they are metaheuristic methods as such. As metaheuristics, they are very desirable for problems where no gradient information is available such as neural architecture search. Unlike CE which was developed purposely for evolving ANNs, GEP was developed to evolve mathematical expressions or programs. However, Ferreira [19] has proposed that GEP could encode ANNs and evolve architectures of ANNs through evolutionary

<sup>1</sup>A unit is a single neuron in the original idea of CE, in this work it represents a convolutional layer of neurons

process. Wang *et al.* [20] have then used GEP to evolve ANNs for regression and classification tasks.

Given that GEP has emerged as a powerful EA due to its simplicity in implementation and multi-gene chromosomes with flexibility in genetic modification operations as compared to CE, and the capability for evolving architectures of ANNs, it is a promising starting point for CNNs architecture search. Although it is easy to implement and flexible in variation operations, GEP is not without weaknesses. The Karva notation of GEP lacks structure-preserving representation and does not allow hierarchical compositions, hence good evolved building-blocks or modules are very likely to be destroyed by genetic modifications in the subsequent generations [21]. Incorporating the graph grammar of CE into the linear fixed-length string of GEP could facilitate the development of building-blocks of different shapes and sizes for evolving architectures of neural networks. Thus, using chromosomes of simple fixed-length string which can be easily modified for architecture search as opposed to the complex tree-like structures in CE that are difficult to modify.

Furthermore, combining well designed generative encodings should produce compact chromosomes to evolve large-scale modular neural networks such as CNN architectures [22]. Also evolving architectures with modularity property is very important, as such property can improve the network training and enhance the network performance [23, 24]. GEP and CE have been largely employed in neural architecture search for artificial neural networks (ANNs) in a small scale [9, 17, 25], this work therefore provides the possibility for CNNs architecture development.

## 3.2 The Symbolic Linear Generative Encoding

The *symbolic linear generative encoding* (SLGE) is a novel generative encoding scheme which combines elements of two well established generative encodings, GEP and CE discussed in section 3.1.1 and section 3.1.2 respectively, to evolve modularized neural network architectures. SLGE harnesses the strengths of GEP and CE for automatic architecture search of DNNs, in particular CNN architectures, for visual perception tasks. By integrating the modularity features of CE into the linear representation of GEP to induce compact chromosomes which are able to evolve building-blocks of different shapes and sizes for construction of modularized CNN architectures.

SLGE embeds local graph transformations in linear fixed-length strings for the representation of candidate CNN architectures. It uses first-in-first-out (FIFO) queue strategy to design a genotype-phenotype mapping mechanism to translate the information encoded in the chromosomes into building-blocks for individual candidate CNN architectures. Besides the conventional genetic modification mechanisms (such as crossover, mutation, selection), SLGE also adopts some of the mechanisms in GEP, such as transposition and gene crossover, and follows the same fundamental evolutionary process as GEP to evolve candidate architectures. Like GEP, any genetic modification



operation in SLGE always results in syntactically correct phenotype, and it is easy to implement as well. With the inherited features of multi-gene and modularity, SLGE can evolve building-blocks which have multi-branch and shortcut (skip) connections [23, 26] similar to the ones commonly adopted by human experts. This is very important because such modularity can improve the network training and enhance the network performance [24, 27].

In SLGE, the cell-based search space [28] is designed to induce a wide range of building-blocks for architectures of CNNs. We implement SLGE on the top of *geppy*<sup>2</sup>, a GEP Python library which is built on top of an evolutionary algorithms framework called DEAP [29]. The CNN architectures are implemented in the PyTorch [30] deep learning framework. The remain of this section will describe the representation of the search space in terms of the genotype representation, the phenotype representation and the genotype-phenotype mapping.

### 3.2.1 Genotype Representation

A genotype or chromosome in SLGE is a simple program to grow a building-block, also called a cell. The program is a linear fixed-length string which consists of multiple genes of equal length, structured in head-and-tail format similar to the genotype representation in GEP (see Figure 3.3). The head of a gene composes of CE graph grammar and the tail is made up of common CNN convolution and pooling operations. Given the length of a gene head  $h$ , the tail length  $t$  is a function of  $h$  expressed as:

$$t = h + 1 \tag{3.3}$$

Thus, the length of a gene in SLGE can be expressed as:

$$2h + 1 \tag{3.4}$$

And the length of a chromosome consists of  $m$  genes is:

$$m * (2h + 1) \tag{3.5}$$

The length of a gene head  $h$  and the number of genes  $m$  in a chromosome are hyperparameters that must be set by the user.

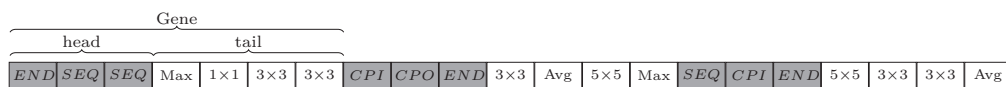


Figure 3.3: A schematic representation of SLGE genotype.

<sup>2</sup> <https://geppy.readthedocs.io/en/latest/index.html>

With the exception of recursive instruction (*REC*), the head of a gene consists of the CE graph grammar described in section 3.1.2. The (*REC*) instruction is not used because it is more appropriate for evolving recurrent neural networks (RNNs) which is out of the scope of this work. Figure 3.4 depicts the graphical illustration of (*SEQ*), (*CPI*) and (*CPO*) instructions. The standard convolution operations such as Conv $1\times 1$ , Conv $3\times 3$  and Conv $5\times 5$ , and average pooling and max pooling operations are used for a tail of a gene to achieve flexibility in the search space. This will enable SLGE to evolve modules similar to human experts designed ones like Inception-ResNet-Blocks [23]. In Figure 3.3,  $1\times 1$ ,  $3\times 3$  and  $5\times 5$  represent Conv $1\times 1$ , Conv $3\times 3$  and Conv $5\times 5$  standard convolution operations respectively, and Avg and Max represent average pooling and max pooling operations respectively. The genotype space representation and phenotype space representation in SLGE are distinctively separated. The algorithm that encodes the building-blocks for candidate architectures as chromosomes of linear fixed-length strings (see example in Figure 3.3) in the genotype space is presented in Algorithm 3. The decoding algorithm, Algorithm 4, translates the chromosomes in the genotype space into the phenotype space of cells (see example in Figure 3.5) for candidate CNN architectures. The complexity of the SLGE search space can be expressed as:  $n \times (\# \text{ of transformation functions})^h \times (\# \text{ of operations})^{h+1}$  possible chromosomes (cells) of candidate CNN architectures, where  $h$  is a gene head length and  $n$  is the number of genes in a chromosome. For example, with a set of four possible operations defined for normal cell and the four CE graph transformation functions, a chromosome with  $h = 3$  and  $n = 3$  will yield 49,152 possible chromosomes of candidate architectures. The large number of possible chromosomes may express the potential diversity of the search space.

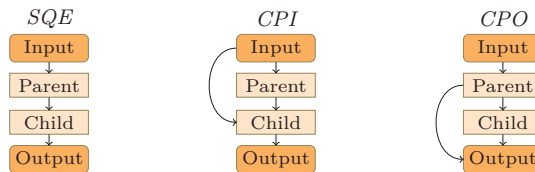


Figure 3.4: The graphical illustration of *SEQ*, *CPI* and *CPO* graph transformation functions of cellular encoding (CE).

### 3.2.2 Phenotype Representation

In SLGE, a phenotype is a building-block (cell) for a candidate architecture. The cell is a directed acyclic graph (DAG):  $G = (V, E)$ , where  $V$  is a set of nodes and  $E$  is a set of connections. The input and output nodes are input and output tensors respectively, and the other nodes represent various convolution and pooling operations. The operations without successor are depthwise concatenated to produce the output tensor, and if an operation has more than one predecessor, the feature maps of the predecessors are added together. The connections are latent information flow direction in the

**Algorithm 3** The algorithm that encodes a building-block for candidate architecture as linear fixed-length string chromosome. As inputs, it takes a CE graph transformation functions set  $fset$  and operations (convolution and pooling) set  $pset$  as basic search units in the search space, and  $m$  (the number of genes) and  $h$  (head length) as hyperparameters. It produces a SLGE chromosome as an output.

---

```

input :  $fset, pset, m, h$ 
output: SLGE chromosome
1  $ch \leftarrow string\_list()$  //Initialize chromosome as an empty string list
2 for  $i \leftarrow 1$  to  $n$  do
3    $g \leftarrow GENE(fset, pset, h)$  //Call GENE function
4    $ch.append(g)$  //Add a gene  $g$  to chromosome  $ch$ 
5 end for
6 return  $ch$  //Return a chromosome of fixed-length  $m(2h + 1)$ 
7 function  $GENE(fset, pset, h)$ 
8    $t \leftarrow h+1$  //Compute the length of a gene tail
9    $g \leftarrow string\_list()$  //Initialize a gene as an empty string list
10  for  $i \leftarrow 1$  to  $h$  do
11     $g.append(select(ce\_funcs))$  //Randomly pick a CE function
12  end for
13  for  $i \leftarrow 1$  to  $t$  do
14     $g.append(select(conv\_ops))$  //Randomly pick a convolution or pooling operation
15  end for
16  return  $g$  //Return a gene of fixed-length  $2h + 1$ 
17 end function

```

---

architecture. Figure 3.5 is the phenotype representation of the genotype (chromosome) presented in Figure 3.3. The process of translating a chromosome in a genotype space into a phenotype space of building-blocks for candidate architectures in Algorithms 4 and 5.

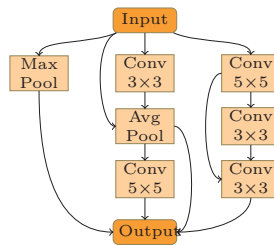


Figure 3.5: A schematic representation of SLGE phenotype. This is the phenotype (cell) representation of the chromosome in Figure 3.3.

### 3.2.3 Genotype-Phenotype Mapping

The development of a phenotype in SLGE starts from an initial cell  $G_\tau$  with the input and the output nodes. Then each gene in the chromosome of the phenotype is developed as a subgraph  $G_i$  with the first operation (convolution or pooling) in its tail as hidden node connected to input and output nodes. The subgraph  $G_i$  is developed by applying the head program of the gene to its tail part. The final cell  $G_\tau$ , that is the phenotype, is the merging of the subgraphs  $G_{i:n}$  at the input and output nodes, that is  $G_\tau = \Gamma(G_{i:n})$ , where  $\Gamma$  is the function to merge the subgraphs  $G_{i:n}$ . The evolved cells are repeatedly stacked, for a predefined number of times, to build modularized CNN

architectures as candidate solutions. The Algorithms 4 and 5 represent the genotype-phenotype mapping in SLGE.

---

**Algorithm 4** The algorithm for translating a SLGE chromosome into a cell or building-block for a candidate architecture.

---

**input :** SLGE chromosome  $\varphi \in \Phi_{genotyp-space}$   
**output:** A directed acyclic graph (*DAG*) of a cell  $G_\tau \in \Omega_{phenotype-space}$

```

1   $num \leftarrow len(\varphi)$  //Get the number of genes in the chromosome  $\varphi$ 
2   $DAG.init()$  //Initialize a cell  $G_\tau$  as DAG with the input and the output
3  for  $i \leftarrow 1$  to  $num$  do
4   $\varphi^{(i)} \leftarrow \varphi[i]$  //Get a gene  $i$  in the chromosome  $\varphi$ 
5  Create a queue  $Q$  which contains all operations in the gene  $\varphi^{(i)}$ 
6   $pnode \leftarrow Q.dequeue(0)$  //Get a parent node  $pnode$  and remove
7   $cnode \leftarrow Q.next()$  //Get a child node  $cnode$ 
8   $G_i \leftarrow subgraph(pnode)$  //Initialize a subgraph  $G_i$  for  $\varphi^{(i)}$  with a  $pnode$  with the input
//and the output
9   $pos \leftarrow 0$ 
10 while  $Q$  is not empty do
11  $gtf \leftarrow \varphi^{(i)}[pos]$  //Get a CE graph transformation function
12 if  $gtf = "END"$  then
13  $DAG.merge(G_i)$  //Merge  $G_i$  to the cell  $G_\tau$  at input and output nodes
14 return  $DAG$  //Return  $DAG$  as cell  $G_\tau$ 
15 end if
16  $TRANSFORM(G_i, gtf, pnode, cnode)$  //Call Algorithm 5
17  $pnode \leftarrow Q.dequeue(0)$ 
18  $cnode \leftarrow Q.next()$ 
19  $pos \leftarrow pos + 1$ 
20 end while
21  $DAG.merge(G_i)$  //Merge  $G_i$  to the cell  $G_\tau$  at input and output nodes
22 end for
23 return  $DAG$  //Return  $DAG$  as cell  $G_\tau$ 

```

---

### 3.3 Preliminary Experiments on CIFAR-10 and CIFAR-100

In this section, SLGE is applied to evolve CNNs in general-purpose image classification domain. The preliminary experiments aimed at verifying the effectiveness and viability of SLGE for discovering cells or building-blocks to evolve modularized CNN architectures. Because remotely-sensed imagery datasets are mostly large, for a fast preliminary experiments, two relatively small general-purpose image classification benchmarks, CIFAR-10 and CIFAR-100, were used. The experimental settings and detailed results are presented in Appendix ??, a conference paper that was presented at 2020 SSCI IEEE conference.

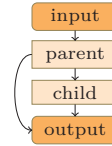
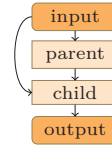
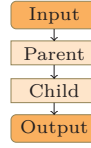
#### 3.3.1 CIFAR-10 and CIFAR-100 Benchmarks

CIFAR-10 and CIFAR-100 benchmarks [31] are widely used general-purpose image classification benchmarks in computer vision and deep learning research. Each dataset consists of 50,000 training

**Algorithm 5** A procedure calls in Algorithm 4 to transform a subgraph  $G_i$  with a parent node  $pnode$  to generate a child node  $cnode$  via the CE graph transformation function  $gtf$ .

```

1 procedure TRANSFORM( $G_i, gtf, pnode, cnode$ )
2   if  $gtf = SEQ$  then
3      $successor \leftarrow list(G_i.successors(pnode))$ 
4     for  $node$  in  $successor$  do
5        $G_i.addEdge(cnode, node)$ 
6        $G_i.removeEdge(pnode, node)$ 
7     end for
8      $G_i.addEdge(pnode, cnode)$ 
9   end if
10  if  $gtf = CPI$  then
11     $predecessor \leftarrow list(G_i.predecessors(pnode))$ 
12     $successor \leftarrow list(G_i.successors(pnode))$ 
13    for  $node$  in  $predecessor$  do
14       $G_i.addEdge(node, cnode)$ 
15    end for
16    for  $node$  in  $successor$  do
17       $G_i.addEdge(cnode, node)$ 
18       $G_i.removeEdge(pnode, node)$ 
19    end for
20     $G_i.addEdge(pnode, cnode)$ 
21  end if
22  if  $gtf = CPO$  then
23     $successor \leftarrow list(G_i.successors(pnode))$ 
24    for  $node$  in  $successor$  do
25       $G_i.addEdge(cnode, node)$ 
26    end for
27     $G_i.addEdge(pnode, cnode)$ 
28  end if
29 end procedure
    
```



and 10,000 testing RGB images. The images are of size  $32 \times 32$  RGB pixels. The images in CIFAR-10 are in 10 classes, and the ones in CIFAR-100 are in 100 classes. In the experiments, SLGE was used to evolve the architectures of CNNs to maximize their classification performance on the CIFAR-10 dataset, and the architecture of the best found network was transferred to CIFAR-100 dataset to verify its robustness. As used in evaluating manually designed CNNs, CIFAR-10 and CIFAR-100 datasets are considered as an acceptable way to evaluate the effectiveness of CNN architectures found by SLGE. Figure 3.6 shows examples of images from each class of the CIFAR-10 dataset.



Figure 3.6: Examples of CIFAR-10 dataset images from the 10 classes [32].

### 3.3.2 Formulating Architecture Search Problem

To search for architectures for CNNs in visual perception tasks, we formulated the neural architecture search (NAS) problem as follows. Given the problem space  $\psi = \{\mathcal{A}, \mathcal{S}, \mathcal{P}, t\mathcal{D}, v\mathcal{D}\}$ , where  $\mathcal{A}$  is the architecture search space,  $\mathcal{S}$  represents the search strategy,  $\mathcal{P}$  denotes the performance measure, and  $t\mathcal{D}$  and  $v\mathcal{D}$  are the training and validation datasets respectively, the objective is to find a small CNN architecture  $a^* \in \mathcal{A}$  via random search or evolutionary search strategy  $\mathcal{S}$ , which maximizes the performance measure  $\mathcal{P}$  of accuracy on the validation dataset  $v\mathcal{D}$  after training it on the training dataset  $t\mathcal{D}$ . Small architecture here means a CNN model has the number of parameters  $\theta$  less than or equal to the target network size. Mathematically, the objective function  $\mathcal{F}$  for the automatic architecture search of CNNs can be formulated as:

$$\begin{aligned} \mathcal{F}(\psi) = \max_{\theta, a} \mathcal{P}(\mathcal{L}(a(\theta), t\mathcal{D} \mid \mathcal{S}, a \in \mathcal{A}), v\mathcal{D}) \\ \text{s.t. the number of parameters } \theta \leq \mathcal{T}_{params} \end{aligned} \quad (3.6)$$

where  $\mathcal{L}$  represents the training of the model parameters  $\theta$  with the loss function and  $\mathcal{T}_{params}$  denotes the target number of parameters.

### 3.3.3 Preliminary Experimental Results

The evolved network architectures were built with  $3 \times 3$  convolution stem and initial output channel size 40, followed by three blocks denoted as  $B=[b_1, b_2, b_3]$ , then a classifier. Each block  $i$  consist of repeated cell of  $b_i$  times. A max-pooling layer inserted between the blocks to downsample. The max-pooling reduces the feature maps of each block by a half and the channels are doubled after each max-pooling operation.

First, we performed six experiments, three each on single-gene and multi-gene chromosomes, on CIFAR-10 dataset using random search strategy with early-stopping. Each multi-gene chromosome consists of two genes. In each experiment, ten chromosomes were randomly generated and trained from the scratch to analysis the effectiveness of a single-gene and a multi-gene SLGE chromosomes. The target number of parameters  $\mathcal{T}_{params}$  was set to 4M. Figure 3.7 shows that the multi-gene chromosomes achieved better classification error rate than single-gene chromosomes. The multi-gene chromosomes performed well because they evolved into a cell with multi-branch connections, which can improve training and enhance the networks performance.

We then adopted the SLGE multi-gene chromosomes and the evolutionary process in GEP to run eight more experiments, two each of four different configurations of multi-gene chromosomes, on CIFAR-10. The best discovered network was built with  $B=[3, 3, 1]$  and has 2.8M parameters. Figure 3.8 shows the graphical representation of the cell that was used to build the best discovered

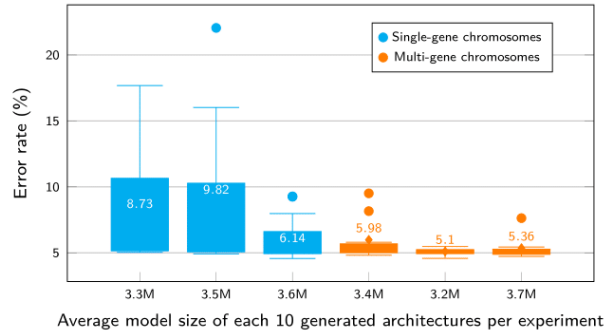


Figure 3.7: SLGE single-gene vs multi-gene chromosomes on CIFAR-10.

network. In experiments, the viability of SLGE was validated in discovering networks that improved the performance of the state-of-the-art manually designed CNNs on CIFAR-10 benchmarks and achieved a competitive results with the state-of-the-art automated NAS methods using fewer parameters and GPU resources. The best discovered network obtained 3.74% classification error rate on CIFAR-10. To evaluate the effectiveness of the best discovered network, we transferred its architecture to CIFAR-100 dataset by replacing the classifier head with 100 classes and trained it from scratch. It achieved 22.95% classification error rate using approximately 2.8M parameters. These are considerable competitive results which demonstrate the effectiveness and viability of SLGE for automated architecture search in visual perception tasks.

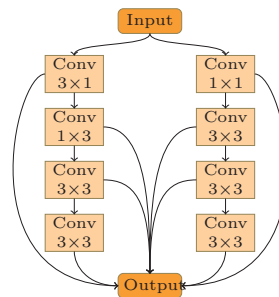


Figure 3.8: Visualization of the cell or building-block for the best discovered network by SLGE on CIFAR-10 dataset.

### 3.4 Extending SLGE to Remote Sensing Image Understanding

DNNs have recently been introduced into the remote sensing (RS) community for the interpretation and understanding of remotely-sensed imagery. Since 2014, CNNs have achieved excellent performance in the understanding and interpretation of high-resolution satellite or aerial images. Several different networks have been suggested for various RS image understanding tasks including land-use and land-cover classification, scene classification, semantic segmentation and object detection, and most of them are based on general-purpose image pre-trained CNNs models [33–35]

Unlike general-purpose images commonly used in the field computer vision (*cf.* section 3.3.1 ),

remote sensing images are usually in a top-view perspective with complex background (see Figure 3.9). Moreover, remote sensing images are considerably different from general-purpose images with respect to spectral bands, spatial resolution and radiometric resolution [36]. Therefore, the pre-trained CNNs learned on general-purpose image dataset (e.g. ImageNet [37]) may not be sufficient for RS image understanding tasks. Inspired by these observations, we are convinced that an automated method to design CNN architectures suitable for RS image understand tasks is very important.

Based on the preliminary experimental results which demonstrate that SLGE is an effective and viable approach for automated NAS in visual perception tasks, we improved the search space and extended SLGE to find well performing CNN architectures for the understanding of high-resolution satellite or aerial images. By this, we introduced automated NAS into the RS community to enable the domain research scientists to adopt DNNs systems with little or no expertise in deep learning. Using random search with early-stopping strategy, as presented in chapter 4, SLGE was used to evolve modularized CNN architectures to classify single-label and multi-label multispectral satellite image scenes and single-label RGB high-resolution aerial image scenes. Also in chapter 5, the evolutionary algorithm procedure in GEP is adopted with an improved version of SLGE search space to evolve modularized CNN architectures, dubbed SLGENet, for semantic segmentation of high-resolution aerial images.

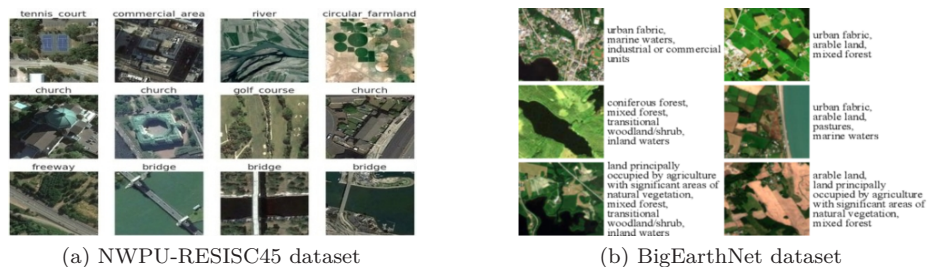


Figure 3.9: Examples of RS images: (a) Single-label RGB high-resolution aerial images from NWPU-RESISC45 dataset [38], and (b) Multi-label multispectral satellite images from BigEarthNet dataset [39].

### 3.5 Summary

This chapter presented a novel representation scheme called SLGE that is capable of evolving modularized CNN architectures for visual perception tasks. The approach combines two well established and powerful generative encoding schemes, called GEP and CE, and harnesses their strengths such as the simplification and modularity features of GEP and CE respectively for efficient automatic architecture search of CNNs via random search and evolutionary algorithm.



Preliminary experimental results are presented in CIFAR-10 and CIFAR-100 general-purpose image classification domain. The results show that networks discovered by SLGE are competitive with the state-of-the-art manually designed and automated NAS-based networks. In the subsequent chapters, the application of SLGE to remote sensing image understanding tasks such as scene classification and semantic segmentation are presented.

### 3.6 References

- [1] C. Ferreira, “Gene expression programming: a new adaptive algorithm for solving problems,” *Complex Systems*, vol. 13, no. 2, pp. 87–129, 2001.
- [2] C. Ferreira, *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*. Springer, 2006.
- [3] M. Oltean, C. Grosan, L. Diosan, and C. Mihaila, “Genetic programming with linear representation: a survey,” *International Journal on Artificial Intelligence Tools*, vol. 18, pp. 197–238, 2009.
- [4] A. Abraham, N. Nedjah, and L. d. M. Mourelle, *Evolutionary Computation: from Genetic Algorithms to Genetic Programming*, 2006, pp. 1–20.
- [5] H. S. Lopes and W. R. Weinert, “Egipsys: an enhanced gene expression programming approach for symbolic regression problems,” *International Journal of Applied Mathematics and Computer Science*, vol. 14, no. 3, pp. 375–384, 2004.
- [6] Y. Peng, C. Yuan, J.-w. Chen, X.-d. Wu, and R.-l. Wang, “Multicellular gene expression programming algorithm for function optimization,” *Control Theory Appl*, vol. 27, no. 11, pp. 1585–1589, 2010.
- [7] J. Zuo, C.-j. Tang, C. Li, C.-a. Yuan, and A.-l. Chen, “Time Series Prediction Based on Gene Expression Programming,” in *Advances in Web-Age Information Management*, 2004, pp. 55–64.
- [8] Chi Zhou, Weimin Xiao, T. M. Tirpak, and P. C. Nelson, “Evolving accurate and compact classification rules with gene expression programming,” *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 6, pp. 519–531, 2003.
- [9] J. Zhong, L. Feng, and Y.-S. Ong, “Gene expression programming: A survey,” *IEEE Computational Intelligence Magazine*, vol. 12, no. 3, pp. 54–72, 2017.
- [10] F. Gruau, “Neural network synthesis using cellular encoding and the genetic algorithm.” Doctoral Dissertation, L’universite Claude Bernard-lyon I, 1994.
- [11] F. Gruau, “Genetic synthesis of boolean neural networks with a cell rewriting developmental process,” in *[Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*, 1992, pp. 55–74.
- [12] F. Gruau, “Genetic synthesis of modular neural networks,” in *Proceedings of the 5th International Conference on Genetic Algorithms*, 1993, p. 318–325.
- [13] F. Gruau and K. Quatramaran, “Cellular encoding for interactive evolutionary robotics,” CWI, Amsterdam, Amsterdam, The Netherlands, Tech. Rep., 1996.
- [14] W. Banzhaf, F. D. Francone, R. E. Keller, and P. Nordin, *Genetic Programming: An Introduction*. San Francisco: Morgan Kaufmann, 1998.
- [15] C. M. Friedrich and C. Moraga, “An evolutionary method to find good building-blocks for architectures of artificial neural networks,” in *Sixth International Conference on Information*

- Processing and Management of Uncertainty in Knowledge Based Systems (IPMU'96)*, vol. 2, 1996, pp. 951–956.
- [16] C. M. Friedrich and C. Moraga, “Using genetic engineering to find modular structures and activation functions for architectures of artificial neural networks,” in *In Proceedings of the Fifth Fuzzy Days*, 1997, pp. 150–161.
- [17] F. Gruau, “Artificial cellular development in optimization and compilation,” in *Towards Evolvable Hardware*. Berlin, Heidelberg: Springer, 1996, vol. 1062, pp. 48–75.
- [18] D. Whitley, F. Gruau, and L. Pyeatt, “Cellular encoding applied to neurocontrol,” in *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, 1995, pp. 460–467.
- [19] C. Ferreira, “Designing neural networks using gene expression programming,” in *Applied Soft Computing Technologies: The Challenge of Complexity*, 2006, pp. 517–535.
- [20] W. Wang, Q. Li, and X. Qi, “Gene Expression Programming Neural Network for Regression and Classification,” in *Proceedings of the 3rd International Symposium on Advances in Computation and Intelligence*, 2008, pp. 212–219.
- [21] X. Li, C. Zhou, W. Xiao, and P. C. Nelson, “Prefix gene expression programming,” in *Proceedings of the GECCO*, 2005, p. 25–31.
- [22] J. Drchal and M. Šnorek, “Tree-Based Indirect Encodings for Evolutionary Development of Neural Networks,” in *Artificial Neural Networks - ICANN 2008*, 2008, pp. 839–848.
- [23] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, p. 4278–4284.
- [24] T. Liu, M. Chen, M. Zhou, S. S. Du, E. Zhou, and T. Zhao, “Towards understanding the importance of shortcut connections in residual networks,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [25] F. Gruau, D. Whitley, and L. Pyeatt, “A comparison between cellular encoding and direct encoding for genetic neural networks,” in *Proceedings of the 1st annual conference on genetic programming*. MIT Press, 1996, pp. 81–89.
- [26] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [27] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.
- [28] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, “Practical block-wise neural network architecture generation,” in *IEEE Conference on CVPR*, 2018, pp. 2423–2432.
- [29] F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné, “Deap: Evolutionary algorithms made easy,” *Journal of Machine Learning Research*, vol. 13, no. 1, p. 2171–2175, 2012.
- [30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.
- [31] A. Krizhevsky, V. Nair, and G. Hinton, “CIFAR-10 and CIFAR-100 datasets,” 2009, Date Accessed: 2021-09-19. [Online]. Available: <https://www.cs.toronto.edu/kriz/cifar.html>

- 
- [32] K. Li, G. Wan, G. Cheng, L. Meng, and J. Han, "Object detection in optical remote sensing images: A survey and a new benchmark," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 159, pp. 296–307, 2020.
- [33] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, "Deep learning in remote sensing applications: A meta-analysis and review," *ISPRS Journal of Photo. and RS*, vol. 152, pp. 166–177, 2019.
- [34] J. Song, S. Gao, Y. Zhu, and C. Ma, "A survey of remote sensing image classification based on CNNs," *Big Earth Data*, vol. 3, no. 3, pp. 232–254, 2019.
- [35] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 22–40, 2016.
- [36] J. A. Richards and X. Jia, "Sources and characteristics of remote sensing image data," in *Remote Sensing Digital Image Analysis: An Introduction*, 1999, pp. 1–38.
- [37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [38] G. Cheng, J. Han, and X. Lu, "Remote sensing image scene classification: Benchmark and state of the art," *Proceedings of the IEEE*, vol. 105, no. 10, pp. 1865–1883, 2017.
- [39] G. Sumbul, A. de Wall, T. Kreuziger, F. Marcelino, H. Costa, P. Benevides, M. Caetano, B. Demir, and V. Markl, "Bigearthnet-mm: A large-scale, multimodal, multilabel benchmark archive for remote sensing image classification and retrieval [software and data sets]," *IEEE Geoscience and Remote Sensing Magazine*, vol. 9, no. 3, pp. 174–180, 2021.

## Chapter 4

# Extending SLGE to Remote Sensing Image Scene Classification

*Random search can be more efficient than nonrandom search—something that Good and Turing had discovered at Bletchley Park. A random network, whether of neurons, computers, words, or ideas, contains solutions, waiting to be discovered, to problems that need not be explicitly defined.*

— **George B. Dyson**  
(Turing’s Cathedral)

So far, based on the preliminary experimental results presented in chapter 3, SLGE has demonstrated to be viable for evolving and optimizing the architectures of CNNs for visual perception tasks. In this chapter, SLGE-based architecture representation is explored by the means of random search with early-stopping strategy to find well performing modularized CNN architectures for the classification of single-label and multi-label multispectral satellite image scenes and single-label RGB high-resolution aerial image scenes. Four RS image scene classification benchmarks, two RGB high-resolution aerial image datasets and two multispectral satellite image datasets, are employed for the experiments. Although random search is a simple search strategy, it is a competitive strategy if the architecture search space is not intractable and it converges to a solution within a distance of the global optimum with probability one. By using random search, we also validate the tractability and expressiveness of SLGE representation space.

---

Parts of this chapter is published in Broni-Bediako *et al.*, 2021 (see Appendix C). Reprinted with permission from the publisher.

## 4.1 Introduction

RS image scene classification has many important applications including urban planning, land resource management, geospatial object detection and environmental monitoring [1]. Vast volumes of very high resolution RS images have become easily available due to progress in satellite technology [2]. For over 60 years, researchers have been applying learning algorithms in resolving different kinds of RS image classification [3]. Interpretation of RS imagery is a complex and tedious task. Until relatively recent years, even a simple task like identifying building footprints requires laborious sub-specialities. Deep learning [4] has opened up an entirely novel frontier of learning algorithms including convolutional neural networks (CNNs) techniques that have been adopted in the RS research community. Since 2014, there has been a significant progress in RS scene classification using CNN models. Most CNN-based RS scene classification studies employed pre-trained networks as feature extractors for scene classification. And other works focus on fine-tuning the pre-trained networks on RS image scene classification datasets [5–7]. Several studies depart from pre-trained networks and manually designed task-specific architectures for RS scene classification [8–10]. And for the first time, Chen *et al.* [11] introduced automated NAS into RS scene classification. Refer to section 2.4.1 of chapter 2 for review on RS scene classification with CNNs.

### 4.1.1 Automated NAS with Random Search

Various search strategies have been adopted in automated NAS to learn CNN architectures for computer vision and natural language processing (NLP) problems. This include reinforcement learning [12, 13], evolutionary algorithms [14, 15], gradient-based methods [16, 17], Bayesian optimization [18] and random search [19, 20]. Random search strategy is a simple optimization method that is able to find models that are good or better within a small fraction of the computation time compare with grid search. It can find better models by effectively searching a larger, less promising search space [21]. Many open-source hyper-parameter optimization tools implement random search, including Vizier [22] and AHSA [23] used in Chen *et al.* [19] and Li & Talwalkar *et al.* [24] respectively, to find CNN architecture via random search. As highlighted in Yu *et al.* [25], random search is competitive with the state-of-the-art reinforcement learning [26, 27] and other NAS methods [14, 16, 28]. And it converges to a solution within a distance of the global optimum with probability one [29, 30]. The efficiency of random search has further been explored using classical random graph models to generate CNNs architectures [20].

## 4.2 Methodology

Based on the formulated architecture search problem in section 3.3.2 of chapter 3, we search for CNN architectures for RS image scene classification using random search with early-stopping strategy. Following previous studies in automated NAS [13, 14], the SLGE-based random search searches for the topological structure of cells which are repeatedly stacked to construct an entire CNN architecture (see Fig. 4.1). The cell is a directed acyclic graph (DAG) generated via the search strategy  $\mathcal{S}$  from the architecture search space  $\mathcal{A}$ . Each node of the cell represents a convolutional layer whereas edges are representation of latent information flow direction in the CNN architecture. To reduce computational cost, proxy task is normally employ on the search space to search for good architectures [26, 31]. And to make the architecture search result robust, we repeat the search several times with different random seeds. The search space and proxy task adopted in the experiments are described as follows.

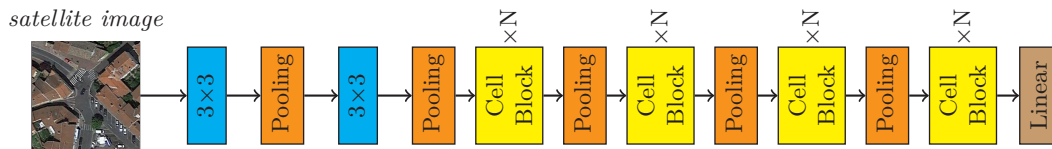


Figure 4.1: The entire CNN architecture is constructed by stacking an auto-generated cell (see Fig. 4.3) as *cell blocks* marked in yellow. The architecture begins with two  $3 \times 3$  convolution layers followed by several auto-generated cell blocks repeated  $N$  times with average pooling layers inserted in between them to downsample the spatial resolutions.

### 4.2.1 Search Space

The SLGE search space represents individual architectures as linear string structures (see Fig. 4.2) as described in chapter 3. And the mapping function presented in chapter 3 is used to translate the linear structure into a cell (see Fig. 4.3), which is typically stacked repeatedly to build a final CNN architecture in Fig. 4.1. Three types of convolution operations  $\text{conv}3 \times 3$ ,  $\text{conv}1 \times 1$  and  $\text{conv}3 \times 3$  depthwise are used as cell nodes. The identity operation which is employed in other studies [14, 16] to enable shortcut connections in a network is not explicitly included in the set of possible convolutional operations because SLGE implicitly encodes the shortcut (or skip) connections (see section 3.2.1 of chapter 3). We adopt four different search spaces of SLGE linear string structures for the experiments (see Table 4.1). This defines the overall search space with nodes between 3 and 16 in a cell, and contains approximately  $2.4 \times 10^6$  possible cells in 16-node case. Refer to chapter 3 for further details of SLGE representation space.

Table 4.1: The four different SLGE search spaces use in the experiments.

Search Space	SLGE Linear Structure Settings		
	Element	Head	Tail
A	3	2	3
B	3	3	4
C	4	2	3
D	4	3	4

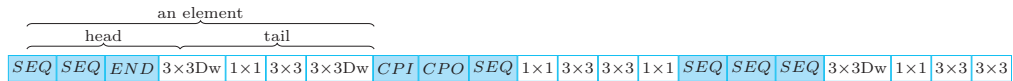


Figure 4.2: SLGE linear structure of the best discovered network. *SEQ*, *CPO*, *CPI* and *END* are transformation functions (refer to section 3.2 of chapter 3 for details), and  $1\times 1$ ,  $3\times 3$  and  $3\times 3Dw$  (depthwise) are convolution operations.

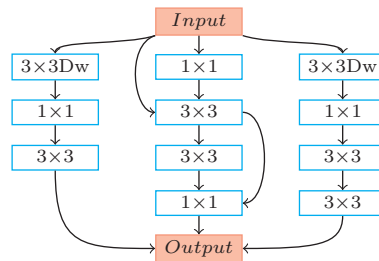


Figure 4.3: SLGE module of the best discovered network. A cell representation of the linear string shown in Fig. 4.2. The convolution operations without successor are depthwise concatenated to produce an output, and if an operation has more than one predecessor, the feature maps of the predecessors are added together. This is a common approach in cell-based search [13].

## 4.2.2 Proxy Task

Searching for large-scale CNN architectures directly on a large datasets such as NWPU-RESISC45 and BigEarthNet is computationally expensive. It takes several days for each architecture to converge. To reduce the amount of computation resources required during the architecture search, we employed a small-scale task as a proxy task for quick-search-evaluation. This may provide a predictive signal about training the large-scale architecture settings. Following previous studies in computer vision [13, 14, 26], we designed a proxy task by employing a small-scale network which consist of few repeated cell blocks in Fig. 4.1 and resize the RS images into  $32\times 32$  low resolution to perform the architecture search over small networks. The proxy task network begins with only one  $3\times 3$  convolution stem with output channel size  $C$ , followed by three cell blocks denoted as  $B = [b_1, b_2, b_3]$ , then a classifier. Each block  $i$  in  $B$  consists of repeated cell of  $b_i$  times with average pooling layer inserted between them to downsample the feature dimension. Average pooling is used to reduce the feature maps resolutions of each cell block by half, then the channels  $C$  is doubled.

We implement the proxy task with 25,200 sample set of NWPU-RESISC45 dataset. The sample

set is split into 80% training and 20% validation subsets with normalization. To speed up the search process, proxy task networks with  $C = 32$  and  $B = [2, 2, 2]$ , which have relatively small model memory-footprint (less than 2M parameters), are used. Each generated proxy network is trained on the training subset without any data augmentation for 100 epochs with batch size of 256 and evaluated on the validation subset to determine its classification accuracy. Then, the top proxy networks are re-trained from scratch to full convergence in the large-scale architecture settings shown in Fig. 4.1 and present as the best CNNs architectures.

## 4.3 Experiments

All the architecture searches are performed on the NWPU-RESISC45 single-label classification task. A total of 400 CNN architectures are searched on a 11GB GPU GeForce GTX1080 Ti machine using five different random seeds for 9.6 GPU days. The best architectures discovered on NWPU-RESISC45 are evaluated on AID single-label, EuroSAT single-label and BigEarthNet multi-label scene classification tasks to verify their effectiveness. We compared the results on NWPU-RESISC45, AID and EuroSAT with the state-of-the-art transfer learning models and a NAS-based baseline model, whereas the results on BigEarthNet are compared with the state-of-the-art CNN models trained from scratch and the baseline. We implement the architectures in the PyTorch [32] deep learning framework.

### 4.3.1 Datasets

The NWPU-RESISC45<sup>1</sup> is a single-label dataset created by the Northwestern Polytechnic University [1]. The dataset consists of 31,500 satellite images categorized into 45 classes. Each class includes 700 images of size  $256 \times 256$  in RGB space with spatial resolution varying from 30m to 0.2m. The images were acquired from Google Earth imagery of more than 100 countries and regions. AID<sup>2</sup> is made up of satellite images acquired from Google Earth imagery of different countries and regions [33]. The dataset consists of 10,000 single-label images categorized into 30 classes. The number of samples per class varies from 220 up to 420. Each image is  $600 \times 600$  pixels in RGB space with spatial resolution varying from about 8m to about 0.5m.

EuroSAT<sup>3</sup> is a single-label multispectral image dataset acquired from Sentinel-2 satellite images of cities in the 34 European countries [34]. The dataset consists of 10 different classes with 2,000 to 3,000 images per class. In total, the dataset has 27,000 images with 13 different spectral bands of size  $64 \times 64$  pixels. We use the images of the RGB and RGB-NIR (RGB and Near-infrared) 10m

<sup>1</sup>[https://1drv.ms/u/s!AmgKYzARBl5ca3HNaHllzp\\_IXjs](https://1drv.ms/u/s!AmgKYzARBl5ca3HNaHllzp_IXjs)

<sup>2</sup><https://captain-whu.github.io/AID/>

<sup>3</sup><https://github.com/phelber/EuroSAT>



bands. BigEarthNet<sup>4</sup> is a multi-label dataset created from Sentinel-2 multispectral satellite images scattered over different European countries [35]. The dataset consists of 590,326 images with 12 different spectral bands. Each image is associated with one or more class labels extracted from CORINE land cover (CLC) map of 2018. Originally, BigEarthNet has 43 classes, then the authors realigned it in a new nomenclature of 19 classes based on CLC Level-3 nomenclature. We use the new nomenclature classes, which is called BigEarthNet19 [36], and the RGB and RGB-NIR 10m bands images of 120×120 pixels.

NWPU-RESISC45, AID and EuroSAT are challenging RS scene classification benchmarks, however, BigEarthNet is the more complex. We randomly split NWPU-RESISC45, AID and EuroSAT into 80/20 ratio for training and testing respectively, and 20% of the training set is used as validation set. The split is applied class-wise. BigEarthNet is split as 269,695 for training, 123,723 for validation and 125,866 for testing as provided by the authors. The testing sets are only used for the final performance evaluation of a model.

### 4.3.2 Training Details

During the architecture search, the candidate architectures are trained and evaluated on NWPU-RESISC45 based on the proxy task described in section 4.2.2. After the search, the cell of the best found architecture is repeatedly stacked as 4 cell-block  $B = [b_1, b_2, b_3, b_4]$  with initial channels  $C = 16$ , and average pooling layer is inserted between them to build the large-scale CNN architecture in Fig. 4.1, subject to model size constraint  $\mathcal{T}_{params} \leq 5.5\text{M}$ . We resized the images in NWPU-RESISC45 and AID to 224×224, but maintained the 64×64 and 120×120 size of EuroSAT and BigEarthNet respectively, and augmented the training subset as in He *et al.* [37]. The large-scale architecture is trained from scratch on NWPU-RESISC45, AID and EuroSAT for 600 epochs, and on BigEarthNet for 200 epochs. We use a batch size of 128 for all the datasets. We use fastai<sup>5</sup> [38], a PyTorch-based library, for the training of the architectures. All the architectures, during search and after search, are trained using 1-cycle policy in Smith [39] with Adam optimizer [40]. The learning rate is set to go from 0.0004 to 0.01 linearly while the momentum goes from 0.95 to 0.85 linearly in phase one of 1-cycle policy. Then in phase two, the learning rates follows cosine annealing from 0.01 to 0, as the momentum goes from 0.85 to 0.95 with the same annealing, and the weight decay is set to 0.0001. These are default values in 1-cycle policy [39].

<sup>4</sup><http://bigearth.net/>

<sup>5</sup><https://fastai1.fast.ai/>

### 4.3.3 Baseline Network

We select EfficientNets [41], a gradient-based NAS model pre-trained on ImageNet, as a baseline for the study. For a fair comparison, we adopt EfficientNet-B0<sup>6</sup> model which has 5.3M parameters [41] comparable to those of our best networks. The experimental results on the four datasets (NWPU-RESISC45, AID, EuroSAT and BigEarthNet) are compared with EfficientNet-B0 trained from scratch and with fine-tuning from the parameters learned on ImageNet. The linear classifier of EfficientNet-B0 was simply replaced with a linear classifier which corresponds to the number of classes in the considered datasets. EfficientNet-B0 was fully trained from scratch with 80/20 training-testing ratio using the hyper-parameters described in section 4.3.2. On fine-tuning, the replaced linear classifier was trained for 100 epochs with the same training settings in section 4.3.2. Then, we halved the learning rate and trained the whole model for 150 epochs. The baseline was fine-tuned with both 20/80 and 80/20 training-testing ratios on NWPU-RESISC45, and 50/50 and 80/20 training-testing ratios on AID. On EuroSAT and BigEarthNet, it was fine-tuned it with only 80/20 training-testing ratios.

### 4.3.4 Searching on NWPU-RESISC45 Dataset

To investigate the effectiveness of proposed method, we conducted several experiments on 4 different search spaces we adopted from SLGE (see Table 4.1). For each search space, the experiment was performed 5 times with different random seeds. We randomly sampled 20 individual cells per experiment to build 20 different proxy architectures, then trained and evaluated based on the proxy task described in section 4.2.2. A total of 400 individual cells were sampled to build 400 different architectures during the architecture search process. Fig. 4.4 shows the search results of the proxy task for each search space. The results show an average accuracy of 82.23% with 1.72% standard deviation over the 400 proxy networks. Although the networks in the proxy task were not trained to converge, the results surpassed the feature extraction method in Cheng *et al.* [1] and compete with the fine-tuning approach in Jiang *et al.* [42].

The cells of top 20 networks (one from each of the five experiments per a search space) found during the architecture search are used to build 20 large-scale architectures (see Fig. 4.1), and trained from scratch using the training settings in section 4.3.2. The architectures were trained and evaluated ten times to reduce the bias effect, and the results were averaged to determine the best network for RS scene classification. The best network achieved  $96.56 \pm 0.13\%$  accuracy as shown in Table 4.2. The results show an average accuracy of 95.93% with 0.21% standard deviation over the top 20 networks.

<sup>6</sup><https://github.com/lukemelas/EfficientNet-PyTorch>

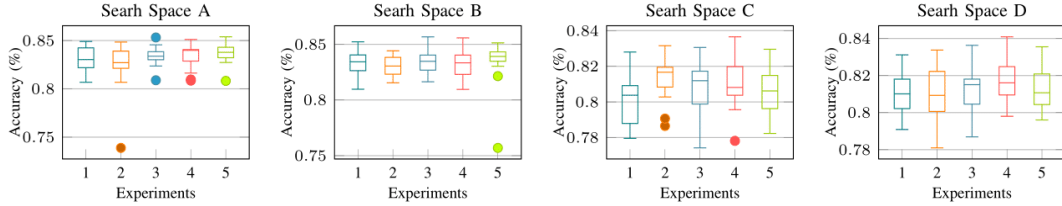


Figure 4.4: Search results of the 400 proxy networks trained according to the proxy task described in section 4.2.2. In each search space, we performed 5 experiments by sampling 20 individual cells per experiment to build 20 different proxy architectures and trained each for 100 epochs. The results show accuracy distribution of 20 networks per experiment in each search space. It took 9.6 GPU days to train and evaluate all the 400 proxy networks.

Table 4.2: The results of top 20 networks found via proxy task and fully trained on NWPU-RESISC45 in the large-scale architecture settings (see Fig. 1). Boldface is the best result.

Search Space	Seed	Cell Block	Accuracy (%)	Size
A	320	[4, 5, 5, 3]	95.37±0.09	5.3M
	321	[4, 4, 4, 3]	95.12±0.19	5.5M
	322	[3, 2, 1, 2]	<b>96.12±0.18</b>	5.5M
	323	[3, 2, 1, 2]	95.45±0.31	5.4M
	324	[3, 3, 1, 2]	95.74±0.23	5.5M
B	330	[2, 2, 1, 1]	96.02±0.14	5.0M
	331	[2, 3, 3, 1]	<b>96.36±0.11</b>	5.4M
	332	[2, 1, 2, 1]	96.13±0.14	5.4M
	333	[3, 4, 4, 1]	95.91±0.21	5.3M
	334	[4, 4, 4, 3]	95.72±0.20	5.0M
C	420	[4, 3, 4, 3]	96.14±0.15	5.5M
	421	[2, 2, 1, 2]	95.73±0.18	5.5M
	422	[2, 3, 2, 1]	<b>96.56±0.13</b>	5.1M
	423	[2, 3, 3, 2]	95.37±0.86	5.2M
	424	[3, 4, 4, 3]	95.81±0.10	5.5M
D	430	[2, 2, 1, 1]	96.08±0.15	5.1M
	431	[3, 4, 4, 1]	<b>96.44±0.21</b>	5.5M
	432	[2, 2, 1, 1]	96.02±0.385	5.0M
	433	[2, 2, 1, 1]	96.28±0.11	5.3M
	434	[2, 2, 2, 1]	96.21±0.13	5.0M

In Table 4.3, we compared the results with the state-of-the-art transfer learning methods and the baseline in terms of classification accuracy, model size and FLOPS. The state-of-the-art transfer learning methods on NWPU-RESISC45 in the literature are trained with 20/80 training-testing ratio. For competitive comparison, we compared our results to these methods (see Table 4.3). However, for a fair comparison, we also compared the results with the state-of-the-art handcrafted CNNs networks and gradient-based NAS networks which are trained from scratch with 80/20 training-testing ratio in [43] (see Table 4.5).

With few parameters, our best network improved the performance of most of the popular transfer learning methods on NWPU-RESISC4. Table 4.3 shows an approximately 0.2% improvement over the state-of-the-art results reported in DLGFF [44]. By analysing Table 4.3, one can observe that

fine-tuning the baseline with 80/20 training-testing ratio achieved a competitive result. However, training the baseline from scratch, our best network achieved an accuracy of approximately 1% higher than the baseline. Table 4.5 shows our best discovered network achieved an approximately 3% higher accuracy than gradient-based NAS model [43]. In addition, the best network improved the performance of the state-of-the-art handcrafted CNN models reported in [43] by 4% (see Table 4.5).

Table 4.3: Comparisons between our method and the baseline network and the state-of-the-art transfer learning (feature extraction and fine-tuning) methods with 20/80 training-testing ratio on NWPU-RESISC45 dataset. Here our method is auto-generated architectures discovered on NWPU-RESISC45. The baseline results are based on our training. Boldface is the best result.

Model	Accuracy (%)	Size	FLOPS	Method	
BoCF with VGGNet-16 [45]	84.32±0.17	138M <sup>†</sup>	15.5B <sup>†</sup>		
SAL-TS-Net fusion with GoogLeNet [46]	87.01±0.19	13.6M <sup>†</sup>	2.80B <sup>†</sup>		
D-CNN with VGGNet-16 [47]	91.89±0.22	138M <sup>†</sup>	15.5B <sup>†</sup>	Feature	
DLGFF with CaffeNet [44]	96.37±0.05	60M <sup>†</sup>	0.72B <sup>†</sup>	Extraction	
DDRL-AM with ResNet-18 [48]	92.46±0.09	11.5M <sup>†</sup>	1.82B <sup>†</sup>	(Handcrafted)	
GLANet with VGGNet-16 [49]	93.45±0.17	138M <sup>†</sup>	15.5B <sup>†</sup>		
FDPResNet with ResNet-50 [50]	95.40±0.11	25.6M <sup>†</sup>	4.12B <sup>†</sup>		
VGGNet-16 with SVM [1]	90.36±0.18	138M <sup>†</sup>	15.5B <sup>†</sup>		
VGGNet-16 with XGBoost [42]	83.35	138M <sup>†</sup>	15.5B <sup>†</sup>	Fine-Tuning	
Fusion of VGGNet-16 & CaffeNet [51]	95.36±0.22	198M <sup>†</sup>	16.2B <sup>†</sup>	(Handcrafted)	
Fusion of ResNet-50 & VGGNet-16 [52]	94.03	163.6M <sup>†</sup>	19.6B <sup>†</sup>		
	91.98*	5.3M	0.39B	Fine-Tuning	
EfficientNet-B0 [41] (baseline)	96.43 <sup>‡</sup>	5.3M	0.39B	(Auto-Generated)	
	95.90 <sup>‡</sup>	5.3M	0.39B	Fully-Trained	
				(Auto-Generated)	
	$B=[3, 2, 1, 2]$	96.12±0.18 <sup>‡</sup>	5.5M	0.58B	
	$B=[2, 3, 3, 1]$	96.36±0.11 <sup>‡</sup>	5.4M	0.91B	Fully-Trained
Our method ( $C=16$ )	$B=[2, 3, 2, 1]$	<b>96.56±0.13<sup>‡</sup></b>	5.1M	0.56B	(Auto-Generated)
	$B=[3, 4, 4, 1]$	96.44±0.21 <sup>‡</sup>	5.5M	0.73B	

<sup>†</sup> Model size and FLOPS are based on the open source backbone pre-trained networks.

\* This result is based on 20/80 training-testing ratio.

<sup>‡</sup> This results is based on 80/20 training-testing ratio.

To provide further understanding of the performance of our proposed approach, we present a confusion matrix (see Fig. 4.5) to illustrate the correct and incorrect classification results of our best network on NWPU-RESISC4. Fig. 4.5 shows that our best network achieved classification accuracy rate greater than 96% on most of the scene classes. Our best network achieved high accuracy rate on image scenes such as *chaparral*, *forest*, *sea ice* and *snowberg* which have single texture and exhibit low inter-class similarity. On the other hand, scenes such as *commercial area*, *industrial area* and *medium residential* which have complex textural composition with high inter-class similarity and intra-class diversity experienced low accuracy rate. The highest misclassification results occurs between *palace* and *church* classes with 89% and 76% accuracy rate respectively. Images in both classes are very similar in structural and textural characteristics (see Fig. 4.6).



best 4 architectures discovered on NWPU-RESISC45 (see Table 4.2) were evaluated on AID. We trained them from scratch with a linear classifier head of 30 classes. The training details remain the same as applied on NWPU-RESISC45.

The results are compared with the state-of-the-art transfer learning methods and the baseline in terms of classification accuracy, model size and FLOPS as shown in Table 4.4. In the literature, the state-of-the-art transfer learning methods on AID are trained with 50/50 training-testing ratio. We compared the results with these methods for competitive comparison (see Table 4.4). Also, for a fair comparison, we compared the results with the state-of-the-art handcrafted CNNs networks and gradient-based NAS networks which are trained from scratch with 80/20 training-testing ratio in [43] (see Table 4.5).

Table 4.4 shows our best network with only 5.1M parameters competed with the state-of-the-art feature extraction methods, D-CNN [47] and GLANet [49], which used VGGNet-16 as backbone pre-trained network. It also improved the performance of the state-of-the-art fine-tuning methods by approximately 1% accuracy rate. By examining Table 4.4, one can observe that the baseline with 80/20 training-testing ratio improved the performance of the best discovered network by approximately 0.5% when fine-tuned. However, training the baseline from scratch, the best discovered network achieved approximately 2% higher accuracy than the baseline. Table 4.5 shows the best discovered network improved the performance of the gradient-based NAS model [43] by 5% accuracy rate. It also improved the performance of the state-of-the-art handcrafted CNN models reported in [43] by 6% (see Table 4.5).

Table 4.4: Comparisons between our method and the baseline network and the state-of-the-art transfer learning (feature extraction and fine-tuning) methods with 50/50 training-testing ratio on AID dataset. Here our method is auto-generated architectures discovered on NWPU-RESISC45 which are evaluated on AID. The baseline results are based on our training. Boldface is the best result.

Model	Accuracy (%)	Size	FLOPS	Method	
AID Benchmark (VGGNet-16) [33]	89.64±0.36	138M <sup>†</sup>	15.5B <sup>†</sup>		
Two-Stream Fusion (VGGNet-16) [53]	94.58±0.25	276M <sup>†</sup>	31.0B <sup>†</sup>	Feature	
SAL-TS-Net fusion with GoogLeNet [46]	95.99±0.35	13.6M <sup>†</sup>	2.80B <sup>†</sup>	Extraction	
D-CNN with VGGNet-16 [47]	<b>96.89±0.10</b>	138M <sup>†</sup>	15.5B <sup>†</sup>	(Handcrafted)	
GLANet with VGGNet-16 [49]	96.66±0.19	138M <sup>†</sup>	15.5B <sup>†</sup>		
Fusion of VGGNet-16 & CaffeNet (LR) [51]	94.93±0.28	198M <sup>†</sup>	16.2B <sup>†</sup>	Fine-Tuning	
Fusion of VGGNet-16 & CaffeNet (SVM) [51]	95.36±0.22	198M <sup>†</sup>	16.2B <sup>†</sup>	(Handcrafted)	
EfficientNet-B0 [41] (baseline)	95.54*	5.3M	0.39B	Fine-Tuning	
	96.65 <sup>‡</sup>	5.3M	0.39B	(Auto-Generated)	
	94.35 <sup>‡</sup>	5.3M	0.39B	Fully-Trained (Auto-Generated)	
Our method ( $C=16$ )	$B=[3, 2, 1, 2]$	95.08±0.18 <sup>‡</sup>	5.5M	0.58B	
	$B=[2, 3, 3, 1]$	95.96±0.20 <sup>‡</sup>	5.4M	0.91B	Fully-Trained
	$B=[2, 3, 2, 1]$	96.10±0.18 <sup>‡</sup>	5.1M	0.56B	(Auto-Generated)
	$B=[3, 4, 4, 1]$	95.49±0.16 <sup>‡</sup>	5.5M	0.73B	

<sup>†</sup> Model size and FLOPS are based on the open source backbone pre-trained networks.

\* This result is based on 50/50 training-testing ratio.

<sup>‡</sup> This result is based on 80/20 training-testing ratio.

Table 4.5: Comparisons between our method and the state-of-the-art handcrafted CNN models and gradient-based NAS method fully trained on NWPU-RESISC45 (NWPU) and AID with 80/20 training-testing ratio. The results of the state-of-the-art handcrafted models are reported in [43]. Boldface is the best result.

Model	NWPU		AID		Method	
	Accuracy (%)	Accuracy(%)	Size	FLOPS		
GoogLeNet [54]	91.61±0.65	88.15±0.77	6.8M	1.40B	Handcrafted	
Inception-V3 [55]	92.58±0.26	89.03±1.27	23.2M	2.85B		
ResNet-50 [37]	92.01±0.29	88.10±0.65	25.6M	4.12B		
DenseNet [56]	92.71±0.17	87.50±0.24	13.7M	3.42B		
MobileNet-V2 [57]	92.29±0.19	89.33±0.37	6.9M	0.58B		
CNN Arch. Learning [43]	–	91.12±0.49	3.8M	–	Auto-Generated (Gradient-Based)	
	93.67±0.24	–	5.2M	–		
Our method ( $C=16$ )	$B=[3, 2, 1, 2]$	96.12±0.18	95.08±0.18	5.5M	0.58B	Auto-Generated (Random Search)
	$B=[2, 3, 3, 1]$	95.96±0.20	96.36±0.11	5.4M	0.91B	
	$B=[2, 3, 2, 1]$	<b>96.56±0.13</b>	<b>96.10±0.18</b>	5.1M	0.56B	
	$B=[3, 4, 4, 1]$	96.44±0.21	95.49±0.16	5.5M	0.73B	

We also present a confusion matrix in Fig. 4.7 to illustrate the classification accuracy. Out of the 30 scene classes, 21 classes were correctly classified over 96% accuracy rate, and only 3 classes (*resort*, *school* and *square*) were classified with accuracy rate lower than 90%. The class with the lowest accuracy of 83% was the *square*, which is mostly confused with the *centre* class. Other confused scenes include *bare-land* versus *desert* and *resort* versus *park* shown in Fig. 4.8. These pairs of scenes are very similar in textural characteristics.

### 4.3.6 Evaluating on EuroSAT Dataset

We evaluated the architecture transferability of networks discovered on a single-label RGB aerial image dataset to a single-label multispectral satellite image dataset. The best 4 architectures discovered on NWPU-RESISC45 (see Table 4.2) single-label RGB image dataset were evaluated on EuroSAT multispectral image dataset. EuroSAT is a single-label classification benchmark with 10 classes. Each architecture was trained from the scratch with a linear classifier head of 10 classes using the training details as applied on the NWPU-RESISC45. Because the images in EuroSAT are of size  $64 \times 64$  pixels, we did not apply the second  $3 \times 3$  convolution and pooling layers in the large-scale architecture settings (see Fig. 4.1).

For comparative evaluation in Table 4.6, we used images in the RGB space and RGB-NIR space of the EuroSAT dataset to train our architectures discovered on NWPU-RESISC45 and also to fine-tune the baseline network. We reported in Table 4.6 the classification accuracy to evaluate the performance of our best discovered networks and the baseline. In the RGB space, as shown in Table 4.6, our best discovered network achieved a classification accuracy of 98.67% which competes with the baseline’s 98.85% accuracy rate, and the 98.57% of the state-of-the-art fine-tuned pre-trained benchmark models reported in Helber *et al.* [34]. However, in the RGB-NIR space, our best discovered network achieved 99.76% accuracy rate, which is approximately 1%

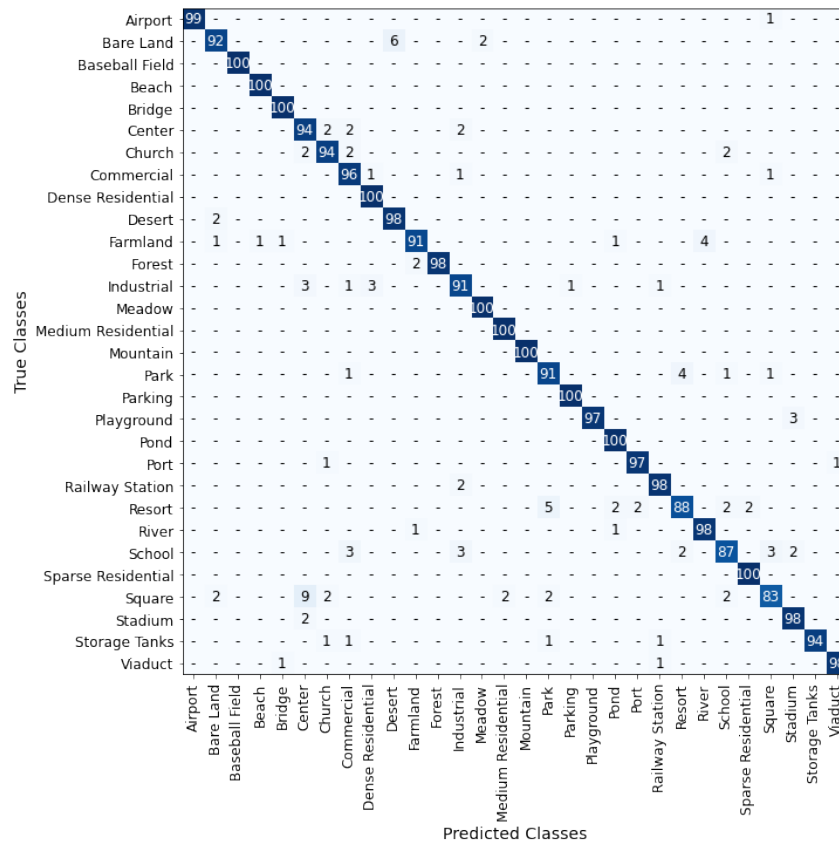


Figure 4.7: Confusion matrix of our best network on AID benchmark (Table 4.4). Diagonal values are correct predictions (%), non-diagonal values are incorrect predictions (%) and the dash (-) means no incorrect prediction.

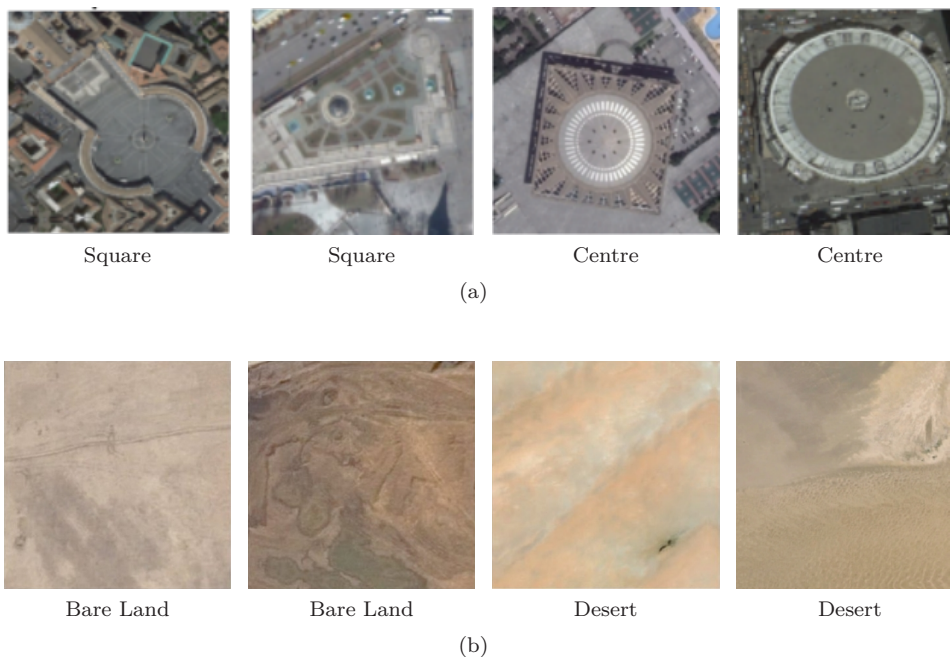


Figure 4.8: Examples of confused classes in AID (see Fig. 4.7). This shows the textural similarity among the images: (a) *square* and *centre* classes (b) *bare-land* and *desert* classes.



higher than the 98.78% of the baseline.

Table 4.6: Comparisons between our method and the baseline network and the state-of-the-art fine-tuned pre-trained benchmark models on EuroSAT in RGB space and RGB-NIR space. Here our method is auto-generated architectures discovered on NWPU-RESISC45 which are evaluated on EuroSAT dataset. Boldface is the best result.

Model	Accuracy (%)		Size	Method	
	RGB	RGB-NIR			
RestNet-50 [34]	98.57	–	25.6M	Fine-Tuning (Handcrafted)	
GoogLeNet [34]	98.18	–	6.8M		
EfficientNet-B0 [41] (baseline)	<b>98.85</b>	98.78	5.3M	Fine-Tuning (Auto-Generated)	
Our method ( $C=16$ )	$B=[3, 2, 1, 2]$	98.52	<b>99.76</b>	5.5M	Auto-Generated (Random Search)
	$B=[2, 3, 3, 1]$	98.31	99.54	5.4M	
	$B=[2, 3, 2, 1]$	98.67	99.63	5.1M	
	$B=[3, 4, 4, 1]$	98.61	99.59	5.5M	

### 4.3.7 Evaluating on BigEarthNet Dataset

We also evaluated the architecture transferability of networks discovered on a single-label RGB aerial image dataset to a multi-label multispectral satellite image dataset. The best 4 architectures discovered on NWPU-RESISC45 (Table 4.2) single-label RGB aerial image dataset were evaluated on BigEarthNet-19 multi-label multispectral satellite image dataset. BigEarthNet-19 is a large-scale Sentinel-2 dataset with 19 classes. Since the images in BigEarthNet are of size  $120 \times 120$  pixels, we did not apply the first pooling layer in the large-scale architecture settings (see Fig. 4.1). This slightly increased the size of our networks, which can be observed by comparing the model size in Table 4.2 and Table 4.7. Each architecture was trained from scratch with a linear classifier of 19 classes. All training settings remained the same as applied on NWPU-RESISC45. In addition to classification accuracy ( $Acc$ ), we reported  $F_1$  score, Recall ( $R$ ) and Precision ( $P$ ) metrics as described in Sumbul *et al.* [36].

In Table 4.7, we compared our results with the state-of-the-art handcrafted CNN models on the RGB images of BigEarthNet in [36]. Our best network achieved multi-label classification accuracy of 93.49%. The best 4 networks we trained on BigEarthNet achieved an average of 93.42% multi-label classification accuracy. Using fewer parameters, our best networks significantly improved the performance of the state-of-the-art handcrafted CNN models on BigEarthNet [36]. An improvement of approximately 9%  $F_1$  score, 8% recall and 4% precision was achieved over ResNet50, K-Branch CNN and ResNet152 respectively. Moreover, for further comparative evaluation, we fine-tuned the baseline network on the RGB-NIR images of BigEarthNet. And as shown in Table 4.7, in the RGB-NIR space, the baseline trailed behind our best network by approximately 0.9% in multi-label classification accuracy, 3.9% in  $F_1$  score, 3.5% in recall metric and 4.5% in precision metric. An example of the BigEarthNet RGB images with the true multi-labels and the predicted

multi-labels by our best network is shown in Fig. 4.9.







Image	True Multi-Label	Predicted Multi-Label	Image	True Multi-Label	Predicted Multi-Label
	Arable land; Inland waters; Coniferous forest; Mixed forest	Arable land; Inland waters; Coniferous forest; Mixed forest; <b>Land principally occupied by agriculture</b>		Natural grassland and sparsely vegetated areas; Broad-leaved forest; Transitional woodland-shrub	Natural grassland and sparsely vegetated areas; Broad-leaved forest; Transitional woodland-shrub
	Arable land; Urban fabric; Complex cultivation patterns; Land principally occupied by agriculture; Pastures	Arable land; Urban fabric; Complex cultivation patterns; Land principally occupied by agriculture; Pastures		<b>Coniferous forest</b> ; Broad-leaved forest; Inland waters; Land principally occupied by agriculture; Mixed forest; Pastures	Broad-leaved forest; Land principally occupied by agriculture; Mixed forest; Pastures; <b>Transitional woodland-shrub</b>
	Agro-forestry areas; Pastures; Transitional woodland-shrub; <b>Arable land</b> ;	Agro-forestry areas; Pastures; Transitional woodland-shrub		Complex cultivation patterns; Arable land; Mixed forest; Pastures; Urban fabric	Complex cultivation patterns; Arable land; Mixed forest; Pastures; Urban fabric

Figure 4.9: An example of the BigEarthNet RGB images with the true multi-labels and the predicted multi-labels by our best network. The green labels are true labels correctly predicted; blue labels are true labels that were not predicted; and red labels are wrongly predicted labels.

Table 4.7: Comparisons between our method and the baseline network and the state-of-the-art handcrafted CNN models on BigEarthNet in RGB space and RGB-NIR space. Here our method is auto-generated architectures discovered on NWPU-RESISC45 which are evaluated on BigEarthNet dataset. The results of the state-of-the-art handcrafted models are reported in [36]. Boldface is the best result.

Band	Model	Acc (%)	$F_1$ (%)	$R$ (%)	$P$ (%)	Size	Method	
RGB	VGGNet-16 [58]	–	76.01	75.85	81.05	138M	Handcrafted	
	VGGNet-19 [58]	–	75.96	76.71	79.87	143M		
	ResNet-50 [37]	–	77.11	77.44	81.39	25.6M		
	ResNet-101 [37]	–	76.49	77.45	80.18	44.5M		
	ResNet-152 [37]	–	76.53	76.24	81.72	60.2M		
	K-Branch CNN [59]	–	72.73	78.96	71.61	–		
	Our method ( $C=16$ )	$B=[3, 2, 1, 2]$	93.34	86.36	86.76	<b>86.32</b>	5.7M	Auto-Generated (Random Search)
		$B=[2, 3, 3, 1]$	93.40	85.84	<b>86.78</b>	85.84	5.6M	
		$B=[2, 3, 2, 1]$	93.43	<b>86.53</b>	85.45	86.26	5.3M	
		$B=[3, 4, 4, 1]$	<b>93.49</b>	86.30	86.75	85.99	5.8M	
	EfficientNet-B0 [41] (baseline)	93.03	85.29	84.69	85.68	5.3M	Fine-Tuning (Auto-Generated)	
RGB-NIR	Our method ( $C=16$ )	$B=[3, 2, 1, 2]$	93.66	88.09	87.60	88.58	5.7M	Auto-Generated (Random Search)
		$B=[2, 3, 3, 1]$	93.87	88.18	87.69	88.67	5.6M	
		$B=[2, 3, 2, 1]$	<b>93.89</b>	<b>89.21</b>	<b>88.22</b>	<b>90.23</b>	5.3M	
		$B=[3, 4, 4, 1]$	93.87	88.55	87.98	89.13	5.8M	

## 4.4 Discussion

Automated architecture search methods would enable the domain scientists to apply CNN techniques with little or no deep learning experience. In this paper, we have introduced SLGE-based random search with early-stopping strategy to automatically search for efficient CNN architectures for both single-label and multi-label RS scene classification tasks. We experimented the proposed method on RGB aerial image datasets, NWPU-RESISC45 and AID, and on multispectral satellite image datasets, EuroSAT and BigEarthNet. Through comparing and analysing the experimental results

with the state-of-the-art transfer learning (feature extraction and fine-tuning) models and the baseline network, we evaluated the effectiveness of the proposed SLGE-based random search with early-stopping strategy for RS scene classification. The results showed that the proposed automated architecture search method discovered architectures which could offer a promising performance in classifying RS image scenes, particularly multispectral satellite image scenes.

As shown in Table 4.6, in the 4-band RGB-NIR space of EuroSAT dataset, our best discovered network outperformed the baseline network by approximately 1.0% classification accuracy rate. Whereas in the RGB space of EuroSAT, our best discovered network achieved a competitive performance that was only 0.2% lower than the baseline. Clearly, since the baseline was pre-trained on RGB images (ImageNet), this might be the reason why it showed 0.2% accuracy rate over our best discovered network in the RGB space of EuroSAT, but trailed behind our best discovered network in the RGB-NIR space by 1.0%. Moreover, in RGB-NIR space of BigEarthNet multi-label dataset, our best discovered network gained a performance boost in  $F_1$  score, recall and precision metrics with respect to the state-of-the-art models reported in [36], which is relatively higher compared to fine-tuning the baseline. Table 4.7 shows that in the RGB-NIR space of BigEarthNet, our best discovered network significantly improved the performance of the state-of-the-art models on BigEarthNet by approximately 12.1% in  $F_1$  score, 9.2% in recall and 8.5% in precision as compared to the baseline which showed approximately 8.2%, 5.7% and 4.0% in  $F_1$  score, recall and precision respectively. Specifically, in Table 4.7, our best discovered network outperformed the baseline by  $F_1$  score of 3.9% , recall of 3.5% and precision of 4.5% in the RGB-NIR space of BigEarthNet. Even in case using 4-band RGB-NIR space of multispectral image dataset, the proposed SLGE-based method achieved relatively better performance in RS image classification task.

The performance comparisons on RGB aerial image datasets (NWPU-RESISC45 and AID) indicate that the proposed SLGE-based method competed with the state-of-the-art transfer learning models and the baseline network. For example, in Table 4.3, with about 91% less memory footprint, our best discovered network achieved 0.2% improvement of classification accuracy over the best feature extraction method, DLGFF [44], and approximately 1.3% improvement over the best fine-tuned model, Fusion of VGGNet-16 & CaffeNet [51]. Also, in Table 4.4, the best discovered network improved the performance of the fine-tuned models by approximately 0.8% classification accuracy rate, and competed with the state-of-the-art feature extraction methods. However, it can be seen from the confusion matrices Fig. 4.5 and Fig. 4.7 that the proposed method demonstrated a poor performance on some scene classes, particularly, *palace* and *church* classes in NWPU-RESISC45 (see Fig. 4.6), and *bareland*, *desert*, *square* and *centre* classes in AID (see Fig. 4.8). The reason is most of the examples of these classes exhibit very high inter-class similarities in textural and structural characteristics. Searching for CNNs that can solve this problem is one of the future

research topics.

## 4.5 Summary

In this chapter, we extended SLGE to classification of remote sensing image scenes. Using SLGE-based architecture representation, we explored random search with early-stopping strategy as an architecture search technique to automatically search for CNNs architectures for remote sensing scene classification task. The experimental results on four remote sensing scene classification datasets, two RGB aerial image benchmarks (NWPU-RESISC45 and AID) and two multispectral satellite image benchmarks (EuroSAT and BigEarthNet), suggest that the auto-generated CNN models demonstrated a promising performance in classifying multispectral satellite image scenes. With fewer parameters, the best discovered networks significantly improved the state-of-the-art models on EuroSAT single-label and BigEarthNet multi-label classification tasks. The next chapter extends to SLGE to aerial/satellite image segmentation task using evolutionary algorithm as a search strategy.

## 4.6 References

- [1] G. Cheng, J. Han, and X. Lu, "Remote sensing image scene classification: Benchmark and state of the art," *Proceedings of the IEEE*, vol. 105, no. 10, pp. 1865–1883, 2017.
- [2] S. S. Durbha, K. R. Kurte, and U. Bhangale, "Semantics and High Performance Computing Driven Approaches for Enhanced Exploitation of Earth Observation (EO) Data: State of the Art," *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, vol. 87, no. 4, pp. 519–539, 2017.
- [3] S. Smillie, "Automatic target recognition: Some considerations," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-2, no. 2, pp. 187–191, 1966.
- [4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [5] X. X. Zhu, D. Tuia, L. Mou, G. Xia, L. Zhang, F. Xu, and F. Fraundorfer, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 8–36, 2017.
- [6] J. Song, S. Gao, Y. Zhu, and C. Ma, "A survey of remote sensing image classification based on CNNs," *Big Earth Data*, vol. 3, no. 3, pp. 232–254, 2019.
- [7] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, "Deep learning in remote sensing applications: A meta-analysis and review," *Journal of Photogrammetry and Remote Sensing*, vol. 152, pp. 166–177, 2019.
- [8] R. Stivaktakis, G. Tsagkatakis, and P. Tsakalides, "Deep learning for multilabel land cover scene categorization using data augmentation," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 7, pp. 1031–1035, 2019.
- [9] Y. Liu and C. Huang, "Scene classification via triplet networks," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 1, pp. 220–237, 2017.
- [10] G. Sumbul and B. Demir, "A deep multi-attention driven approach for multi-label remote sensing image classification," *IEEE Access*, vol. 8, pp. 95 934–95 946, 2020.

- 
- [11] Y. Chen, K. Zhu, L. Zhu, X. He, P. Ghamisi, and J. A. Benediktsson, “Automatic design of convolutional neural network for hyperspectral image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 9, pp. 7048–7066, 2019.
- [12] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, “Efficient architecture search by network transformation,” in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, p. 2787–2794.
- [13] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, “Practical block-wise neural network architecture generation,” in *Proceedings of the IEEE Conference on CVPR*, 2018, pp. 2423–2432.
- [14] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” in *Proceedings of the AAAI*, 2019.
- [15] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, “Completely automated CNN architecture design based on blocks.” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2019.
- [16] H. Liu, K. Simonyan, and Y. Yang, “DARTS: Differentiable architecture search,” in *Proceedings of the International Conference on Learning Representations*. ICLR, 2019.
- [17] S. Xie, H. Zheng, C. Liu, and L. Lin, “SNAS: stochastic neural architecture search,” in *Proceedings of the International Conference on Learning Representations*, 2019.
- [18] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, “Progressive neural architecture search,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [19] L.-C. Chen, M. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens, “Searching for efficient multi-scale architectures for dense image prediction,” in *Advances in neural information processing systems*, 2018, pp. 8699–8710.
- [20] S. Xie, A. Kirillov, R. Girshick, and K. He, “Exploring randomly wired neural networks for image recognition,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.
- [21] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.
- [22] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, “Google vizier: A service for black-box optimization,” in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1487–1495.
- [23] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, J. Ben-tzur, M. Hardt, B. Recht, and A. Talwalkar, “A System for Massively Parallel Hyperparameter Tuning,” in *Proceedings of Machine Learning and Systems*, vol. 2, 2020, pp. 230–246.
- [24] L. Li and A. Talwalkar, “Random search and reproducibility for neural architecture search,” in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. UAI, 2019.
- [25] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann, “Evaluating the search phase of neural architecture search,” in *Proceedings of the International Conference on Learning Representations*, 2020.
- [26] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2018, pp. 8697–8710.
- [27] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, “Efficient neural architecture search via parameters sharing,” in *Proceedings of the 35th ICML*, vol. 80, 2018, p. 4095–4104.
- [28] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, “Neural architecture optimization,” in *Advances in neural information processing systems*, 2018, pp. 7816–7827.

- [29] F. J. Solis and R. J. B. Wets, "Minimization by random search techniques," *Mathematics of Operations Research*, vol. 6, no. 1, p. 19–30, 1981.
- [30] Z. B. Zabinsky, *Random Search Algorithms*. American Cancer Society, 2011.
- [31] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proceedings of the 5th International Conference on Learning Representations*, 2017.
- [32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.
- [33] G. Xia, J. Hu, F. Hu, B. Shi, X. Bai, Y. Zhong, L. Zhang, and X. Lu, "Aid: A benchmark data set for performance evaluation of aerial scene classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3965–3981, 2017.
- [34] P. Helber, B. Bischke, A. Dengel, and D. Borth, "Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 7, pp. 2217–2226, 2019.
- [35] G. Sumbul, M. Charfuelan, B. Demir, and V. Markl, "Bigearthnet: A large-scale benchmark archive for remote sensing image understanding," in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 2019, pp. 5901–5904.
- [36] G. Sumbul, J. Kang, T. Kreuziger, F. Marcelino, H. Costa, P. Benevides, M. Caetano, and B. Demir, "Bigearthnet dataset with a new class-nomenclature for remote sensing image understanding," 2020, arXiv:2001.06372v2 [cs.CV].
- [37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [38] J. Howard and S. Gugger, "Fastai: A layered api for deep learning," *Information*, vol. 11, no. 2, 2020.
- [39] L. N. Smith, "A disciplined approach to neural network hyperparameters: Part1–learning rate, batch size, momentum, and weight decay," 2018, arXiv:1803.09820 [cs.LG].
- [40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.
- [41] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, 2019, pp. 6105–6114.
- [42] S. Jiang, H. Zhao, W. Wu, and Q. Tan, "A novel framework for remote sensing image scene classification." *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 42, no. 3, 2018.
- [43] J. Chen, H. Huang, J. Peng, J. Zhu, L. Chen, W. Li, B. Sun, and H. Li, "Convolution neural network architecture learning for remote sensing scene classification," 2020, arXiv:2001.09614 [cs.CV].
- [44] Q. Zhu, Y. Zhong, Y. Liu, L. Zhang, and D. Li, "A deep-local-global feature fusion framework for high spatial resolution imagery scene classification," *Remote Sensing*, vol. 10, no. 4, p. 568, 2018.
- [45] G. Cheng, Z. Li, X. Yao, L. Guo, and Z. Wei, "Remote sensing image scene classification using bag of convolutional features," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 10, pp. 1735–1739, 2017.

- 
- [46] Y. Yu and F. Liu, "Dense connectivity based two-stream deep feature fusion framework for aerial scene classification," *Remote Sensing*, vol. 10, no. 7, p. 1158, 2018.
- [47] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han, "When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative cnns," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 5, pp. 2811–2821, 2018.
- [48] J. Li, D. Lin, Y. Wang, G. Xu, and C. Ding, "Deep discriminative representation learning with attention map for scene classification," 2019, arXiv:1902.07967 [cs.CV].
- [49] Y. Guo, J. Ji, X. Lu, H. Huo, T. Fang, and D. Li, "Global-local attention network for aerial scene classification," *IEEE Access*, vol. 7, pp. 67 200–67 212, 2019.
- [50] R. Dong, D. Xu, L. Jiao, J. Zhao, and J. An, "A fast deep perception network for remote sensing scene classification," *Remote Sensing*, vol. 12, no. 4, p. 729, 2020.
- [51] Y. Yu and F. Liu, "Aerial scene classification via multilevel fusion based on deep convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 2, pp. 287–291, 2018.
- [52] Y. Li, Q. Wang, X. Liang, and L. Jiao, "A novel deep feature fusion network for remote sensing scene classification," in *IEEE International Geoscience and Remote Sensing Symposium*, 2019, pp. 5484–5487.
- [53] Y. Yu and F. Liu, "A two-stream deep fusion framework for high-resolution aerial scene classification," *Computational Intelligence and Neuroscience*, p. 8639367, 2018.
- [54] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.
- [55] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
- [56] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2261–2269.
- [57] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.
- [58] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations*, 2015.
- [59] G. Sumbul and B. Demir, "A novel multi-attention driven system for multi-label remote sensing image classification," in *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2019, pp. 5726–5729.

## Chapter 5

# Extending SLGE to Aerial/Satellite Image Segmentation

*Evolution continually innovates, but at each level it conserves the elements that are recombined to yield the innovations.*

—John H. Holland

(1929–2015)

The work presented in this chapter extended the SLGE-based architecture representation to semantic segmentation task of RS image understanding. By improving the SLGE representation space, we construct a two-separate search space representation: one for *normal* cells and the other for *atrous spatial pyramid pooling* (ASPP) cells. Using evolutionary algorithm with genetic modification mechanisms such as uniform mutation, two-point crossover and gene crossover, we joint search for a normal cell and an ASPP cell as a pair of cells to build a modularized encoder-decoder CNN architecture, which we called SLGENet, for solving aerial or satellite image semantic segmentation problem. Three ISPRS benchmarks are employed to verify the performance of the SLGENet in aerial image segmentation tasks. In this regard, we validate the effectiveness and robustness of SLGE representation space.

### 5.1 Introduction

Aerial or satellite image segmentation (also known as land cover classification) aims to assign semantic labels to each pixel in an aerial or a satellite image based on their spatial and/or spectral information. This allows for smart identification and classification of land use, with numerous

---

Parts of this chapter is published in Broni-Bediako *et al.*, 2021 (see Appendix D). Reprinted with permission from the publisher.



applications including urban planning, land resource management and environmental monitoring [1–3].

Semantic segmentation of aerial imagery dates back to the 1960s [4] and it remains an important research problem in the field of remote sensing (RS) image analysis [5]. Traditionally, boundary-based and region-based algorithms [6–8], and handcrafted feature extraction methods with support vector machine and artificial neural network (ANN) classifiers [9–12] are employed for aerial image semantic segmentation. Convolutional neural networks (CNNs) methods [13], which have achieved remarkable performance in the field of computer vision [14], have recently emerged as the leading approach in RS image analysis [15] including aerial image segmentation [16–18]. Especially, fully convolutional networks (FCNs) [19] have achieved significant results in aerial image segmentation [20–22]. The vast majority of current works [23–27] on semantic segmentation of high-resolution aerial and satellite images adopt the encoder-decoder structures, for example U-Net [28] and SegNet [29], which efficiently extract long-range context features to improve the accuracy of boundary segmentation. Other studies [30–33] employed pre-trained networks such as ResNet [34], DenseNet [35] and Xception [36] as backbone networks, and dense conditional random fields (CRFs) or atrous spatial pyramid pooling (ASPP) modules are used on top of those networks to extract multi-scale context features for semantic segmentation. These methods have also achieved a remarkable performance in aerial and satellite image semantic segmentation [37–40]. Refer to section 2.4.2 of chapter 2 for review on RS image segmentation with CNNs.

### 5.1.1 Automated NAS with Evolutionary Algorithms

Evolutionary NAS is a bio-inspired automated NAS approach that imitates the basic principles of biological evolution to automate the architecture design for neural networks [41]. Evolutionary NAS has been of interest in the AI community for three decades [42], dates back to the 80s when Miller *et al.* [43] first used genetic algorithm to evolve simple ANN architectures and optimized their parameters with a gradient-based method. Most of the earlier works evolve both the network architecture and its parameters at a small scale [41]. Since 2017, the growing interest in automated CNN architecture design [44, 45] has seen classical evolutionary algorithms, such as genetic algorithms [46] and genetic programming [47], also being exploited in the search for CNNs architectures for image classification in the field of computer vision [48]. Besides achieving promising performance [49–55], they also consume fewer computational resources than their competitor, reinforcement-based NAS [56, 57]. Furthermore, they have proven to be competitive with handcrafted architectures of human experts [58]. Galván and Mooney [48] and Baldominos *et al.* [42] have provided comprehensive survey on recent evolutionary NAS methods.

## 5.2 Methodology

Using the formulated architecture search problem presented in section 3.3.2 of chapter 3, we search for modularized encoder-decoder CNN architectures for aerial or satellite image segmentation task using evolutionary algorithm search strategy introduced in gene expressing programming [59]. This section describes SLGE representation of a two-separate search spaces for *normal* and *ASPP* cells that are based on the standard SLGE representation presented in chapter 3 and evolutionary NAS components: fitness evaluation, selection and reproduction, and evolutionary search process.

### 5.2.1 Search Space

We adopt a pre-defined network structure illustrated in Fig. 5.1 and a cell-based search space is designed to induce a wide range of *normal* cells [57] and *ASPP* cells [60] to evolve modularized encoder-decoder CNNs architectures. The *normal* cells and *ASPP* cells are respectively used to build the encoder and the decoder of the modularized encoder-decoder CNN architectures. As illustrated in Fig. 5.1, the pre-defined network structure begins with a fixed stem of two-layer  $3 \times 3$  convolution, each of which reduces the spatial resolution of an input image by a factor of 2 and doubles the output channels. Then, it is followed by normal cells, repeatedly stacked four times, to build the network backbone. Each normal cell block differs in spatial resolution by a downsampled factor of 2 and output channels by double. An ASPP cell is attached to each normal cell of network backbone feature map to extract multi-scale information of the image. The output of each ASPP cell is bilinear upsampled to the original resolution of the input image. They are then summed to produce the semantic labelling prediction of the image.

The normal cell and the ASPP cell are encoded as a pair of linear string structures (see Fig. 5.2) based on the SLGE representation introduced in chapter 3. Then the mapping function presented in section 3.2.3 of chapter 3 is used to translate the pair of the linear structures (see Fig. 5.2) into a pair of cells (see Fig. 5.3), which is used to build a final modularized encoder-decoder CNN architecture in Fig. 5.1. The set of possible operations in the normal cell consists of:

- $3 \times 3$  and  $5 \times 5$  depthwise-separable convolutions,
- $3 \times 3$  average pooling and max pooling,

and in the ASPP cell:

- $1 \times 1$  convolution,
- $3 \times 3$  atrous depthwise-separable convolution with rate  $r$ , where  $r \in \{1, 2, 3, \dots, 9\}$ , and

- average spatial image pooling with spatial size  $s$ , where  $s \in \{1, 2, 4\}$ . In each pooling operation, a  $1 \times 1$  convolution is applied after the average pooling, then followed by bilinear upsampling to the spatial resolution of the input tensor.

These are very common convolution operations in modern CNNs [31, 36]. All the operations have appropriate strides and paddings applied to preserve the spatial resolution of the feature maps. Since SLGE implicitly encodes the shortcut (or skip) connections (see section 3.2.1 of chapter 3), the identity operation which is employed in other studies [58, 61] to enable shortcut connections in a network is not explicitly included in the set of possible operations. The complexity of the search space is the sum of the complexity of normal cell search space and ASPP cell search space. Refer to chapter 3 for further details of SLGE representation space.

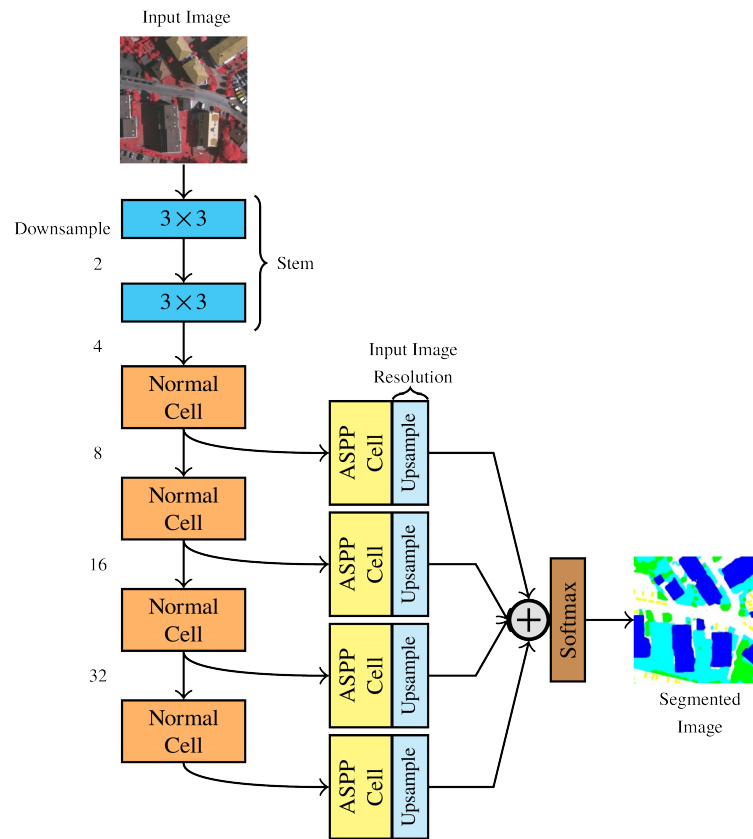


Figure 5.1: A network structure diagram of the modularized encoder-decoder CNN architecture. The normal cell and ASPP cell described in section 5.2.1 are used in the encoder and the decoder respectively.

## 5.2.2 Fitness Evaluation

The evaluation function aims to assign fitness value to individuals with respect to how well they adapt to the environment in consideration. It forms the basis for selecting individuals as parents to reproduce offspring for the next generation. The higher the fitness value, the more likely the

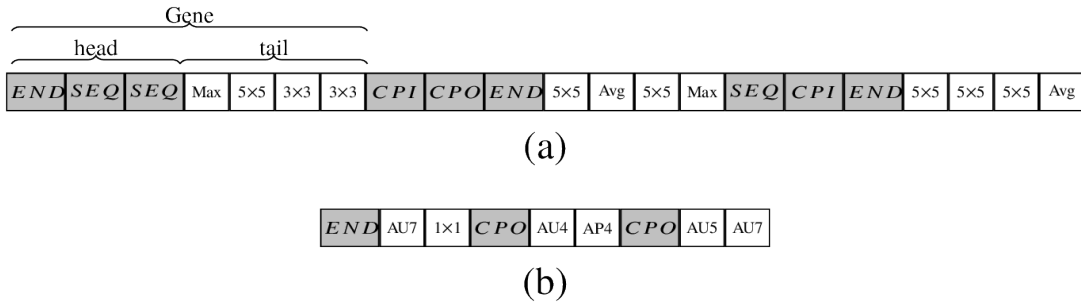


Figure 5.2: Schematic representation of SLGE pair of chromosomes. The pair of chromosomes of the best network, SLGENet-3, in Tables 5.3, 5.4 and 5.5. (a) A chromosome of 3 genes with head length of 3 for a normal cell (see Fig. 5.3a). The head of a gene may consists of *SEQ*, *CPO*, *CPI* or *END* (CE graph transformation functions), and a tail of operations defined for normal cell. (b) A chromosome of 3 genes with head length of 1 for ASPP cell (see Fig. 5.3b). The head of gene may be *CPO* or *END* with a tail which consists of possible operations defined for ASPP cell. The ‘AU4’, ‘AU5’ and ‘AU7’ are atrous depthwise-separable convolution with rates 4, 5 and 7 respectively; and ‘AP4’ is an average spatial pooling with size 4. See section 5.2.1 for the operations defined for normal cell and ASPP cell.

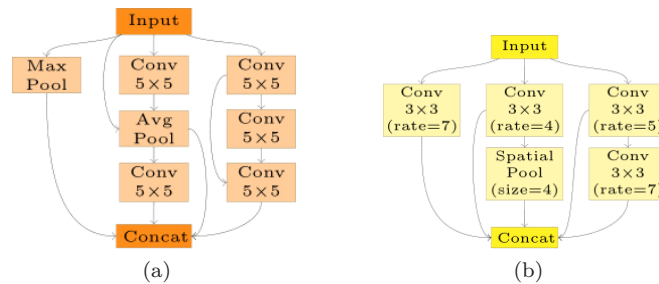


Figure 5.3: Evolutionary discovered pair of cells of the best network, SLGENet-3, in Tables 5.3, 5.4 and 5.5. The (a) normal cell and (b) ASPP cell are the phenotypes of the chromosomes illustrated in Fig. 5.2a and Fig. 5.2b respectively. (a) The convolution operations without successor are depthwise concatenated to produce an output, and if an operation has more than one predecessor, the feature maps of the predecessors are added together. (b) The feature maps of all branches within the cell are depthwise concatenated to produce an output.

individual will have progeny and survive into the next generation. The function is composed from the quality measure in the phenotype space of the individuals with respect to the task at hand. Since the proposed method is interested in solving aerial image segmentation problem, the best strategy to assign fitness to an individual is the pixel-wise classification accuracy of the architecture decoded by the individual.

The fitness evaluation of individuals is presented in Algorithm 6. In the two-separate SLGE representation, an individual constitutes a pair of chromosomes which consists of a chromosome for normal cell and a chromosome for ASPP cell as defined in section 5.2.1. Algorithm 6 encapsulates the objective function  $\mathcal{F}$  (see Equation 3.6) and the decoding algorithm (see Algorithm 4) presented in chapter 3. During fitness evaluation, the decoding algorithm translates the information encoded in the pair of chromosomes of an individual into pair of cells (normal and ASPP) to build the corresponding modularized encoder-decoder CNN architecture (see Fig. 5.1). The architecture is

then trained based on the objective function  $\mathcal{F}$  for a fixed number of epochs. The objective function  $\mathcal{F}$  aims to find an architecture that maximize pixel-wise classification accuracy rate on a validation dataset, subject to model size constraint  $\mathcal{T}_{params}$ . The loss function in the training function  $\mathcal{L}$  of the objective function  $\mathcal{F}$  is used to train the architecture via back-propagation on the training set. Then, the performance of the architecture on the validation set is assigned as fitness to its corresponding individual in the genotype space, where genetic modification may occur to reproduce offspring for the next generation. We present the training details of architectures in section 5.3.2.

---

**Algorithm 6** Fitness evaluation of individual candidate architectures.

---

**input :** Population  $P$  of individuals (pairs of chromosomes), a training data  $t\mathcal{D}$ , and a validation data  $v\mathcal{D}$

**output:** Population  $P$  with fitness of the individuals

- 1 **for** each individual in  $P$  **do**
  - 2   Transform the information encoded in the pair of chromosomes into pair of cells using Algorithm 4
  - 3   Build a CNN architecture (see Fig. 5.1) with the cells, such that the number of parameters  $\theta \leq \mathcal{T}_{params}$
  - 4   Train the CNN architecture on  $t\mathcal{D}$  via the objective function  $\mathcal{F}$  in Equation 3.6
  - 5   Evaluate the pixel-wise classification accuracy of the trained CNN on  $v\mathcal{D}$
  - 6   Assign the accuracy of the CNN as the fitness of the *individual*
  - 7 **end for**
  - 8 **return**  $P$  with fitness
- 

### 5.2.3 Selection and Reproduction

In SLGE representation, individuals are selected as parents of the next generation according to the luck of the draw by roulette-wheel sampling [46] on their fitness values. We adopt simple elitism approach [62] as the survivor selection mechanism. In this way, the top individual of each generation is survived and reproduced without modification. Explicitly replicating the top individual into the next generation guarantees that the best trait never extinct in SLGE as the evolution progresses. Thus, together with the parent selection scheme, a continuous improvement may be achieved. During reproduction, the replication of the top individual occurs before parents are selected and modified to create offspring for the next generation.

To modify a chromosome of an individual for reproduction, we adopt three genetic operators: uniform mutation, two-point crossover and gene crossover. The uniform mutation and two-point crossover are standard genetic operators [46]. Gene crossover operator was first introduced in GEP [59]. The mutation is applied randomly at a rate  $p_m$  to all genes in the selected chromosome (see Algorithm 7). The mutation rate  $p_m$  is the same as the two one-point mutations per chromosome used in Ferreira [59], which is expressed as 2 divided by (*# of genes*  $\times$  *gene length*).

Since SLGE chromosomes are fixed-length strings, applying the two-point crossover and gene crossover is very simple, just as in classical genetic algorithm [46]. As clearly illustrated in Fig. 5.4, in the two-point crossover operation, two selected parent chromosomes are paired side by side and

**Algorithm 7** The mutation operation in SLGE representation

**input :** An individual *indv* with a pair of chromosomes, and a CE graph transformation functions set *fset* (see section 3.1.2 of chapter 3) and operation sets *pset* as defined in section 5.2.1

**output:** Offspring (i.e a mutated individual *indv*)

```

1 for each chromosome in indv do
2    $p_m \leftarrow 2$  divided by (# of genes  $\times$  gene length) //Calculating
   the mutation rate  $p_m$ 
3 for each gene in chromosome do
4   Randomly pick a value  $r$  between 0 and 1
5   if  $r < p_m$  then
6     for each function in gene head do
7       Randomly select a transformation function from fset to mutate the gene head
8     end for
9     for each operation in gene tail do
10      Randomly select an operation from pset to mutate the gene tail
11    end for
12  end if
13 end for
14 end for
15 return Return the mutated individual indv

```

randomly split at two points. Then, the elements in between the split points are exchanged between the parents to create two new offspring. Gene crossover operation is a two-point crossover where the elements in between the split points constitute a gene. That is, in gene crossover operation, an entire gene is exchanged between two parent chromosomes to create two new offspring. Fig. 5.5 is an illustration of gene crossover operation. The two-point crossover constantly destroys old building-blocks and creates new ones during reproduction [59]. To reduce the disruptive effects on the genotype space, the selected parent individuals are modified as follows: chromosomes for normal cells undergo both mutation and crossover operations, whereas chromosomes for ASPP cells undergo only mutation modification. We use two-point crossover rate of 0.6 and gene crossover rate of 0.3. These values are defaults crossover rates in GEP [59].

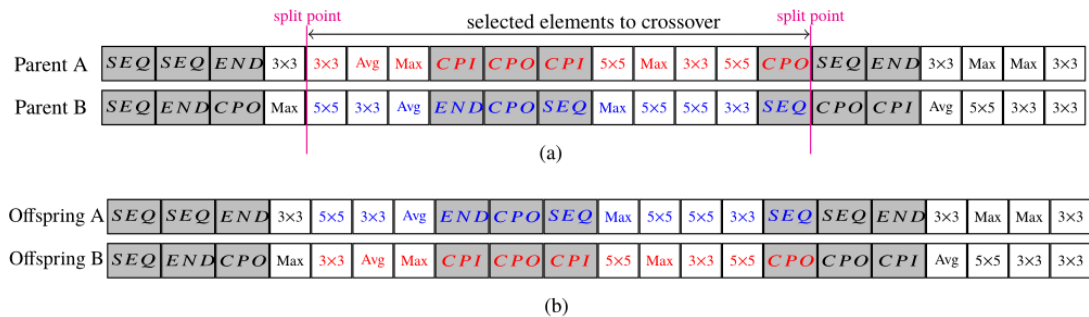


Figure 5.4: Illustration of two-point crossover operation. (a) Two selected parent chromosomes for the crossover operation. (b) The generated offspring. The red elements in Parent A are exchanged with the blue elements in Parent B to generate Offspring A and B respectively.

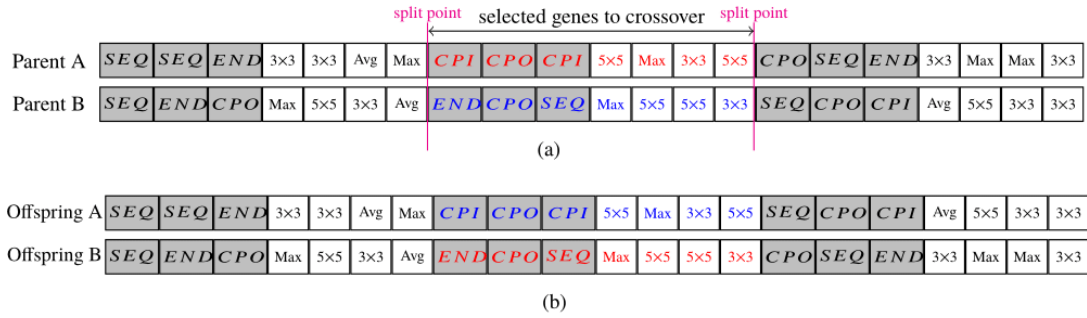


Figure 5.5: Illustration of gene crossover operation. (a) Two selected parent chromosomes for the crossover operation. (b) The generated offspring. The gene with red elements in Parent A are exchanged with the gene with blue elements in Parent B to generate Offspring A and B respectively.

## 5.2.4 Evolutionary Search Process

The overview of SLGE evolutionary architecture search process is schematically presented in Fig. 5.6. Like all evolutionary neural architecture search methods, the proposed method works with populations of individuals which must be created initially in order to begin the process. The initial population of  $N$  individuals is randomly generated based on the SLGE representation algorithm presented in Algorithm 3 (see section 3.2 of chapter 3). As mentioned in section 5.2.2, every individual in the population has a pair of chromosomes (i.e. a chromosome for normal cell and a chromosome for ASPP cell as defined in section 5.2.1). The subsequent populations are progenies of the initial population via genetic modification.

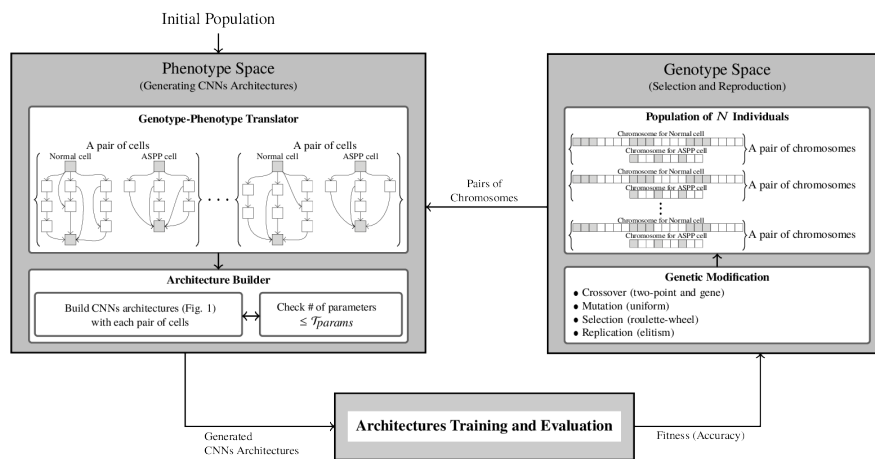


Figure 5.6: The overview of SLGE evolutionary neural architecture search.

After the population is initialized, each individual (pair of chromosomes) in the population is then transformed respectively into pairs of cells (normal and ASPP) using the mapping (decoding) algorithm presented in Algorithm 4 (see section 3.2 of chapter 3). These pairs of cells are then used to build modularized encoder-decoder CNNs architectures (see Fig. 5.1), subject to the

maximum number of parameters constraint. The architectures are then trained and evaluated based on Algorithm 6 (see section 5.2.3), to determine the fitness of their corresponding individuals. Individuals are then selected according to their fitness, which is the performance of pixel-wise classification accuracy, to reproduce offspring for the next generation as described in section 5.2.3. Individuals which have higher fitness value are more likely to have progeny and survive into the next generation. The offspring are then subjected to the same evolutionary process: transformation of the individuals into CNNs architectures, trained to determine their fitness, selection, and reproduction with genetic modification. This process is repeated for a certain the number of generations.

## 5.3 Experiments

The experiments are aimed to verify whether the SLGE evolutionary architecture search is able to find CNNs architectures which can achieve a promising performance on aerial or satellite image semantic segmentation tasks. By so doing, we validate the effectiveness and robustness of SLGE representation space.

### 5.3.1 Datasets

The ISPRS Vaihingen and Potsdam datasets consist of very high resolution aerial images over Vaihingen and Potsdam respectively (both in Germany) for the 2D semantic labelling contest of ISPRS WG III/4<sup>1</sup>. The images in Vaihingen and Potsdam datasets are 3-band infrared, red and green (IRRG) and 4-band infrared, red, green, blue (IRRGB) image data respectively, with corresponding digital surface model (DSM) and normalized DSM (nDSM) data. They are densely labelled with 6 classes: buildings, impervious surfaces (e.g. roads), low vegetation, trees, cars and clutter. Overall, there are 33 images (tiles) of average size  $2500 \times 2000$  pixels at ground sampling distance (GSD) of about 9cm in the Vaihingen dataset, and 38 images of size  $6000 \times 6000$  pixels at GSD of about 5 cm in Potsdam dataset. Among them, only 16 images in Vaihingen dataset and 24 images in Potsdam dataset were publicly available for the contest. The benchmark contest ended in 2018 summer, and since then, all the data has been publicly available. Nevertheless, just as it was in the contest, the images which were originally undisclosed in each dataset are used only for testing. And following Marmanis *et al.* [23], we use 4 of the originally disclosed images in each dataset for validation, and the remaining 12 and 20 images of Vaihingen and Potsdam respectively are used for training. In all the experiments, we only use the 3-band IRRG image data in both datasets. To keep consistent with previous works [18, 23, 24, 63–65], the clutter class which is very rare is considered as a reject class, so that the results are reported on buildings, impervious surfaces, low vegetation, trees and cars classes.

<sup>1</sup> <https://www2.isprs.org/commissions/comm2/wg4/benchmark/semantic-labeling/>



The UAVid dataset<sup>2</sup> is ISPRS semantic segmentation benchmark for UAV imagery created by Lyu *et al.* [66] in 2020. The dataset consists of 420 high-resolution UAV images focusing on street scenes. It has been split as 200 for training, 70 for validation and 150 for testing by the authors. Among them, the ground truth of only the training and validation are available, and those of the testing are withheld by the authors for online contest. We use the validation set only for testing, and randomly split the training set into 80/20 ratio for training and validation. The images are of the size  $4096 \times 2160$  or  $3840 \times 2160$  pixels, and are densely labelled with 8 classes: building, road, static car, tree, low vegetation, human, moving car and background clutter. The dataset also contains 42 video sequences captured with 4K high-resolution in oblique views. In this work we only use the image data. Compared to ISPRS Vaihingen and Potsdam datasets, UAVid dataset has more images, but the size of the images in Potsdam are quite large. In terms of the total number of labelled pixels, the Vaihingen and Potsdam datasets are one tenth and one half of the size of UAVid dataset, respectively.

### 5.3.2 Training Details

The size of the aerial images in the datasets is very large and they can not be processed directly in deep networks, because it requires a great amount of GPU memory to store the intermediate feature maps in the networks. We follow the common practice and use a sliding window approach to extract  $256 \times 256$  pixel patches as adopted in Chai *et al.* [18]. We use a stride of 64 pixels for Vaihingen dataset and a stride of 128 pixels for Potsdam dataset. On UAVid, the authors used large patches of  $2048 \times 1024$  pixels with a stride of 1024 by 512 pixels. We opt for  $400 \times 400$  with stride of 200 pixels due to GPU memory constraint. Using such moderate patch sizes and strides allow us to extract more training samples, which acts as data augmentation, in addition to rotation and flipping techniques applied. During testing, we average the prediction scores on the overlapping regions.

The architectures are implemented with PyTorch [67] framework and fastai [68], a PyTorch-based library, is used to train them. The value of initial output channels,  $C$ , determines the size of the networks. To meet the model size target  $\mathcal{T}_{params} = 30M$ , we set  $C = 48$ . All the architectures are trained using 1-cycle policy in Smith [69] with Adam optimizer [70] implemented in fastai. The learning rate is set to go from 0.0004 to 0.01 linearly while the momentum goes from 0.95 to 0.85 linearly in phase one of 1-cycle policy. Then in phase two, the learning rates follows cosine annealing from 0.01 to 0, as the momentum goes from 0.85 to 0.95 with the same annealing, and the weight decay is set to 0.0001. These are default values in 1-cycle policy [69]. The architectures weights are randomly initialized using the policy in He *et al.* [71]. The architectures optimization is

<sup>2</sup><https://uavid.nl/>

performed for a total of 100 epochs using batch size of 10 for both Vaihingen and Potsdam datasets and 4 for UAVid dataset.

### 5.3.3 Searching on Vaihingen dataset

We perform architecture search on the ISPRS Vaihingen dataset for aerial image segmentation. Applying architecture search to semantic segmentation task is computationally expensive, since the architecture search space is large and it takes an excessive amount of time and computation to train each candidate architecture in a large-scale training settings. Therefore, to reduce the search time and computation resources required for the architecture search, we employ a small-scale task as a proxy task for fast-search. This may provide a predictive signal about the performance in a large-scale architecture training settings. Following previous works in computer vision [57, 58, 60], we design a proxy task by employing a small-scale architecture training settings. More specifically, we randomly select half of the training set of Vaihingen dataset, and extract  $128 \times 128$  pixel patches with a stride of 64 pixels, as adopted in Audebert *et al.* [24], for training and evaluation of candidate architectures. In addition, we implement the candidate architectures (see Fig. 5.1) with a smaller initial output channels,  $C = 24$ , and employ early stopping to train them not to convergence. In the experiments, we train each candidate architecture for only 20 epochs with 80/20 training-validation ratio and batch size of 30 using the training details in section 5.3.2.

Based on the evolutionary process in section 5.2.4, we performed the two separate experiments with two different pairs of chromosomes in Table 5.1. Both the population size and the number of generations were set to 20. Using a larger population size with a larger number of generations might yield a better performance, but not without additional computational cost. Notwithstanding, such optimization settings is out of the scope of this work because the current settings, as commonly adopted in the literature [52], can achieve a promising performance (see Table 5.3). It is also worth mentioning that using proxy task in NAS process may generate inaccurate ranking of the architectures, since the networks with fewer parameters may converge faster and produce better results for a few epochs than cumbersome ones [72]. Although the networks with good performance might be abandoned, we adopt the proxy task approach to show that we can find good architectures even with limited computation resources. All the experiments were performed on a single workstation with a 11GB GPU GeForce GTX 1080 Ti running on Linux Mint 19 Cinnamon. It took 2.5 GPU days for each of the two experiments.

In Fig. 5.7a and Fig. 5.7b, we visualize the evolutionary trajectories of the two pairs of chromosomes in order to understand the process of discovering the best CNN architecture. The figures show the fitness statistics of the individuals selected via evolutionary selection in each generation of the experiments. We present the statistics in terms of pixel-wise classification

Table 5.1: The two pairs of chromosomes settings for Normal cell and ASPP cell used in the experiments. Chromosome settings for ASPP cell is the same for both pairs.

Pair of Chromosomes		# of Genes	Head Length	Tail Length
Pair 1	Normal Cell	3	2	3
	ASPP Cell	3	1	2
Pair 2	Normal Cell	3	3	4
	ASPP Cell	3	1	2

accuracy of the best individual and the mean with standard deviation over all individuals in each generation using a dotted line and a solid line with shaded deviation respectively. The best and mean accuracy results increase as the evolution progresses. The sharp increase in the accuracy at the earlier generations is due to the random initialization of the population at the beginning of the evolution process. As the process progresses, the improvement in the accuracy lessens. And it can be observed that the variation in the accuracy of each generation also lessens. This means the evolution is towards a steady state in discovering CNNs architectures on the Vaihingen dataset, and the proposed algorithm seems to converge with the setting of 20 generations.

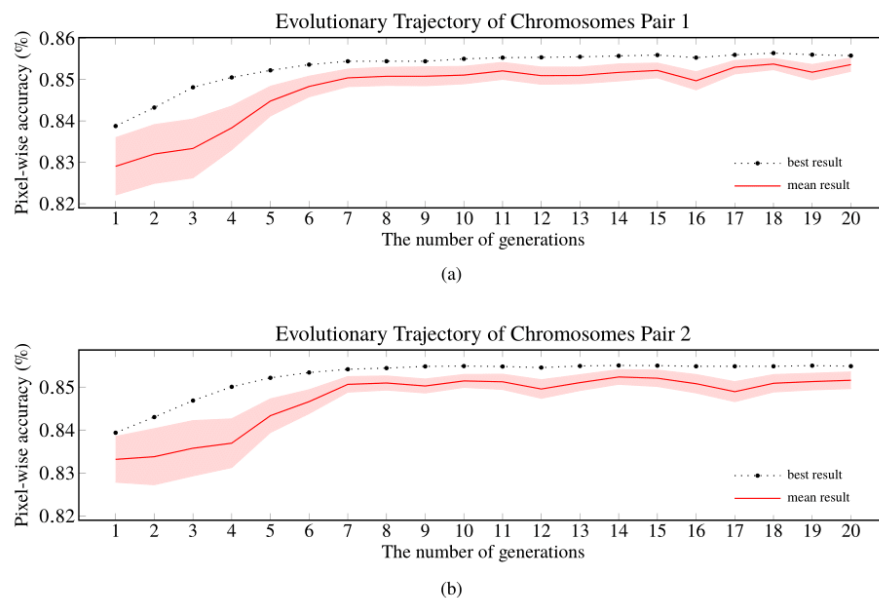


Figure 5.7: Evolutionary trajectories of the proposed method in searching for the best CNN architecture on Vaihingen dataset. (a) Trajectory of the pair of chromosomes ‘Pair 1’ (see Table 5.1). (b) Trajectory of the pair of chromosomes ‘Pair 2’ (see Table 5.1).

### 5.3.4 Baseline

We performed a simple random search on the Vaihingen dataset as a baseline for the study. Here we seek to compare SLGE evolutionary approach against random search to verify the effectiveness of the SLGE search space, since a simple random search is a competitive search strategy [72] that

is generally used in automated neural architecture search as a baseline [58, 61]. We randomly generated ten individuals, five each from the two pairs of chromosomes settings in Table 5.1, to build ten architectures with no evolutionary process. This means architectures in the baseline experiments are not built by any genetic modification. They are randomly constructed so that all individuals in the search space are equally likely, just as the initial population in the evolutionary search. All the networks were trained from scratch using the training settings in section 5.3.2.

In Table 5.2, the results show an average accuracy of 81.4% with 7.7% standard deviation over the five randomly generated networks from ‘Pair 1’ chromosomes (Table 5.1), and an average accuracy of 85.9% with 7.2% standard deviation over the five randomly generated networks from ‘Pair 2’ chromosomes (Table 5.1). The large variation of accuracy among the networks resembles the randomized behaviour in the earlier generations of the evolutionary approach (*cf.* Fig. 5.7). The best baseline network is evolved from the ‘Pair 2’ chromosomes. It achieved 87.5% mean  $F_1$  score and 90.3% overall accuracy as shown in Table 5.2. The evolutionary approach is compared favourably against the baseline (see Table 5.3). However, the baseline is competitive with the Vaihingen contest leaderboard<sup>3</sup> state-of-the-art models, which demonstrates the effectiveness of our proposed encoding scheme of the search space.

Table 5.2: Baseline results on Vaihingen datasets. The values are the overall accuracy of the ten baseline networks, which five each were randomly generated from the two pairs of chromosomes settings in Table 5.1. Boldface is the best results.

	Overall Accuracy (%)				
	1	2	3	4	5
Pair 1	87.5	71.3	89.3	86.0	72.8
Pair 2	71.6	<b>90.3</b>	90.0	89.7	88.2

## 5.4 Results and Discussion

After the search, the top four networks found, which are denoted as SLGENet- $i$  in the Tables 5.3, 5.4 and 5.5, are trained on the three ISPRS datasets from scratch to full convergence using the training settings in Section ?? to precisely measure their effectiveness. The networks denoted as SLGENet-1 and SLGENet-2 are evolved from the pair of chromosomes ‘Pair 1’, and SLGENet-3 and SLGENet-4 are evolved from the pair of chromosomes ‘Pair 2’ in Table 5.1. The best network discovered is SLGENet-3. Fig. 5.3 is the structural representation of the pair of cells used to built the SLGENet-3. We report per-class  $F_1$  scores and overall pixel-wise classification accuracy on Vaihingen and Potsdam datasets as in Audebert *et al.* [24] and in conformity to the ISPRS 2D

<sup>3</sup> <https://www2.isprs.org/commissions/comm2/wg4/results/vaihingen-2d-semantic-labeling-contest/>

semantic labelling contest [73]. In addition, we report the mean  $F_1$  over all classes. On UAVid, we report per-class intersection-over-union (IoU) scores (also know as Jaccard Index) and the mean IoU over all classes as proposed by Lyu *et al.*[66], in addition to the overall accuracy. We compared and discussed the results with the state-of-the-arts models. Additionally, we compared the number of parameters.

### 5.4.1 ISPRS Vaihingen

In Table 5.3, we report the results on the Vaihingen test set of the top four networks. The best network, SLGENet-3, achieved mean  $F_1$  score of 91.1% and accuracy of 92.1% with 24.8M parameters. Fig. 5.3 depicts the pair of cells (normal and ASPP) of the best network. Compared with the baseline, the best network has an advantage of 3.6% in mean  $F_1$  score and 1.8% in the overall accuracy. We further compared with several state-of-the-art models to evaluate their performance. Among the state-of-the-art models in Table 5.3, CASIA2 achieved the best results in the mean  $F_1$  score and the overall accuracy. Our best network, SLGENet-3, improves the performance of CASIA2 by 1.0% in both the mean  $F_1$  score and the overall accuracy. However, with respect to the impervious surface, buildings and cars classes, our SLGENet-3 trailed behind HRNet, CASIA2 and SiameseDenseU-Net by 0.7%, 0.3% and 0.9%  $F_1$  scores respectively. In general, the top four networks improve the performance of the state-of-the-art models by the least of 0.5% in both the mean  $F_1$  score and the overall accuracy. Most of the state-of-the-art works did not report the number of parameters of their models. Nevertheless, comparing the number of parameters with their base networks such as VGG, SegNet, U-Net and FCN, our networks use relatively fewer parameters to improve the performance of the state-of-the-art models. Examples of semantic segmentation results on the Vaihingen test set with the SLGENet-3 is presented in the first row of Fig. 5.8.

Table 5.3: Comparison between our method and the baseline and the state-of-the-art models (NAS and handcrafted) on the Vaihingen test set. The values are the per-class  $F_1$  scores, mean  $F_1$  and overall pixel-wise classification accuracy. Boldface is the best result.

Method (Model)	Params	$F_1$ score (%)					mean $F_1$ (%)	Accuracy (%)	
		Imp. surf.	Buildings	Low veg.	Trees	Cars			
PDN [74] <sup>‡</sup>	–	88.6	92.7	78.4	88.1	81.2	85.8	87.3	
DCNNs [18]	132M <sup>†</sup>	91.5	94.7	81.9	88.5	74.0	86.1*	89.2	
SiameseDenseU-Net [26]	60.0M <sup>†</sup>	92.1	95.6	80.0	88.5	<b>91.3</b>	89.5	89.5	
V-FuseNet [24]	30.0M <sup>†</sup>	91.0	94.4	84.5	89.9	86.3	89.2*	90.0	
HRNet [75]	65.9M <sup>†</sup>	<b>94.7</b>	92.9	83.2	88.9	84.3	88.8	90.1	
GSN [65]	44.5M <sup>†</sup>	92.2	95.1	83.7	89.9	82.4	88.7	90.3	
DLR_9 [23]	88.0M	92.4	95.2	83.9	89.9	81.2	88.5*	90.3	
FCN_MFS_DSMBkd [76]	138M <sup>†</sup>	92.3	95.8	83.8	89.6	86.4	89.6*	90.6	
CASIA2 [64]	138M <sup>†</sup>	93.2	<b>96.0</b>	84.7	89.9	86.7	90.1*	91.1	
Baseline (Random search)	20.6M	92.1	94.0	83.2	90.0	78.5	87.5	90.3	
Our method (Evo. NAS)	SLGENet-1	24.7M	93.3	95.3	84.7	91.2	88.4	90.6	91.7
	SLGENet-2	25.5M	93.6	95.2	84.3	91.0	89.7	90.7	91.6
	SLGENet-3	24.8M	94.0	95.7	84.8	<b>91.5</b>	89.4	<b>91.1</b>	<b>92.1</b>
	SLGENet-4	23.9M	93.9	95.7	<b>85.0</b>	91.2	89.9	<b>91.1</b>	92.0

<sup>‡</sup>The state-of-the-art NAS method, the other state-of-the-art models are handcrafted methods

<sup>†</sup>The number of parameters of the base networks including SegNet, U-Net, and FCNs adopted in the state-of-the-art models

\*These values are calculated from the results of the authors

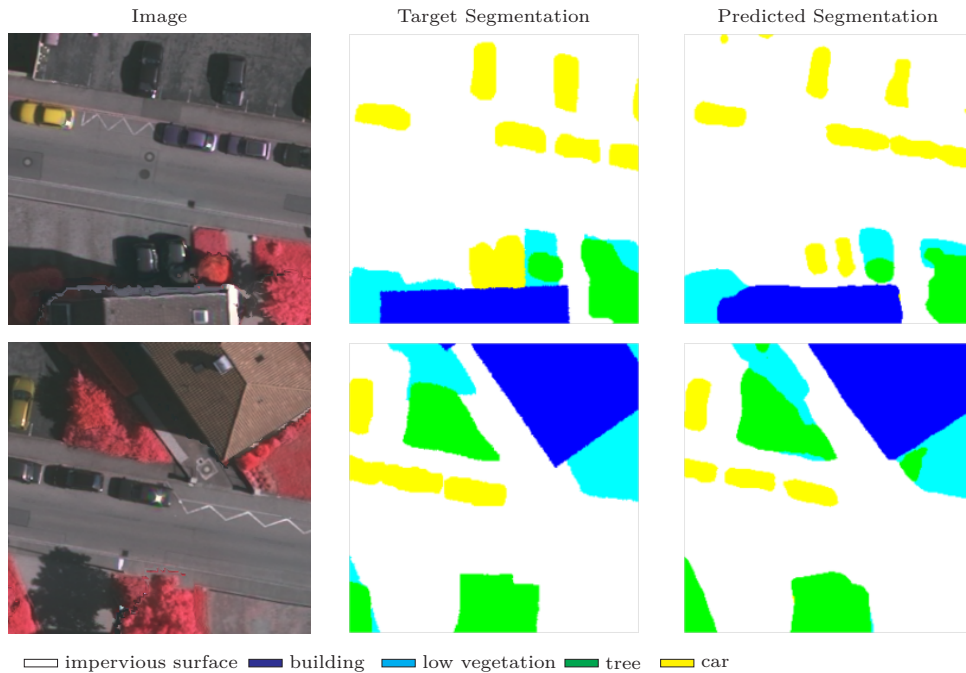


Figure 5.8: Examples of semantic segmentation results on ISPRS Vaihingen dataset.  $256 \times 256$  image patches of Vaihingen area 2. The predicted segmentation maps are produced by the best network, SLGENet-3.

#### 5.4.2 ISPRS Potsdam

Table 5.4 presents the results of our top four networks on the Potsdam test set in comparison to state-of-the-art networks. The best network on the Potsdam dataset is the same one we found on the Vaihingen dataset. With 24.8M parameters, the best network, SLGENet-3, achieved 94.4% in the mean  $F_1$  score and 94.2% in the overall accuracy on the Potsdam dataset. Compared with the SSNet-50, which is the best model among the state-of-the-art models in Table 5.4, our best network shows an improvement of 2.6% and 1.5% in the mean  $F_1$  score and the overall accuracy respectively. However, our SLGENet-3 lagged behind SSNet-50 by 1.5%  $F_1$  score with respect to the impervious surface class. This means, the best network was not able to improve the performance on the impervious surface class in the Potsdam dataset, just as in the Vaihingen dataset. With the exception of SLGENet-2, our networks improve the performance of the state-of-the-art models by the least of 0.5% in both the mean  $F_1$  score and the overall accuracy using fewer parameters. Though, our SLGENet-2 trailed behind SSNet-50 in the overall accuracy, it competes with SSNet-50 in the mean  $F_1$  score, and improves the performance of the other state-of-the-art models using fewer parameters. We present examples of semantic segmentation results on the Potsdam test set with the SLGENet-3 in the second row of Fig. 5.9.

Table 5.4: Comparison between our method and the state-of-the-art handcrafted models on the Potsdam test set. The values are the per-class  $F_1$  scores, mean  $F_1$  and overall pixel-wise classification accuracy. Boldface is the best result.

Method (Model)	Params	$F_1$ score (%)					mean $F_1$ (%)	Accuracy (%)	
		Imp. surf.	Buildings	Low veg.	Trees	Cars			
DCNNs [18]	132M <sup>†</sup>	92.3	97.0	86.8	86.9	94.5	91.5*	90.1	
DST_5 [77]	132M <sup>†</sup>	92.5	96.4	86.7	88.0	94.7	91.7*	90.3	
V-FuseNet [24]	30.0M <sup>†</sup>	92.7	96.3	87.3	88.5	95.4	92.0*	90.6	
Multi-filter CNN [78]	30.0M <sup>†</sup>	90.9	96.9	76.0	73.8	86.9	84.9*	90.6	
CASIA2 [64]	138M <sup>†</sup>	93.3	97.0	87.7	88.4	96.2	92.5*	91.1	
HRNet [75]	65.9M <sup>†</sup>	93.8	97.5	87.8	88.8	96.0	92.8	91.5	
SSNet-50 [79]	30.0M <sup>†</sup>	<b>97.4</b>	94.6	89.6	89.1	93.7	92.9	92.8	
Our method (Evo. NAS)	SLGENet-1	24.7M	95.5	97.3	89.6	89.7	98.1	94.0	93.8
	SLGENet-2	25.5M	94.4	96.2	88.6	88.7	97.4	93.0	92.3
	SLGENet-3	24.8M	95.9	<b>97.7</b>	<b>90.0</b>	<b>90.2</b>	98.2	<b>94.4</b>	<b>94.2</b>
	SLGENet-4	23.9M	95.8	<b>97.7</b>	89.9	90.0	<b>98.3</b>	94.3	94.1

<sup>†</sup>The number of parameters of the base networks including SegNet, VGG, and FCNs adopted in the state-of-the-art models

\*These values are calculated from the results of the authors

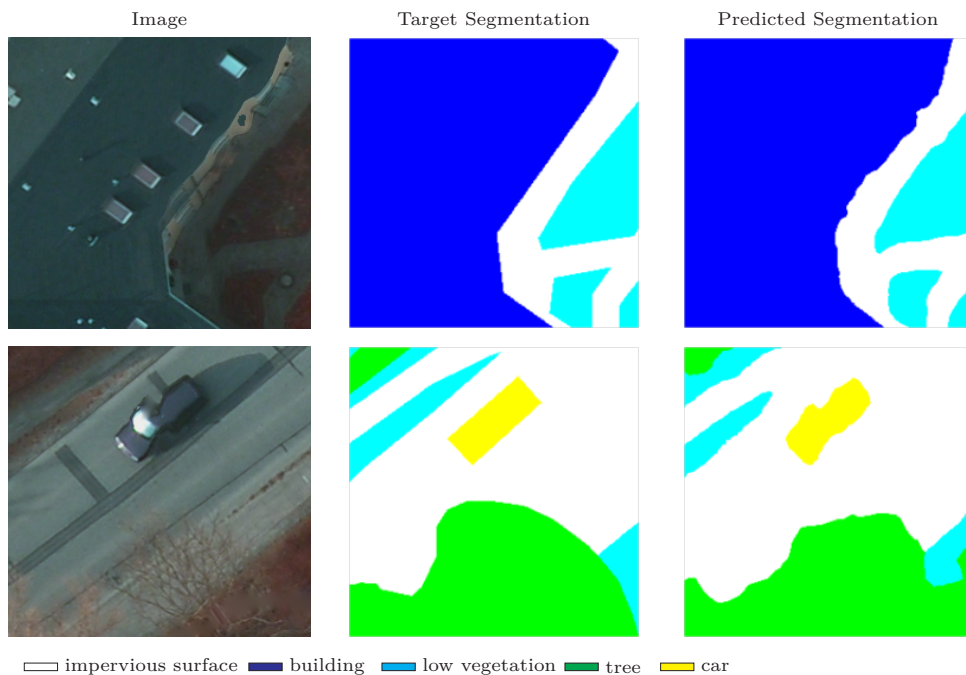


Figure 5.9: Examples of semantic segmentation results on ISPRS Potsdam dataset.  $256 \times 256$  image patches of Potsdam area 3\_13. The predicted segmentation maps are produced by the best network, SLGENet-3.

### 5.4.3 ISPRS UAVid

To further evaluate the effectiveness of the top four networks, the networks are also trained on the new high-resolution ISPRS dataset for UAV imagery semantic segmentation. In Table 5.5, we report the results on the UAVid validation set which was only used for testing in this work, since the target maps of the test set is not publicly available. We compared with the results reported by the authors [66] of the UAVid dataset and the online leaderboard state-of-the-art results. Among the authors' models in Table 5.5, the MS-Dilation+PRT+FSO achieved the best mean IoU score of 50.1%. Our best network, SLGENet-3, with 24.8M parameters significantly improves the performance by 19.5% mean IoU score. Additionally, SLGENet-3 and the runner-up

network, SLGENet-4, achieved an overall accuracy of 87.8%, placing our networks amongst the best of the online leaderboard state-of-the-art. Although, our best network, SLGENet-3, lagged behind the best online leaderboard results (KangyangWU) in Table 5.5, SLGENet-3 improves the performance of KangyangWU by 1.1%, 1.0% and 7.0% IoU scores on building, low vegetation and human classes respectively. On the whole, the top four networks compete with the online leaderboard state-of-the-art. They significantly improve the performance of Lyu *et al.*[66] by at least 18.9% mean IoU score using fewer parameters. Examples of semantic segmentation results on the UAVid dataset with the SLGENet-3 is presented in the third row of Fig. 5.10.

Table 5.5: Comparison between our method and the results reported in Lyu *et al.* [66] and the online leaderboard on the UAVid dataset. The values are the per-class IoU, mean IoU and the overall pixel-wise classification accuracy. Boldface is the best result

Method (Model)	Params	IoU score (%)								mean IoU (%)	Accuracy (%)
		Building	Tree	Clutter	Road	Low veg.	Static car	Moving car	Human		
U-Net+PRT+FSD [66]	30.0M <sup>†</sup>	79.0	73.8	46.4	65.3	43.5	26.8	56.6	0.0	48.9	-
MS-Dilation+PRT [66]	132M <sup>†</sup>	79.7	74.6	44.9	65.9	46.1	21.8	57.2	8.0	49.8	-
MS-Dilation+PRT+FSD [66]	132M <sup>†</sup>	80.9	75.5	46.3	66.7	47.9	22.3	56.9	4.2	50.1	-
Clairvoyance*	-	88.6	80.1	70.1	66.9	<b>81.7</b>	72.6	67.9	23.5	69.0	87.6
YeLyu*	-	88.6	80.4	66.9	81.6	61.6	<b>75.6</b>	77.2	31.3	70.1	87.2
KangyangWU*	-	90.2	<b>81.9</b>	<b>71.5</b>	<b>82.9</b>	66.4	67.7	<b>77.7</b>	33.2	<b>71.4</b>	<b>88.3</b>
Our method (Evo. NAS)	SLGENet-1 24.7M	90.9	76.7	63.4	78.9	66.9	66.9	74.5	39.4	69.0	87.5
	SLGENet-2 25.5M	91.2	76.8	64.1	79.2	67.4	67.7	74.6	39.9	69.5	87.7
	SLGENet-3 24.8M	91.3	77.1	64.4	79.4	67.4	67.7	74.4	<b>40.2</b>	69.6	87.8
	SLGENet-4 23.9M	<b>91.4</b>	77.1	64.4	79.2	67.3	66.5	73.6	<b>40.2</b>	69.4	87.8

\*Online leaderboard state-of-the-art results on UAVid dataset at <https://competitions.codalab.org/competitions/25224#results>

<sup>†</sup>The number of parameters of the base networks (U-Net and FCNs) of the models

Note: The Lyu *et al.*[66] models and the online leaderboard models are handcrafted methods

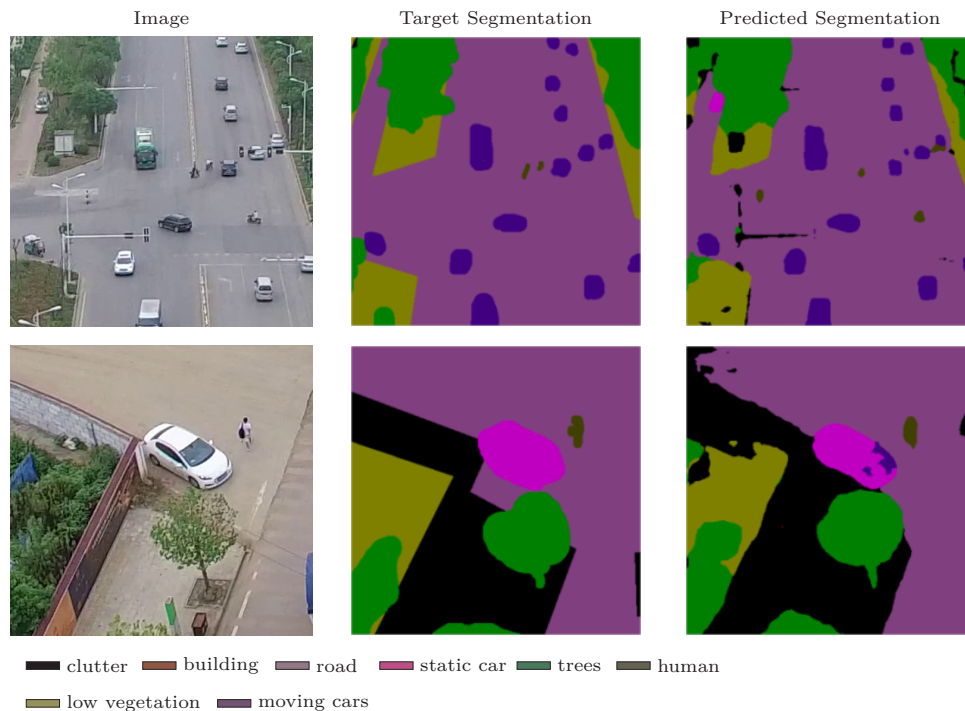


Figure 5.10: Examples of semantic segmentation results on ISPRS UAVid dataset.  $400 \times 400$  image patches of UAVid seq37/000300.png. The predicted segmentation maps are produced by the best network, SLGENet-3.



## 5.5 Summary

In this chapter, an evolutionary neural architecture search with SLGE architecture representation is explored for aerial/satellite image semantic segmentation. The evolutionary-based SLGE architecture search for semantic labelling task in remote sensing image analysis was achieved through (1) the construction of cell-based search space by leveraging the complementary strengths of gene expression programming and cellular encoding and (2) the joint search for normal and atrous spatial pyramid pooling cells with shortcut and multi-branch connections as building-blocks to construct modularized encoder-decoder CNN architectures. We evaluate the result of the search, SLGENet, on three aerial image semantic segmentation benchmarks, with comparison to the recent state-of-the-art systems. On ISPRS Vaihingen and Potsdam 2D semantic labelling challenges, SLGENet achieved performance slight gains in the overall accuracy. On the ISPRS UAVid segmentation benchmark for UAV imagery, SLGENet significantly improved the performance and the results position SLGENet amongst the state-of-the-art. The next chapter presented the summary and concluding remarks of the work presented in this dissertation.

## 5.6 References

- [1] L. Zhang, L. Zhang, and B. Du, “Deep learning for remote sensing data: A technical tutorial on the state of the art,” *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 22–40, 2016.
- [2] X. X. Zhu, D. Tuia, L. Mou, G. Xia, L. Zhang, F. Xu, and F. Fraundorfer, “Deep learning in remote sensing: A comprehensive review and list of resources,” *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 8–36, 2017.
- [3] T. Hofer, F. Bachofer, and C. Kuenzer, “Object detection and image segmentation with deep learning on earth observation data: A review—part ii: Applications,” *Remote Sensing*, vol. 12, no. 18, p. 3053, 2020.
- [4] K. S. Fu, D. A. Landgrebe, and T. L. Phillips, “Information processing of remotely sensed agricultural data,” *Proceedings of the IEEE*, vol. 57, no. 4, pp. 639–653, 1969.
- [5] X. Yuan, J. Shi, and L. Gu, “A review of deep learning methods for semantic segmentation of remote sensing imagery,” *Expert Systems with Applications*, vol. 169, p. 114417, 2021.
- [6] Y. J. Zhang, “Evaluation and comparison of different segmentation algorithms,” *Pattern Recognition Letters*, vol. 18, no. 10, pp. 963–974, 1997.
- [7] A. P. Carleer, O. Debeir, and E. Wolff, “Comparison of very high spatial resolution satellite image segmentations,” in *Image and Signal Processing for Remote Sensing IX*, vol. 5238, 2004, pp. 532–542.
- [8] A. Carleer, O. Debeir, and E. Wolff, “Assessment of very high spatial resolution satellite image segmentations,” *Photogrammetric Engineering & Remote Sensing*, vol. 71, no. 11, pp. 1285–1294, 2005.
- [9] J. A. Benediktsson, J. A. Palmason, and J. R. Sveinsson, “Classification of hyperspectral data from urban areas based on extended morphological profiles,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 3, pp. 480–491, 2005.

- 
- [10] Liangpei Zhang, Xin Huang, Bo Huang, and Pingxiang Li, "A pixel shape index coupled with spectral information for classification of high spatial resolution remotely sensed imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 10, pp. 2950–2961, 2006.
- [11] X. Huang and L. Zhang, "Morphological building/shadow index for building extraction from high-resolution imagery over urban areas," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 1, pp. 161–172, 2012.
- [12] L. Zhang, L. Zhang, D. Tao, and X. Huang, "On combining multiple features for hyperspectral remote sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 3, pp. 879–893, 2012.
- [13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [14] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.
- [15] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, "Deep learning in remote sensing applications: A meta-analysis and review," *ISPRS journal of photogrammetry and remote sensing*, vol. 152, pp. 166–177, 2019.
- [16] N. Audebert, A. Boulch, B. Le Saux, and S. Lefèvre, "Distance transform regression for spatially-aware deep semantic segmentation," *Computer Vision and Image Understanding*, vol. 189, p. 102809, 2019.
- [17] M. Carvalho, B. Le Saux, P. Trouvé-Peloux, F. Champagnat, and A. Almansa, "Multitask learning of height and semantics from aerial images," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 8, pp. 1391–1395, 2020.
- [18] D. Chai, S. Newsam, and J. Huang, "Aerial image semantic segmentation using DCNN predicted distance maps," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 161, pp. 309–322, 2020.
- [19] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.
- [20] M. Volpi and D. Tuia, "Dense semantic labeling of subdecimeter resolution images with convolutional neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 2, pp. 881–893, 2017.
- [21] G. Fu, C. Liu, R. Zhou, T. Sun, and Q. Zhang, "Classification for High Resolution Remote Sensing Imagery Using a Fully Convolutional Network," *Remote Sensing*, vol. 9, no. 5, 2017.
- [22] G. Chen, C. Li, W. Wei, W. Jing, M. Woźniak, T. Blažauskas, and R. Damaševičius, "Fully Convolutional Neural Network with Augmented Atrous Spatial Pyramid Pool and Fully Connected Fusion Path for High Resolution Remote Sensing Image Segmentation," *Applied Sciences*, vol. 9, no. 9, 2019.
- [23] D. Marmanis, K. Schindler, J. D. Wegner, S. Galliani, M. Datcu, and U. Stilla, "Classification with an edge: Improving semantic image segmentation with boundary detection," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 135, pp. 158–172, 2018.
- [24] N. Audebert, B. Le Saux, and S. Lefèvre, "Beyond RGB: Very high resolution urban remote sensing with multimodal deep networks," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 140, pp. 20–32, 2018.
- [25] R. Li, W. Liu, L. Yang, S. Sun, W. Hu, F. Zhang, and W. Li, "Deepunet: A deep fully convolutional network for pixel-level sea-land segmentation," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, pp. 3954–3962, 2018.

- [26] R. Dong, L. Bai, and F. Li, "SiameseDenseU-Net-based Semantic Segmentation of Urban Remote Sensing Images," *Mathematical Problems in Engineering*, vol. 2020, p. 1515630, 2020.
- [27] F. I. Diakogiannis, F. Waldner, P. Caccetta, and C. Wu, "ResUNet-a: A deep learning framework for semantic segmentation of remotely sensed data," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 162, pp. 94–114, 2020.
- [28] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015, pp. 234–241.
- [29] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [30] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," *arXiv preprint arXiv:1412.7062*, 2014.
- [31] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.
- [32] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," in *Computer Vision – ECCV 2018*, 2018, pp. 833–851.
- [33] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, "DenseASPP for semantic segmentation in street scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3684–3692.
- [34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Conference on CVPR*, 2016, pp. 770–778.
- [35] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on CVPR*. IEEE, 2017, pp. 2261–2269.
- [36] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.
- [37] Y. Liu, S. Piramanayagam, S. T. Monteiro, and E. Saber, "Dense semantic labeling of very-high-resolution aerial imagery and lidar with fully-convolutional neural networks and higher-order CRFs," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 1561–1570.
- [38] Z. Li, R. Wang, W. Zhang, F. Hu, and L. Meng, "Multiscale features supported deeplabv3+ optimization scheme for accurate water semantic segmentation," *IEEE Access*, vol. 7, pp. 155 787–155 804, 2019.
- [39] J. Liu, Z. Wang, and K. Cheng, "An improved algorithm for semantic segmentation of remote sensing images based on deeplabv3+," in *Proceedings of the 5th International Conference on Communication and Information Processing*, 2019, p. 124–128.
- [40] M. R. Heffels and J. Vanschoren, "Aerial imagery pixel-level segmentation," 2020, arXiv:2012.02024v1.
- [41] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.
- [42] A. Baldominos, Y. Saez, and P. Isasi, "On the automated, evolutionary design of neural networks: past, present, and future," *Neural Computing and Applications*, vol. 32, no. 2, pp. 519–545, 2020.

- 
- [43] G. F. Miller, P. M. Todd, and S. U. Hegde, “Designing neural networks using genetic algorithms,” in *Proceedings of the Third International Conference on Genetic Algorithms*, 1989, p. 379–384.
- [44] T. Elsken, J. H. Metzen, and F. Hutter, “Neural architecture search: A survey,” *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.
- [45] M. Wistuba, A. Rawat, and T. Pedapati, “A survey on neural architecture search,” 2019, arXiv:1905.01392.
- [46] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Addison-Wesley Longman Publishing Co., Inc., 1989.
- [47] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.
- [48] E. Galván and P. Mooney, “Neuroevolution in deep neural networks: Current trends and future challenges,” 2020, arXiv:2006.05415.
- [49] L. Xie and A. Yuille, “Genetic cnn,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1388–1397.
- [50] Z. Chen, Y. Zhou, and Z. Huang, “Auto-creation of effective neural network architecture by evolutionary algorithm and resnet for image classification\*,” in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 3895–3900.
- [51] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, “Nsga-net: neural architecture search using multi-objective genetic algorithm,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 419–427.
- [52] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, “Automatically designing cnn architectures using the genetic algorithm for image classification,” *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3840–3854, 2020.
- [53] M. Suganuma, S. Shirakawa, and T. Nagao, “A genetic programming approach to designing convolutional neural network architectures,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, p. 497–504.
- [54] B. Evans, H. Al-Sahaf, B. Xue, and M. Zhang, “Evolutionary deep learning: A genetic programming approach to image classification,” in *2018 IEEE Congress on Evolutionary Computation (CEC)*, 2018, pp. 1–6.
- [55] Y. Bi, B. Xue, and M. Zhang, “An evolutionary deep learning approach using genetic programming with convolution operators for image classification,” in *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 3197–3204.
- [56] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” in *5th International Conference on Learning Representations*, 2017.
- [57] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in *IEEE Conference on CVPR*, 2018, pp. 8697–8710.
- [58] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” in *Proceedings of the AAAI conference on Artificial Intelligence*, vol. 33, 2019, pp. 4780–4789.
- [59] C. Ferreira, *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*. Springer, 2006.
- [60] L.-C. Chen, M. D. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens, “Searching for efficient multi-scale architectures for dense image prediction,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, p. 8713–8724.
- [61] H. Liu, K. Simonyan, and Y. Yang, “DARTS: Differentiable architecture search,” in *Proceedings of the International Conference on Learning Representations*. ICLR, 2019.

- [62] J. Vasconcelos, J. Ramirez, R. Takahashi, and R. Saldanha, "Improvements in genetic algorithms," *IEEE Transactions on Magnetics*, vol. 37, no. 5, pp. 3414–3417, 2001.
- [63] J. Liu, S. Wang, X. Hou, and W. Song, "A deep residual learning serial segmentation network for extracting buildings from remote sensing imagery," *International Journal of Remote Sensing*, vol. 41, no. 14, pp. 5573–5587, 2020.
- [64] Y. Liu, B. Fan, L. Wang, J. Bai, S. Xiang, and C. Pan, "Semantic labeling in very high resolution images via a self-cascaded convolutional neural network," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 145, pp. 78–95, 2018.
- [65] H. Wang, Y. Wang, Q. Zhang, S. Xiang, and C. Pan, "Gated Convolutional Neural Network for Semantic Segmentation in High-Resolution Images," *Remote Sensing*, vol. 9, no. 5, 2017.
- [66] Y. Lyu, G. Vosselman, G.-S. Xia, A. Yilmaz, and M. Y. Yang, "UAVid: A semantic segmentation dataset for UAV imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 165, pp. 108 – 119, 2020.
- [67] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.
- [68] J. Howard and S. Gugger, "Fastai: A layered api for deep learning," *Information*, vol. 11, no. 2, p. 108, 2020.
- [69] L. N. Smith, "A disciplined approach to neural network hyperparameters: Part1–learning rate, batch size, momentum, and weight decay," 2018, arXiv:1803.09820 [cs.LG].
- [70] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.
- [71] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.
- [72] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann, "Evaluating the search phase of neural architecture search," in *International Conference on Learning Representations*, 2020.
- [73] F. Rottensteiner, G. Sohn, M. Gerke, and J. D. Wegner, "ISPRS 2D Semantic Labeling Contest." [Online]. Available: <https://www2.isprs.org/commissions/comm2/wg4/benchmark/semantic-labeling/>
- [74] J. Li, W. Diao, X. Sun, Y. Feng, W. Zhang, Z. Chang, and K. Fu, "Automated and lightweight network design via random search for remote sensing image scene classification," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLIII-B2-2020, pp. 1217–1224, 2020.
- [75] J. Zhang, S. Lin, L. Ding, and L. Bruzzone, "Multi-scale context aggregation for semantic segmentation of remote sensing images," *Remote Sensing*, vol. 12, no. 4, p. 701, 2020.
- [76] W. Sun and R. Wang, "Fully convolutional networks for semantic segmentation of very high resolution remotely sensed images combined with dsm," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 3, pp. 474–478, 2018.
- [77] J. Sherrah, "Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery," 2016, arXiv:1606.02585.
- [78] Y. Sun, X. Zhang, Q. Xin, and J. Huang, "Developing a multi-filter convolutional neural network for semantic segmentation using high-resolution aerial imagery and LiDAR data," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 143, pp. 3–14, 2018.

- [79] J. Liu, S. Wang, X. Hou, and W. Song, "A deep residual learning serial segmentation network for extracting buildings from remote sensing imagery," *International Journal of Remote Sensing*, vol. 41, no. 14, pp. 5573–5587, 2020.

## Chapter 6

# Conclusions and Future Work

*The coming era of Artificial Intelligence will not be the era of war, but be the era of deep compassion, non-violence, and love.*

—Amit Ray

(Compassionate Artificial Intelligence)

The research work presented in this dissertation investigates whether the complementary strengths of gene expression programming (GEP) [1] and cellular encoding (CE) [2] can be combined into a new representation scheme to evolve modularized architectures for deep neural networks (DNNs), in particular, for convolutional neural networks (CNNs). The novel encoding scheme called *symbolic linear generative encoding* (SLGE) is introduced and implemented for automated neural architecture search (NAS) in visual perception tasks. The research was conducted towards the applications in remote sensing (RS) image understanding to illustrate the capability of the SLGE in the architecture search of CNNs. This chapter provides the summary of the main results of the work presented in this dissertation and how they contribute to the achievement of the research aims and objectives expressed in the introductory chapter. Finally, the chapter highlights some future research directions to further extend this line of research.

### 6.1 SLGE: Design and Development

The work in this dissertation harnessed the strengths of GEP and CE to produce a novel representation scheme called SLGE, that is capable of evolving modularized architectures for CNNs using random search and evolutionary algorithm. The SLGE was designed and implemented using the simplicity features of GEP with the modularity property of CE to construct the representation space of CNN architectures. SLGE embeds the graph grammar of CE into the genotype representation of GEP to represent candidate architectures as multi-gene chromosomes of linear fix-length strings and utilises the phenotype representation of CE. Additionally, SLGE

employs standard convolution and pooling operations as the basic search units to achieve flexibility in the search space as opposed to the sophisticated ResNet-block and DenseNet-block adopted in other studies [3–7]. This enables SLGE to evolve building-blocks with shortcut connections and multi-branch connections, similar to the commonly adopted ones by human experts [8–10], to develop modularized CNN architectures.

The preliminary experimental results presented in chapter 3 on general purpose image classification benchmarks, CIFAR-10 and CIFAR-100, indicate that the networks evolved by SLGE via evolutionary algorithm (and random search strategy as baseline) achieved a competitive performance with the state-of-the-art networks, and in some case improved the performance, using fewer number of parameters. The results suggest that SLGE is a viable and effective representation scheme for architecture search of CNNs for visual perception tasks. Also very important is the fact the networks evolved by SLGE have modules with shortcut and multi-branch connections which can improve training and performance of the networks. Therefore, the design and development of SLGE and the preliminary experimental results obtained in general purpose image classification, have fulfilled the achievement of the first two research aims through the accomplishment of the first two research objectives that are expressed in the introductory chapter.

## 6.2 Extending SLGE to RS Image Understanding

In this work, to evaluate the SLGE representation space in terms of expressiveness and tractability for evolving modularized CNN architectures, and also to verify the performance and robustness of the architectures evolved by SLGE in visual perception tasks, we extended SLGE to RS image understanding. We used metaheuristic optimization algorithms, namely random search with early-stopping strategy and evolutionary algorithm approach, to evolve SLGE-based architectures for RS image understanding tasks. To accelerate the architecture search process, we developed a simple objective (fitness) function with the number of parameters of a network as a constraint and employed proxy-task approach to approximate the network performance during search without requiring a full training of the evolving networks. The networks evolved by SLGE were validated against various benchmarks in RS image understanding.

The random search with early-stopping strategy was adopted to explore the SLGE architecture representation to evolve architectures for CNNs in RS image scene classification. In classifying the scenes of multispectral satellite images and high-resolution RGB aerial images, SLGE networks achieved highly competitive results in both single-label and multi-label classification tasks. In fact, with fewer parameters, SLGE networks improved the performance of most related works and the results position our method amongst the best of the state-of-the-art (see the work presented in chapter 4). For a simple random search to discover well-performing networks show that the SLGE



representation space is expressive and not intractable.

Furthermore, the representation space of SLGE was improved to construct two-separate search space representation: normal cell search space and atrous spatial pyramid pooling (ASPP) cell search space. And we used evolutionary algorithm approach with genetic modification operators such as uniform mutation, two-point crossover and gene crossover to joint search for a normal cell and an ASPP cell as a pair of cells to evolve modularized encoder-decoder CNN architecture, dubbed SLGENet, for semantic segmentation of high-resolution satellite and UAV (unmanned aerial vehicle) images. The architectures of the best discovered networks on a satellite image benchmark was transferred to UAV and another satellite image benchmarks. SLGENet improved the performance of the state-of-the-art networks on both the satellite and UAV image benchmarks using a reasonable computational resources of 2.5 GPU days with fewer parameters (see the work presented in chapter 5). The results demonstrate the effectiveness and robustness of SLGENet on challenging ISPRS segmentation benchmarks.

The experimental results on various RS image understanding benchmarks presented in chapter 4 and chapter 5 contribute to the accomplishment of the last two research aims and objectives that are expressed in the introductory chapter. Just as important, extending SLGE to RS image understanding, we have introduced automated NAS into the RS community to help domain scientists to adopt DNNs for their work with little or without deep learning experience.

### 6.3 Future Work and Concluding Remarks

Within the current framework, related visual perception tasks such as instance segmentation and object detection might be plausible. Moreover, expanding the current cell-based search space to include a global search space might be beneficial to develop completely automatic architecture search of CNNs. Another work is implementing real-world AutoDNN system with SLGE to show its practical and commercial viability to interpret remotely-sensed imagery or any other computer vision related problem.

Finally, with the accomplishment of the research aims and objectives expressed in the introductory chapter, we hope the work in this dissertation would have a positive impact in future research in the field of automated NAS. The results presented were influenced by the world of remote sensing image understanding. Hopefully, this might also suggest something inspiring and opens new research path in such an interdisciplinary field, as well as practical applications in automated NAS method for real-world situations. This could improve the user experience and productivity in designing AI-enabled remote sensing applications for urban planning, land resource management, disaster management, environmental monitoring, and among others.

## 6.4 References

- [1] C. Ferreira, “Gene expression programming: a new adaptive algorithm for solving problems,” *Complex Systems*, vol. 13, no. 2, pp. 87–129, 2001.
- [2] F. Gruau, “Neural network synthesis using cellular encoding and the genetic algorithm.” Doctoral Dissertation, L’universite Claude Bernard-lyon I, 1994.
- [3] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, “Completely automated cnn architecture design based on blocks,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2019.
- [4] T. Hassanzadeh, D. L. Essam, and R. A. Sarker, “An evolutionary denseres deep convolutional neural network for medical image segmentation,” *IEEE Access*, vol. 8, pp. 212 298–212 314, 2020.
- [5] A. Rajagopal, G. P. Joshi, A. Ramachandran, R. T. Subhalakshmi, M. Khari, S. Jha, K. Shankar, and J. You, “A deep learning model based on multi-objective particle swarm optimization for scene classification in unmanned aerial vehicles,” pp. 135 383–135 393, 2020.
- [6] B. Wang, Y. Sun, B. Xue, and M. Zhang, “Evolving deep neural networks by multi-objective particle swarm optimization for image classification,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 490–498.
- [7] L. Yao, H. Xu, W. Zhang, X. Liang, and Z. Li, “SM-NAS: Structural-to-Modular Neural Architecture Search for Object Detection,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, pp. 12 661–12 668, 2020.
- [8] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, p. 4278–4284.
- [9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.
- [10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

# Appendices

## Appendix A

### Source Code

The *Symbolic Linear Generative Encoding* (SLGE) is implemented on top of `geppy`<sup>1</sup> (a Python library for Gene Expression Programming) and `DEAP`<sup>2</sup> (an evolutionary computation framework for rapid prototyping developed in Python). SLGE extends the `geppy` to encode deep neural networks and `DEAP` provides the fundamental support for evolutionary computation. The `PyTorch`<sup>3</sup> deep learning library is used for the implementation and training of the deep neural networks that are automatically discovered via SLGE representation space.

The source code is hosted at [https://github.com/cliffbb/geppy\\_nn](https://github.com/cliffbb/geppy_nn). The code is intended to be used for teaching and academic research only. The source code is not well documented at the moment. However, in the near future, the author would provide a full documentation and user guide for easy use and possible extension of the code.

---

<sup>1</sup><https://github.com/ShuhuaGao/geppy>

<sup>2</sup><https://github.com/DEAP/deap>

<sup>3</sup><https://pytorch.org/>

# Appendix B

## Paper I

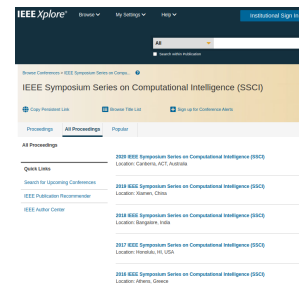
### Evolutionary NAS with Gene Expression Programming of Cellular Encoding

**Cliford Broni-Bediako, Yuki Murata, Luiz H. B. Mormille, Masayasu Atsumi**

Published in *Proceedings of IEEE Symposium Series on Computational Intelligence (SSCI)*, December 2020, pp. 2670–2676.

DOI: 10.1109/SSCI47803.2020.9308346.

© 2020 IEEE. Reprinted with permission from the publisher.



**Erratum:** The author “Cliford Broni-Bediako” should read “Clifford Broni-Bediako”.

### Paper Abstract

The renaissance of neural architecture search (NAS) has seen classical methods such as genetic algorithms (GA) and genetic programming (GP) being exploited for convolutional neural network (CNN) architectures. While recent work have achieved promising performance on visual perception tasks, the direct encoding scheme of both GA and GP has functional complexity deficiency and does not scale well on large architectures like CNN. To address this, we present a new generative encoding scheme—*symbolic linear generative encoding* (SLGE)—simple, yet a powerful scheme which embeds local graph transformations in chromosomes of linear fixed-length string to develop CNN architectures of variant shapes and sizes via an evolutionary process of gene expression programming. In experiments, the effectiveness of SLGE is shown in discovering architectures that improve the performance of the state-of-the-art handcrafted CNN architectures on CIFAR-10 and CIFAR-100 image classification tasks; and achieves a competitive classification error rate with the existing NAS methods using fewer GPU resources.

# Appendix C

## Paper II

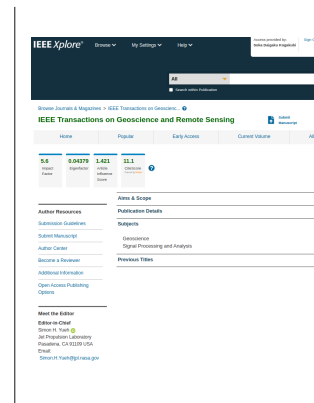
### Searching for CNN Architectures for Remote Sensing Scene Classification

Clifford Broni-Bediako, Yuki Murata, Luiz H. B. Mormille, Masayasu Atsumi

Published in *IEEE Transactions on Geoscience and Remote Sensing*, July 2021, pp. 1–13.

DOI: 10.1109/TGRS.2021.3097938.

© 2021 IEEE. Reprinted with permission from the publisher.



**Errata:** The cell block “[2, 2, 1, 21]” in Table II should read “[2, 2, 1, 2]” and the statement “...the overall search space with nodes between 2 and 16 in a cell...” at the end of Section A of the Methodology should read “...the overall search space with nodes between 3 and 16 in a cell...”.

### Paper Abstract

Convolutional neural network (CNN) models for remote sensing (RS) scene classification are largely built on pre-trained networks which are trained on the general-purpose ImageNet dataset in computer vision. The pre-trained networks can easily be adapted for transfer learning in RS scene classification. However, the accuracy of transfer learning may decline as RS images are considerably different from other images. Thus, the pre-trained CNN model learned on ImageNet may not be sufficient for the accurate classification of RS image scenes. Furthermore, most of the pre-trained models have large memory footprints which place a further burden on computational requirements. In this work, we explore SLGE-based random search with early-stopping in the search for CNN architectures for both single-label and multi-label RS scene classification tasks. In SLGE, the architecture search space is capable of representing multipath Inception-like modular cells with skip-connections similar to human-experts designs. The experimental results on four RS scene classification benchmarks show that the automatically discovered networks demonstrate the promising capability in classifying multispectral satellite image scenes compared with fine-tuned pre-trained CNN models. Using fewer parameters with 0.56B FLOPS, our best network achieves a classification accuracy rate of 96.56% and 96.10% on NWPU-RESISC45 single-label and AID single-label RGB aerial image datasets respectively, and classification accuracy rate of 99.76% and 93.89% on EuroSAT single-label and BigEarthNet multi-label multispectral satellite image datasets respectively. The results position our approach amongst the best of the state-of-the-art.

## Appendix D

### Paper III

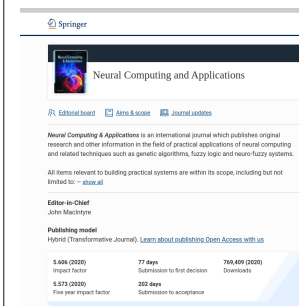
# Evolutionary NAS for Aerial Image Segmentation with Gene Expression Programming of Cellular Encoding

**Clifford Broni-Bediako, Yuki Murata, Luiz H. B. Mormille, Masayasu Atsumi**

Published in *Neural Computing and Applications*, October 2021.

DOI: 10.1007/s00521-021-06564-9.

© 2021 Springer Nature. Reprinted with permission from the publisher.



## Paper Abstract

Recently, neural architecture search (NAS) has gained a lot of attention as a tool for constructing deep neural networks automatically. NAS methods have successfully found convolutional neural networks (CNNs) that exceed human expert-designed networks on image classification in computer vision. However, there are growing demands for semantic segmentation in several areas including remote sensing image analysis. In this paper, we introduce an evolutionary NAS method for semantic segmentation of high-resolution aerial images. The proposed method leverages the complementary strengths of gene expression programming (GEP) and cellular encoding (CE) to develop an encoding scheme, called *symbolic linear generative encoding* (SLGE), for evolving cells (directed acyclic graphs) as building-blocks to construct modularized encoder-decoder CNNs via an evolutionary process. SLGE can evolve cells with multi-branch and shortcut connections similar to the Inception-ResNet-like modules which can improve training and inference performance in deep neural networks. In experiments, we demonstrate the effectiveness of the proposed method on the challenging ISPRS Vaihingen, Potsdam and UAVid semantic segmentation benchmarks. Compared with recent state-of-the-art systems, our network, dubbed SLGENet, improves the overall accuracy performance on Vaihingen and Potsdam; and achieves a competitive overall accuracy on UAVid using fewer parameters. Our method achieves promising results in a little time of 2.5 GPU days.

## Appendix E

### List of Publications

The work presented in this dissertation is based on the following publications:

**C. Broni-Bediako**, Y. Murata, L. H. B. Mormille, and M. Atsumi, “Evolutionary NAS for Aerial Image Segmentation with Gene Expression Programming of Cellular Encoding”. In: *Neural Computing and Applications*. 2021. DOI: 10.1007/s00521-021-06564-9.

**C. Broni-Bediako**, Y. Murata, L. H. B. Mormille, and M. Atsumi, “Searching for CNN Architectures for Remote Sensing Scene Classification”. In: *IEEE Transactions on Geoscience and Remote Sensing*. pp. 1–13, 2021. DOI: 10.1109/TGRS.2021.3097938.

**C. Broni-Bediako**, Y. Murata, L. H. B. Mormille, and M. Atsumi, “Evolutionary NAS with Gene Expression Programming of Cellular Encoding”. In: *Proceedings of 2020 IEEE Symposium Series on Computational Intelligence*. pp. 2670–2676, 2020. DOI: 10.1109/SSCI47803.2020.9308346.

In addition to the papers included in this dissertation, the author has also contributed to the following publications:

L. H. Mormille, **C. Broni-Bediako**, and M. Atsumi, “Regularizing Self-Attention on Vision Transformers with 2D Spatial Distance Loss”. In: *Proceedings of the Joint Symposium of The Twenty-Seventh International Symposium on Artificial Life and Robotics, The Seventh International Symposium on BioComplexity, and The Fifth International Symposium on Swarm Behavior and Bio-Inspired Robotics*. pp. 726–731, 2022.

E. B. Appiah, **C. Broni-Bediako**, K. Nagatsuka, and M. Atsumi, “Prediction of Topic Sentiment Polarity of Customers’ Reviews from Business Attributes using Dual-head MLP”. In: *Proceedings of the Joint Symposium of The Twenty-Seventh International Symposium on Artificial Life and Robotics, The Seventh International Symposium on BioComplexity, and The Fifth International Symposium on Swarm Behavior and Bio-Inspired Robotics*. pp. 653–657, 2022.

K. Nagatsuka, **C. Broni-Bediako**, M. Atsumi, “Pre-training a BERT with Curriculum Learning by Increasing Block-Size of Input Text”. In: *Proceedings of RANLP 2021: Recent Advances in Natural Language Processing*. pp. 993–1000, 2021. DOI: 10.26615/978-954-452-072-4\_113.



---

**C. Broni-Bediako**, F. A. Katsriku, T. Unemi, M. Atsumi, J.-D. Abdulai, N. Shinomiya, and E. Owusu, “El Niño-Southern Oscillation Forecasting using Complex Networks Analysis of LSTM Neural Networks”. In: *Artificial Life and Robotics*. Vol. 24, no. 4, pp. 445–451, 2019. DOI: 10.1007/s10015-019-00540-2.

**C. Broni-Bediako**, F. Apietu Katsriku, T. Unemi, N. Shinomiya, J.-D. Abdulai and M. Atsumi, “El Niño-Southern Oscillation Forecasting using Complex Networks Analysis of LSTM Neural Networks”. In: *Proceedings of the Joint Symposium of The Twenty-Third International Symposium on Artificial Life and Robotics and The Third International Symposium on BioComplexity*. pp. 100–105, 2018. DOI: 10.1007/s00521-021-06564-9.